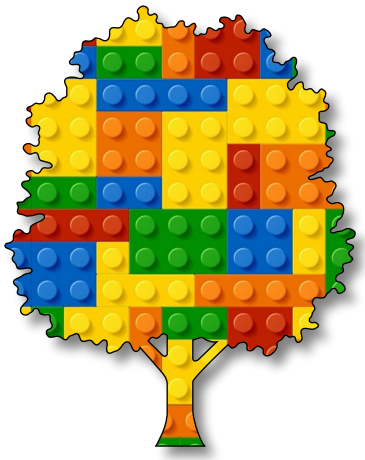
Algorithmic Art

Module F

Unit #2

HTML



Content

Module F Unit #2 HTML DOM Elements

Introduction to DOM elements

In short, text boxes, sliders, and buttons.

This unit covers a topic broadly called DOM elements that you might use in a website involving CSS, HTML, and JavaScript. Here, we will be looking at the index.html file as well as the sketch.js file, so we will be giving them a separate heading to identify them like so...

sketch.js

```

```

index.html

```

```

You can access the list of files as shown below in Fig. 1. Just click on the arrow in the grey box, and the list of files will open up on the left-hand side. Then click on the tab for index.html, and you should get the following. You will be able to see the following files...

index.html
sketch.js
style.css

It defaults to sketch.js. Notice in the above image that it includes other sketches which you will learn to add and use later on. Don't worry at this stage what all the lines of code mean; they are there because you need them to run p5.js. Have a close look and see what you can glean. But for now, it is only sketch.js and index.html that we are bothered about.

Challenge

Click on the style.css tab

Sketch F2.1 starting sketch

Starting with this standard sketch of drawing a circle with a grey background.

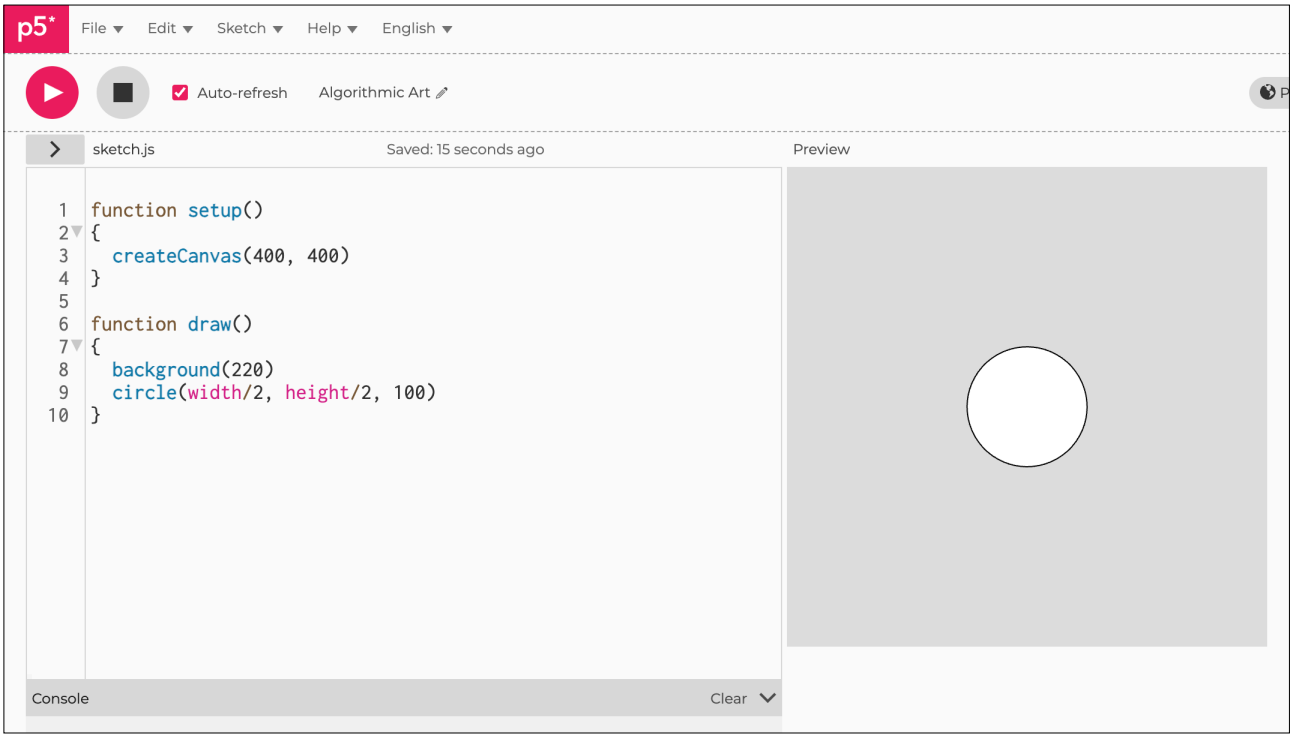
sketch.js

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

function draw()
{
  background(220)
  circle(width/2, height/2, 100)
}
```

Notes

Nothing very new here.

Figure F2.1



Sketch F2.2 creating html elements

Using the `mousePressed()` function to create an HTML event that is a header with the words `"This is a h2 header"`. Also creates a paragraph element for a random number.

sketch.js

```
function setup()
{
  createCanvas(400, 400)
  createElement('h2', 'This a h2 header')
  createP('Click the mouse to generate a random number')
}

function mousePressed()
{
  createP('a random number ' + floor(random(100)))
}

function draw()
{
  background(220)
  circle(width/2, height/2, 100)
}
```

🗒️ Notes

We have added `createElement()` which allows us to add header HTML in p5.js. It takes two arguments, `h1` to `h6`, and the text in speech marks. The `createP()` creates a paragraph. Notice that they are both added after the canvas, not in the canvas. You can easily interact with these DOM elements in the p5.js sketch.

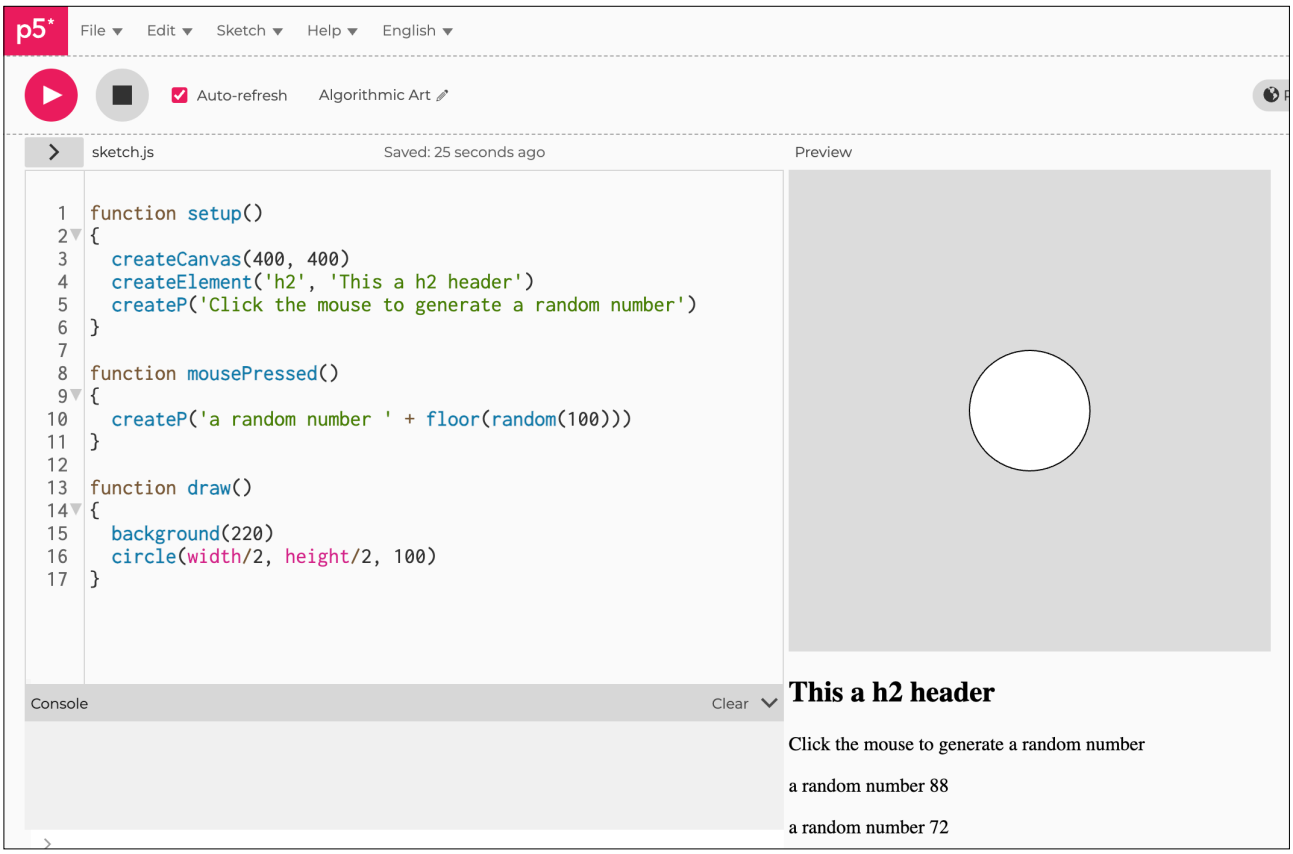
🌻 Challenges

1. Create more interactive paragraph elements
2. Use more heading values, i.e. cycle through the six different sizes of headings - maybe using a loop.

🛠️ Code Explanation

`createElement('h2', 'This a h2 header')`	Creates an element which is a header of size h2
`createP`	Creates a paragraph

Figure F2.2


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 25 seconds ago
Preview
function setup()
{
createCanvas(400, 400)
createElement('h2', 'This a h2 header')
createP('Click the mouse to generate a random number')
}
function mousePressed()
{
createP('a random number ' + floor(random(100)))
}
function draw()
{
background(220)
circle(width/2, height/2, 100)
}
Console
Clear
This a h2 header
Click the mouse to generate a random number
a random number 88
a random number 72

Sketch F2.3 position

❗ It's quicker and easier to start a new sketch.
There are a number of fun things happening here. It is to give you a taster of what you can do with HTML code.

sketch.js

```
let canvas
let heading
let x = 100
let y = 100

function setup()
{
  canvas = createCanvas(400, 400)
  canvas.position(200, 200)
  heading = createElement('h1', 'click on the mouse')
}

function mousePressed()
{
  heading.html('I am feeling a bit wobbly')
}

function draw()
{
  background(220)
  circle(x, y, 100)
  heading.position(x, y)
  x += random(-2, 2)
  y += random(-2, 2)
}
```

Notes

In this example, you can set the position of the canvas separate from the DOM element. Although the wobbly header follows the same x, y, the circle is relative to the top left-hand corner of the canvas (0, 0), and the wobbly text is relative to the top left-hand corner of the preview window (browser).

You create a variable called heading (it can be called anything) so that it will store the createElement(h1, 'Click on the mouse'). This can then be changed later to 'I'm feeling a bit wobbly'.

🌻 Challenges

1. Try differently named variables for heading and canvas.
2. Could you create another event other than changing the heading?

Figure F2.3

Sketch F2.4 starting sketch

❗ A new starting sketch. In this bit, we are going to look at adding buttons.

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

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

Notes

Bog standard sketch.

Sketch F2.5 background colour

First off, we will create a variable for the background colour.

```
let backgroundColour

function setup()
{
  createCanvas(400, 400)
  backgroundColour = color(220)
}

function draw()
{
  background(backgroundColour)
}
```

Notes

This creates exactly the same grey background as we had in the previous sketch.

Sketch F2.6 adding a button

Here we create a button. We don't give it a position, so it defaults to the bottom of the canvas.

```
let backgroundColour
let button

function setup()
{
  createCanvas(400, 400)
  backgroundColour = color(220)
  button = createButton('clcik on this button')
}

function draw()
{
  background(backgroundColour)
}
```

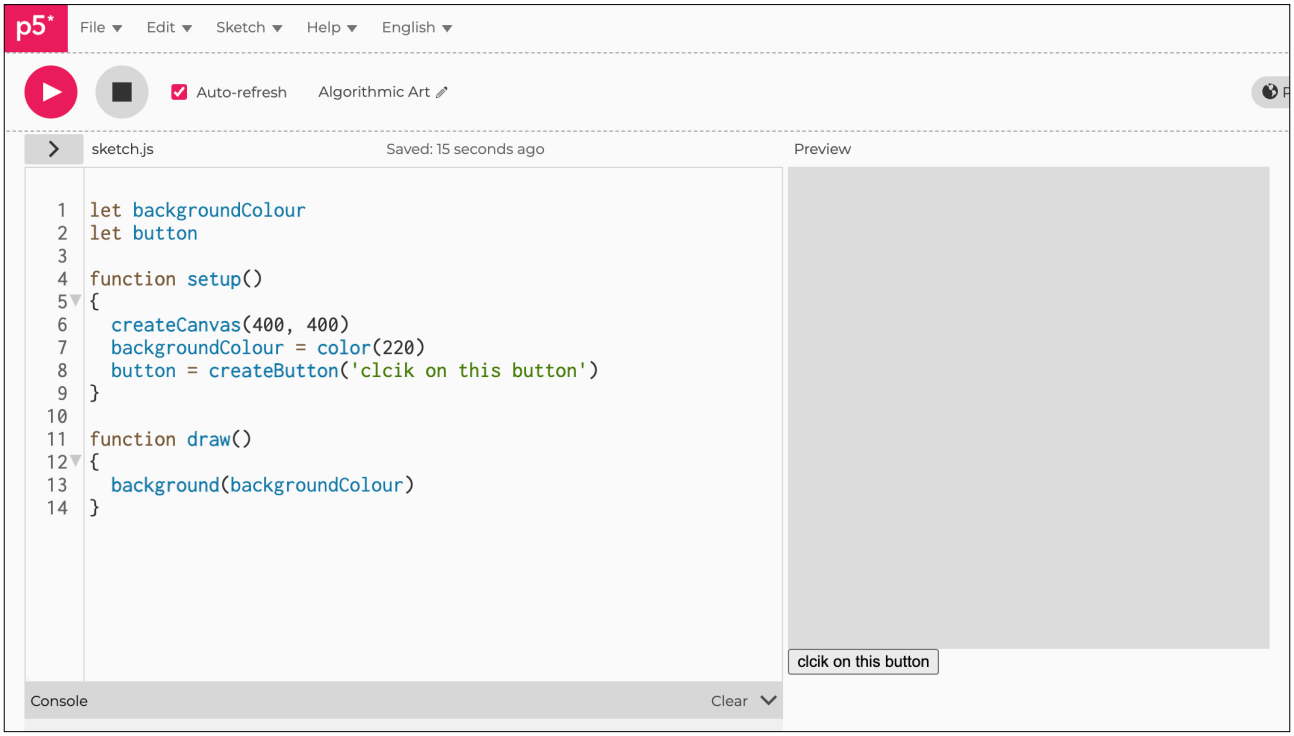
🗒️ Notes

This button doesn't do anything yet because we need to give it a call back, in other words, give it something to do when it is clicked. Also, you can change the position, the font size, the colour, and even the button background colour, but that is for another day.

🌻 Challenge

Change the text.

Figure F2.6


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
let backgroundColour
let button
function setup()
{
createCanvas(400, 400)
backgroundColour = color(220)
button = createButton('clcik on this button')
}
function draw()
{
background(backgroundColour)
}
clcik on this button
Console
Clear

Sketch F2.7 the callback

What we want to do in this instance is to change the background colour (random) every time we click on the button. For that, we need to create a function that is called when the button is pressed. We will call it `changeColour()`, isn't it original!

```
let backgroundColour
let button

function setup()
{
  createCanvas(400, 400)
  backgroundColour = color(220)
  button = createButton('clcik on this button')
  button.mousePressed(changeColour)
}

function changeColour()
{
  backgroundColour = color(random(255), random(255), random(255))
}

function draw()
{
  background(backgroundColour)
}
```

Notes

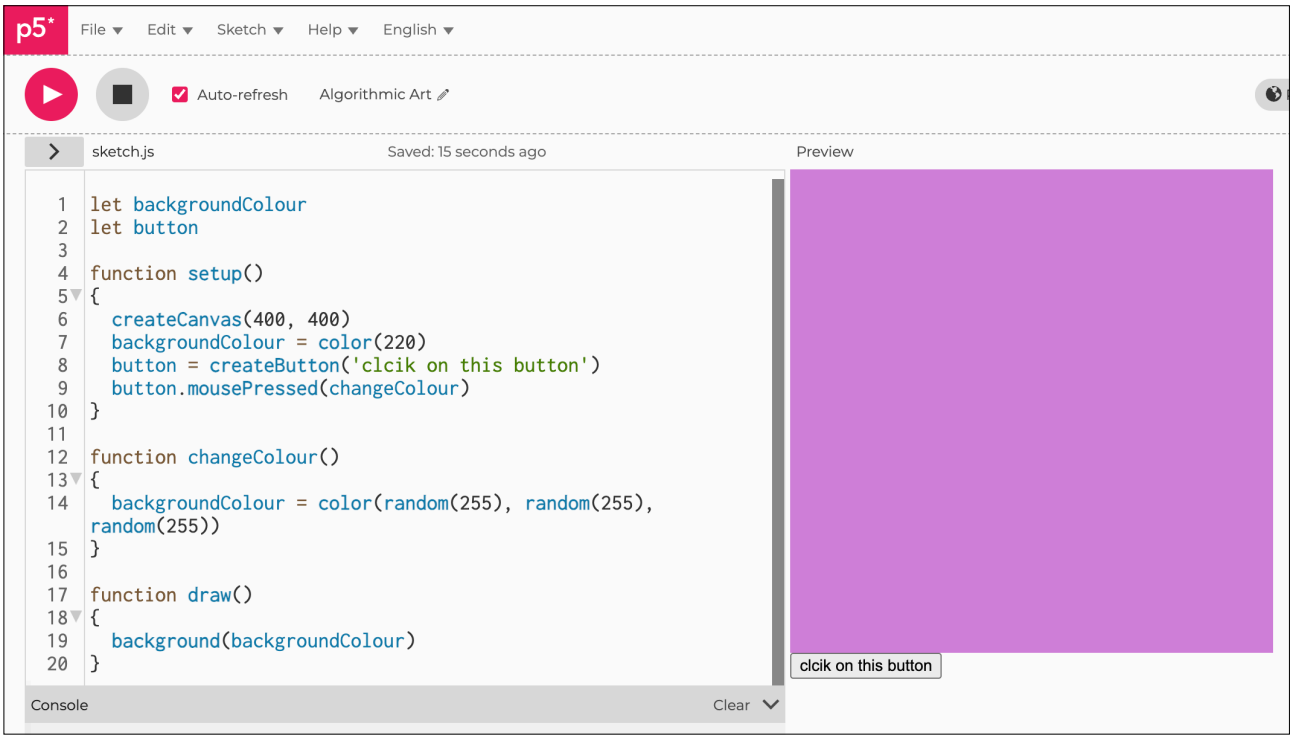
As you press the button, it calls a function called `changeColour()`. This creates a random background colour change, which is then executed in draw. So every time you click the button, it generates a new random background colour, which is sent to the draw function. The draw function loops continuously and updates the bicolour variable. The `color()` function is a special function that stores three arguments: the red, blue, and green elements.

Notice that the keyword in this `color()` is the American spelling; you cannot change that to the British spelling.

🌻 Challenge

1. Create another function on `button.mousePressed()` e.g. change the size or position of a circle.
2. Could you have a random number of circles or just add an extra one each time?

Figure F2.7



Sketch F2.8 slider

❗ Starting a new sketch.

We use the `createSlider()` function, and it returns a value depending on where the slider point is. The `createP()` function just creates the text instructions (a paragraph DOM element).

sketch.js

```
let slider

function setup()
{
  createCanvas(400, 400)
  rectMode(CENTER)
  createP('move the slider')
  slider = createSlider(1, 400, 100)
}

function draw()
{
  background(220)
  square(width/2, height/2, slider.value())
}
```

🗒️ Notes

Using a slider DOM element. It has three arguments:

1️⃣ the first one is the minimum value,

2️⃣ the second is the maximum value, and

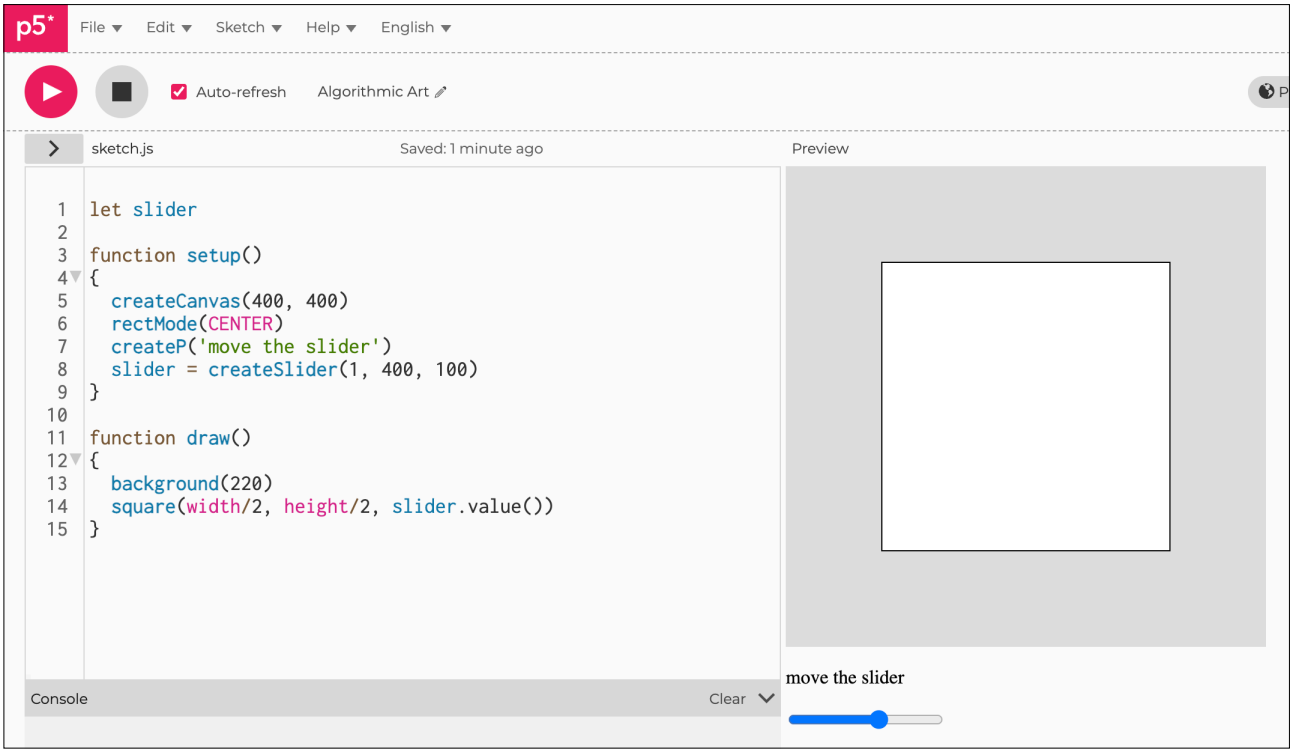
3️⃣ the third is the starting value.

It returns a single value between the minimum and maximum value as you slide the slider.

🌻 Challenges

1. Add more sliders for different effects.
2. Change the colour with the slider.

Figure F2.8


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 1 minute ago
Preview
let slider
function setup()
{
createCanvas(400, 400)
rectMode(CENTER)
createP('move the slider')
slider = createSlider(1, 400, 100)
}
function draw()
{
background(220)
square(width/2, height/2, slider.value())
}
Console
Clear
move the slider

Sketch F2.9 text box

You can use text from a text box as an input.

sketch.js

```
let input

function setup()
{
  createCanvas(400, 400)
  createP('Type your name here')
  input = createInput()
  textSize(40)
}

function draw()
{
  background(220)
  text(input.value(), 10, 50)
}
```

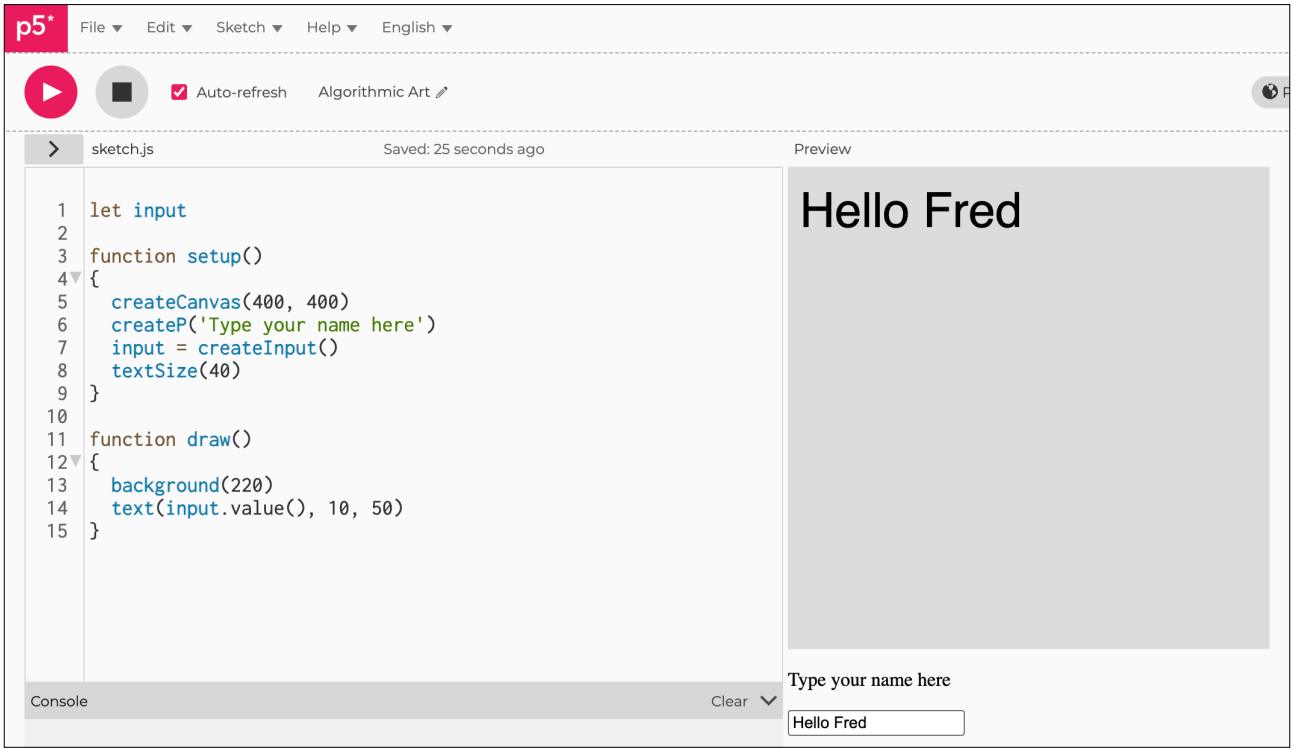
Notes

Creates a text box that you can enter text into. In this example, it prints it onto the canvas and into the window below the canvas. To handle the text, you create a variable called input and a variable called name. The input variable will hold the string from the inputted text. The name variable will adopt that and create a paragraph element using `createP()`.

Challenges

1. Make the text wobble as you type it in.
2. Add a button to make the text change colour.

Figure F2.9


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 25 seconds ago
Preview
let input
function setup()
{
createCanvas(400, 400)
createP('Type your name here')
input = createInput()
textSize(40)
}
function draw()
{
background(220)
text(input.value(), 10, 50)
}
Hello Fred
Console
Clear
Type your name here
Hello Fred

Sketch F2.10 hovering over text

Using HTML to change the text

sketch.js

```
let textP

function setup()
{
  noCanvas()
  textP = createP('Hover your mouse here')
  textP.mouseOver(overpara)
  textP.mouseOut(outpara)
}

function overpara()
{
  textP.html('Hello')
}

function outpara()
{
  textP.html('Good Bye')
}
```

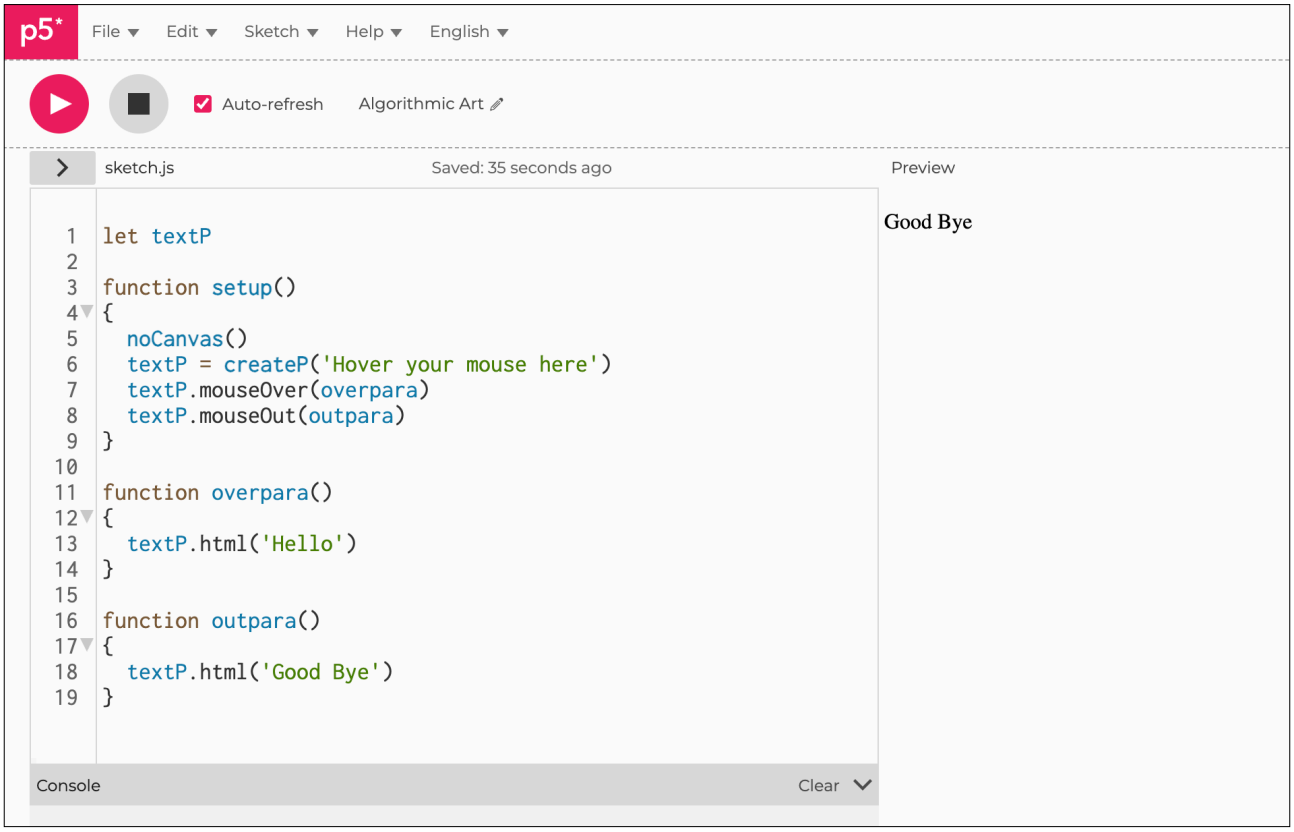
🗒️ Notes

You start by creating a variable called `textP` which will store the text. The text will change when your mouse hovers over the text using the `mouseOver()` function. The reverse applies when you move your mouse away from over the text using the function `mouseOut()`. To facilitate this, you create two functions called `overpara()` and `outpara()`. The `noCanvas()` function just removes the canvas! Also, you don't need the `draw()` function.

🌻 Challenges

1. How would you create more paragraphs that change when you hover over them?
2. What else could you change when you hover over the paragraphs?

Figure F2.10


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 35 seconds ago
Preview
Good Bye
let textP
function setup()
{
noCanvas()
textP = createP('Hover your mouse here')
textP.mouseOver(overpara)
textP.mouseOut(outpara)
}
function overpara()
{
textP.html('Hello')
}
function outpara()
{
textP.html('Good Bye')
}
Console
Clear

Sketch F2.11 hovering over the canvas

❗ Start a new sketch.
Three ways to change the colour of the background.

sketch.js

```
let backgroundColour

function setup()
{
  canvas = createCanvas(400, 400)
  canvas.mouseOver(changeColour)
  canvas.mouseOut(changeColour)
  canvas.mousePressed(changeColour)
  backgroundColour = color(220)
}

function changeColour()
{
  backgroundColour = color(random(255), random(255), random(255))
}

function draw()
{
  background(backgroundColour)
}
```

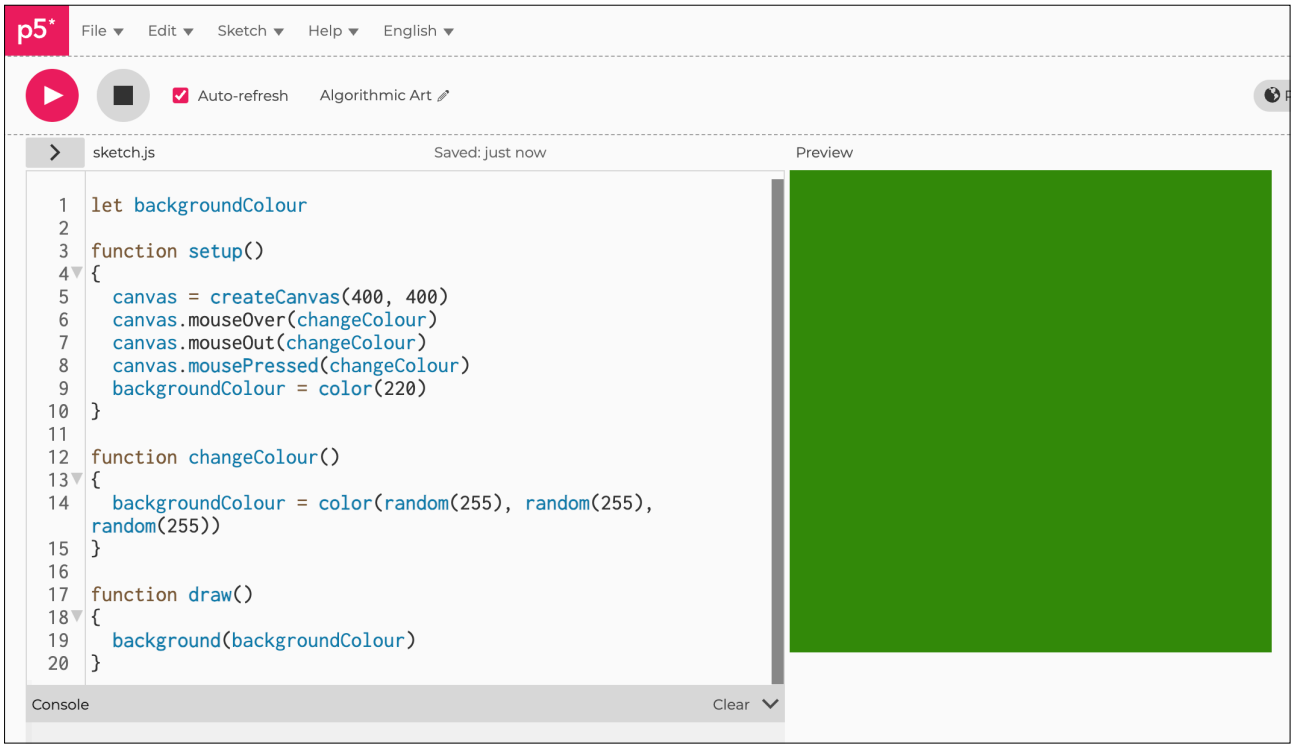
🗒️ Notes

Here are three ways to change the colour of the background. Using `mouseOver()`, `mouseOut()`, and `mousePressed()`.

🌻 Challenges

1. How would you change just the colour of the circle using `mouseOver()`?
2. How would you change the colour when the mouse is released? There is a clue in the name.

Figure F2.11


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
let backgroundColour
function setup()
{
canvas = createCanvas(400, 400)
canvas.mouseOver(changeColour)
canvas.mouseOut(changeColour)
canvas.mousePressed(changeColour)
backgroundColour = color(220)
}
function changeColour()
{
backgroundColour = color(random(255), random(255), random(255))
}
function draw()
{
background(backgroundColour)
}
Console
Clear

Introduction to CSS

CSS stands for Cascading Style Sheets. It is the style that you see on a website, the colours, spacing, format, and general appearance.

You can have headers <h1>_____</h1>
Paragraphs _____

A great place to look at all the possibilities is w3schools.com.
This is essential if you want to build and design your own website and is an integral part of HTML working alongside JavaScript. The p5.js web editor allows you to see the results in the canvas, so I recommend that you explore this side of coding. Check out the CSS file and try something.

I would spend longer on this unit topic, but there is so much to it, and there is so much else to get through. Maybe another time I might add something about website building, only so many hours in a day, though.

Sketch F2.12 introduction to the index.html file

! Delete all the code in `sketch.js` and move over to the `index.html` file.
Using the `index.html` file, you can add headers and paragraphs using HTML markup. The `<h1>` always has a corresponding `</h1>` to close. Keep this `index.html` file as is for the next few sketches.

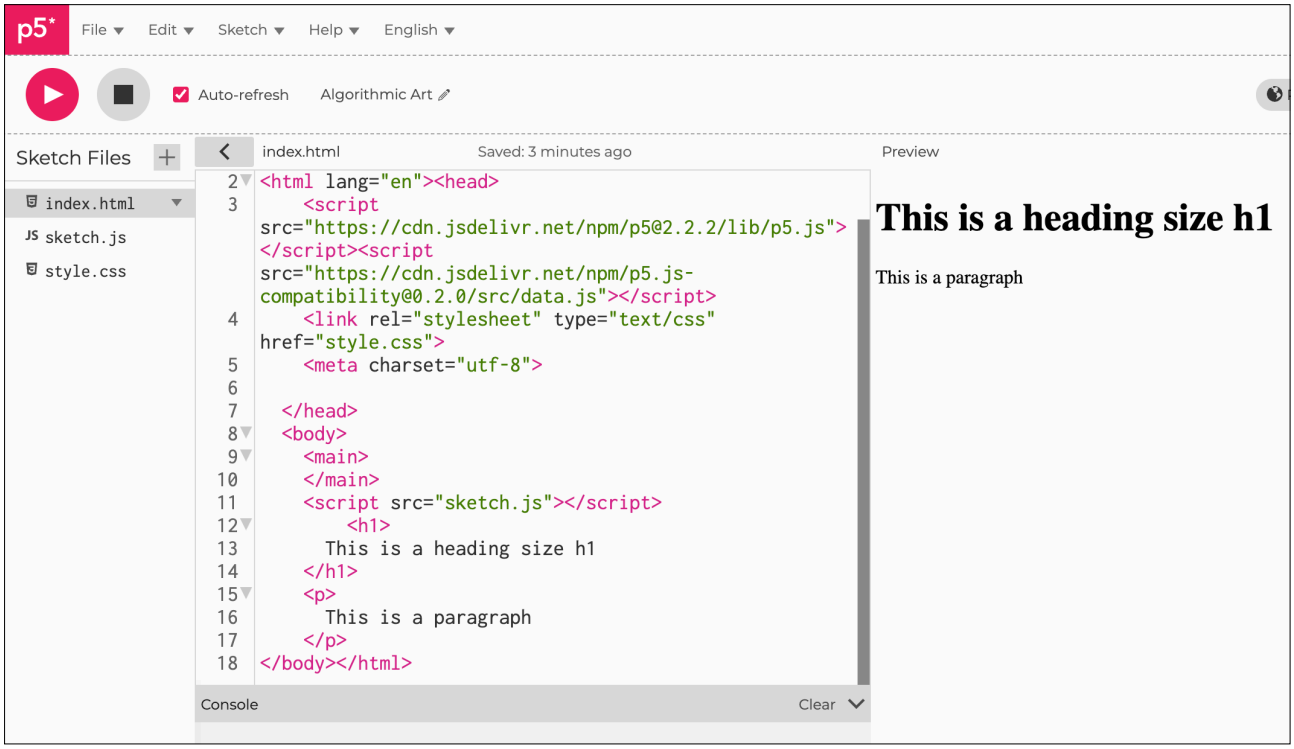
index.html

```
<!DOCTYPE html>
<html lang="en"><head>
    <script src="https://cdn.jsdelivr.net/npm/p5@2.2.2/lib/p5.js"></
script><script src="https://cdn.jsdelivr.net/npm/p5.js-compatibility@0.2.0/
src/data.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">


  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
        <h1>
      This is a heading size h1
    </h1>
    <p>
      This is a paragraph
    </p>
</body></html>
```

🗒️ Notes

This is the sort of coding you would see on a website.

Figure F2.12


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
sketch.js
style.css
index.html
Saved: 3 minutes ago
Preview
<html lang="en"><head>
<script
src="https://cdn.jsdelivr.net/npm/p5@2.2.2/lib/p5.js">
</script><script
src="https://cdn.jsdelivr.net/npm/p5.js-
compatibility@0.2.0/src/data.js"></script>
<link rel="stylesheet" type="text/css"
href="style.css">
<meta charset="utf-8">
</head>
<body>
<main>
</main>
<script src="sketch.js"></script>
<h1>
This is a heading size h1
</h1>

This is a paragraph

</body></html>
Console
Clear
This is a heading size h1
This is a paragraph

Sketch F2.13 adding a sketch

❗ Move over to sketch.js.
The corresponding sketch.

sketch.js

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

function draw()
{
  background(220)
  circle(width/2, height/2, 100)
}
```

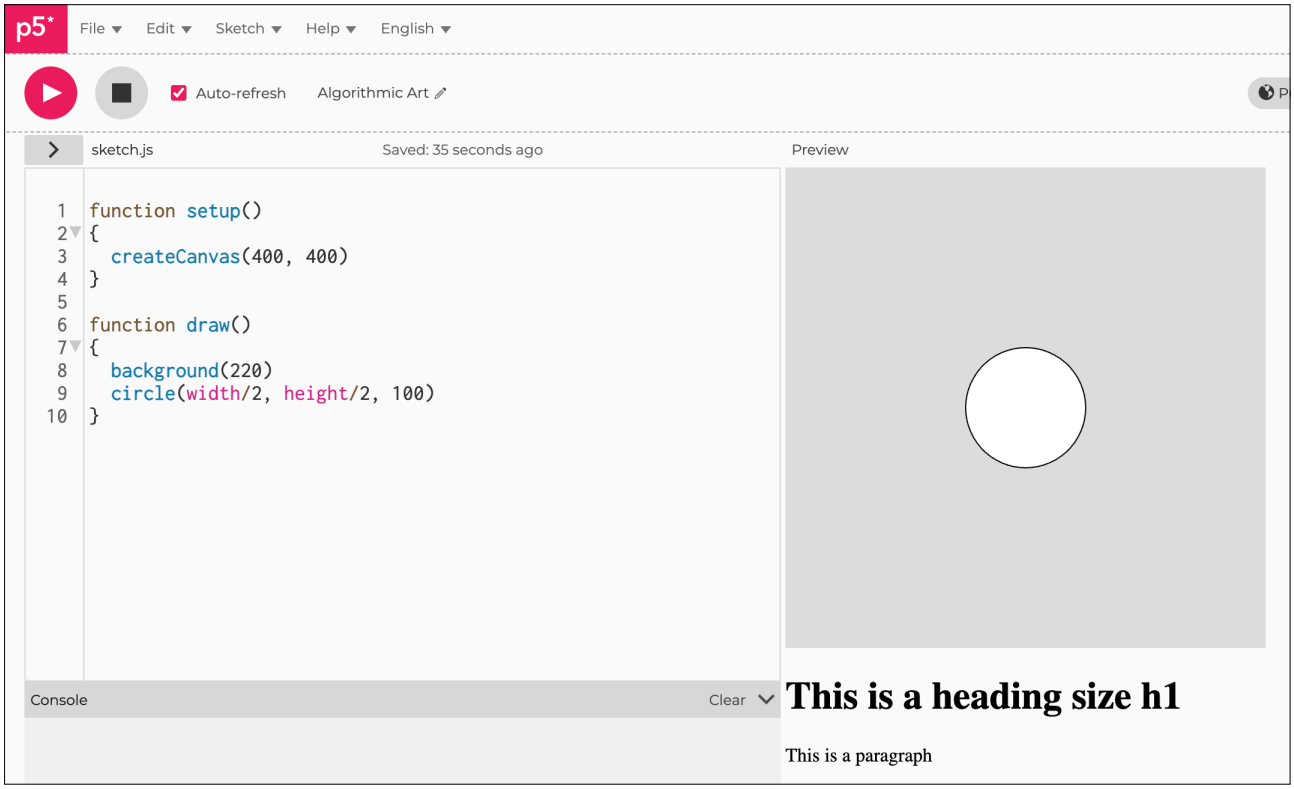
🗒️ Notes

A simple example of putting a header and a paragraph in the `index.html`. The notation `<h1> </h1>` are the tags for header size 1. The `<p> </p>` tags are for a paragraph. Also note that the sketch is drawn first.

🌻 Challenges

1. Try using a different header, e.g. `<h3> </h3>` as well.
2. Add another paragraph.

Figure F2.13


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 35 seconds ago
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
circle(width/2, height/2, 100)
}
Console
Clear
This is a heading size h1
This is a paragraph

Sketch F2.14 CSS changed

❗ Remove the `draw()` function and canvas (`noCanvas()`)
To see the effect of the CSS on the text. You can describe the background colour and the padding space around the text in pixels (px).

sketch.js

```
let text

function setup()
{
  noCanvas()
  text = createP('Changing the CSS around the text')
  text.mouseOver(changeStyle)
  text.mouseOut(revertStyle)
}

function changeStyle()
{
  text.style('background-color', 'yellow')
  text.style('padding', '32px')
}

function revertStyle()
{
  text.style('background-color', 'orange')
  text.style('padding', '8px')
}
```

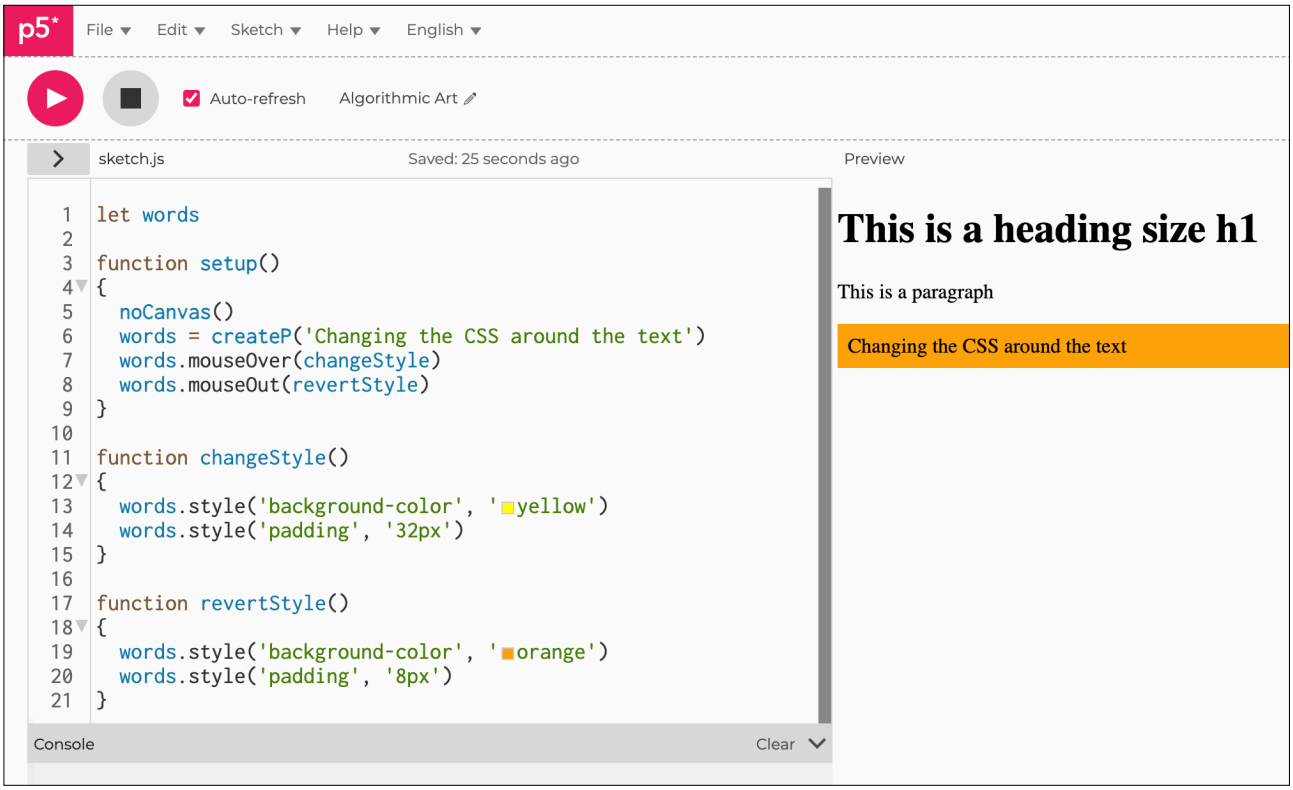
🗒️ Notes

This demonstrates how you can interact with text and change the style as you hover over it. There are two elements changing the padding, which is the amount of space around the text and the background colour. The suffix px means pixels.

🌻 Challenges

1. Try different colours
2. Try different amounts of padding
3. Add more text

Figure F2.14


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 25 seconds ago
Preview
let words
function setup()
{
noCanvas()
words = createP('Changing the CSS around the text')
words.mouseOver(changeStyle)
words.mouseOut(revertStyle)
}
function changeStyle()
{
words.style('background-color', 'yellow')
words.style('padding', '32px')
}
function revertStyle()
{
words.style('background-color', 'orange')
words.style('padding', '8px')
}
This is a heading size h1
This is a paragraph
Changing the CSS around the text
Console
Clear

Sketch F2.15 submit button

In this next bit, we are going to create a submit button that will take data (your name) and push it onto the canvas.

❗ Start a new sketch, and also remove the HTML tags in the index.html file.

First, we create an input box to input our name.

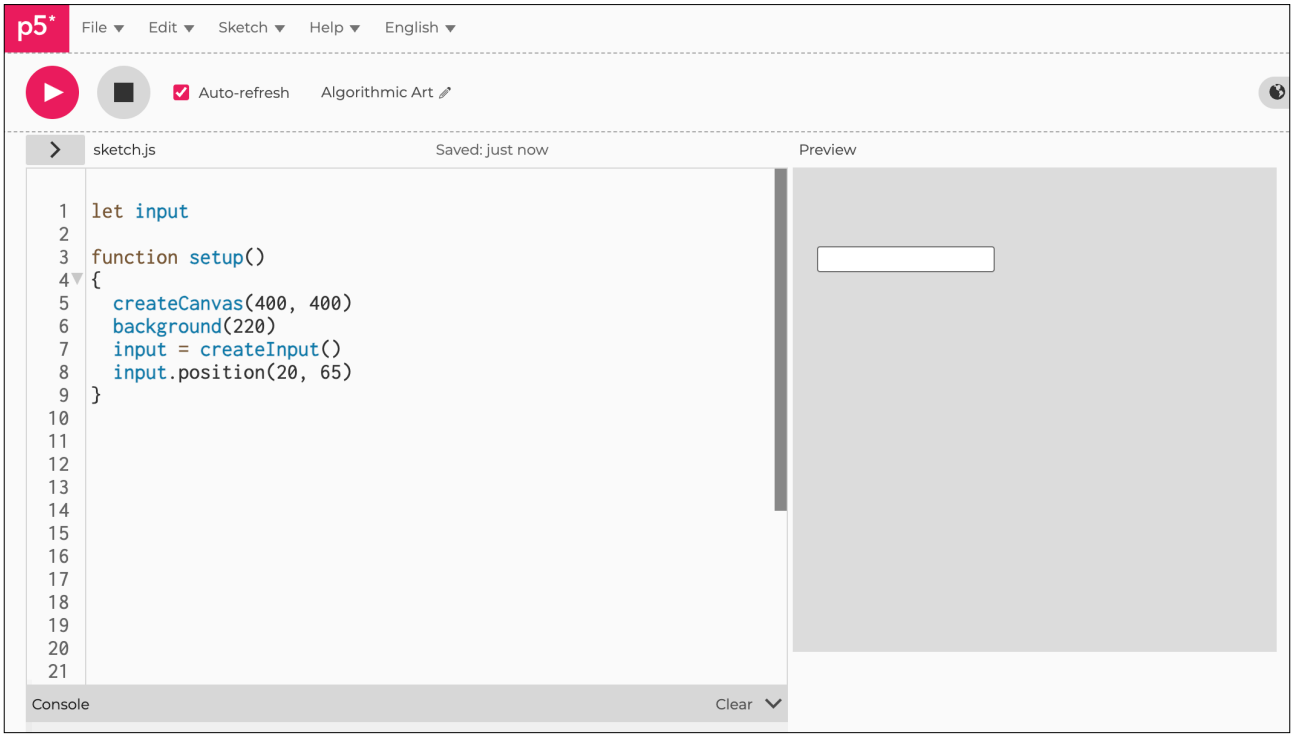
sketch.js

```
let input


function setup()
{
  createCanvas(400, 400)
  background(220)
  input = createInput()
  input.position(20, 65)
}
```

🗒️ Notes

This does nothing yet, even though you can type into it.

Figure F2.15


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
let input
function setup()
{
createCanvas(400, 400)
background(220)
input = createInput()
input.position(20, 65)
}
Console
Clear

Sketch F2.16 adding the button

We put the submit button next to the input box.

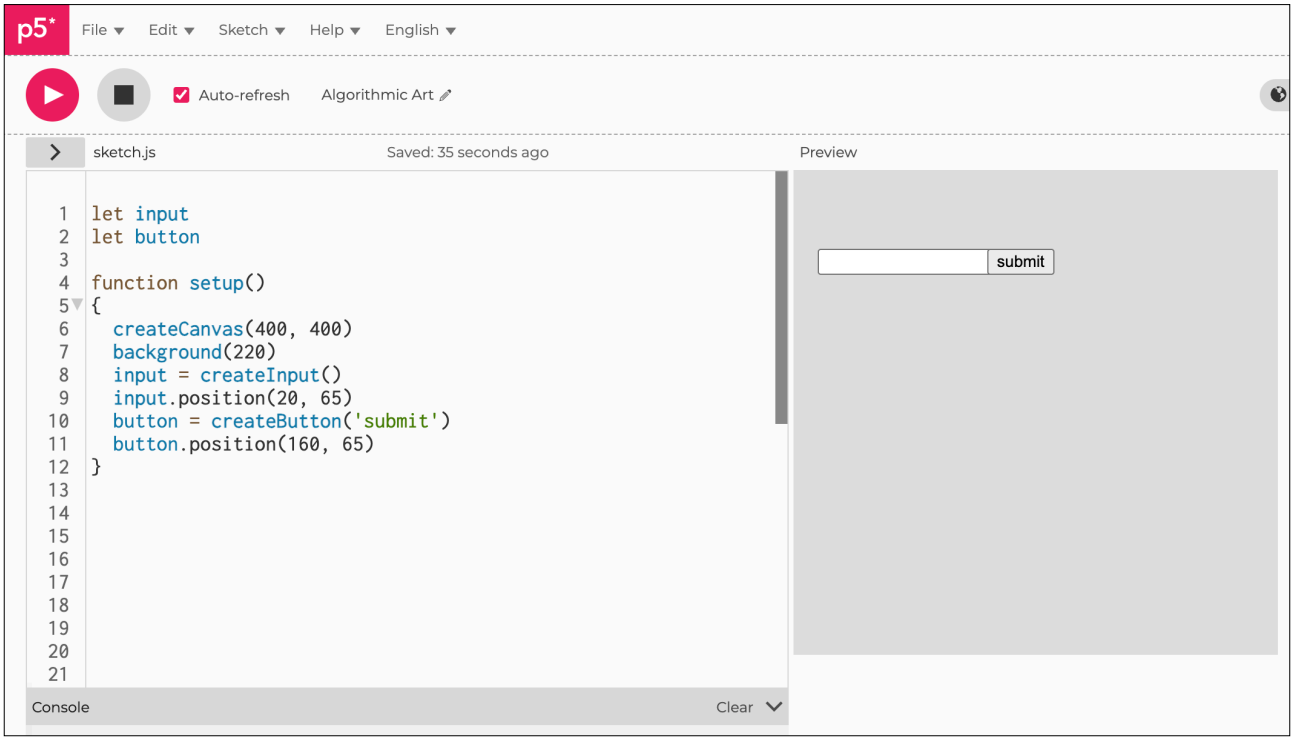
sketch.js

```
let input
let button

function setup()
{
  createCanvas(400, 400)
  background(220)
  input = createInput()
  input.position(20, 65)
  button = createButton('submit')
  button.position(160, 65)
}
```

Notes

This still does nothing.

Figure F2.16


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 35 seconds ago
Preview
let input
let button
function setup()
{
createCanvas(400, 400)
background(220)
input = createInput()
input.position(20, 65)
button = createButton('submit')
button.position(160, 65)
}
submit
Console
Clear

Sketch F2.17 input function

Next, we create a function to take the input data (your name or whatever). This is a function we have called `inputName()`. The variable `name` takes the inputted name and puts it as text on the canvas.

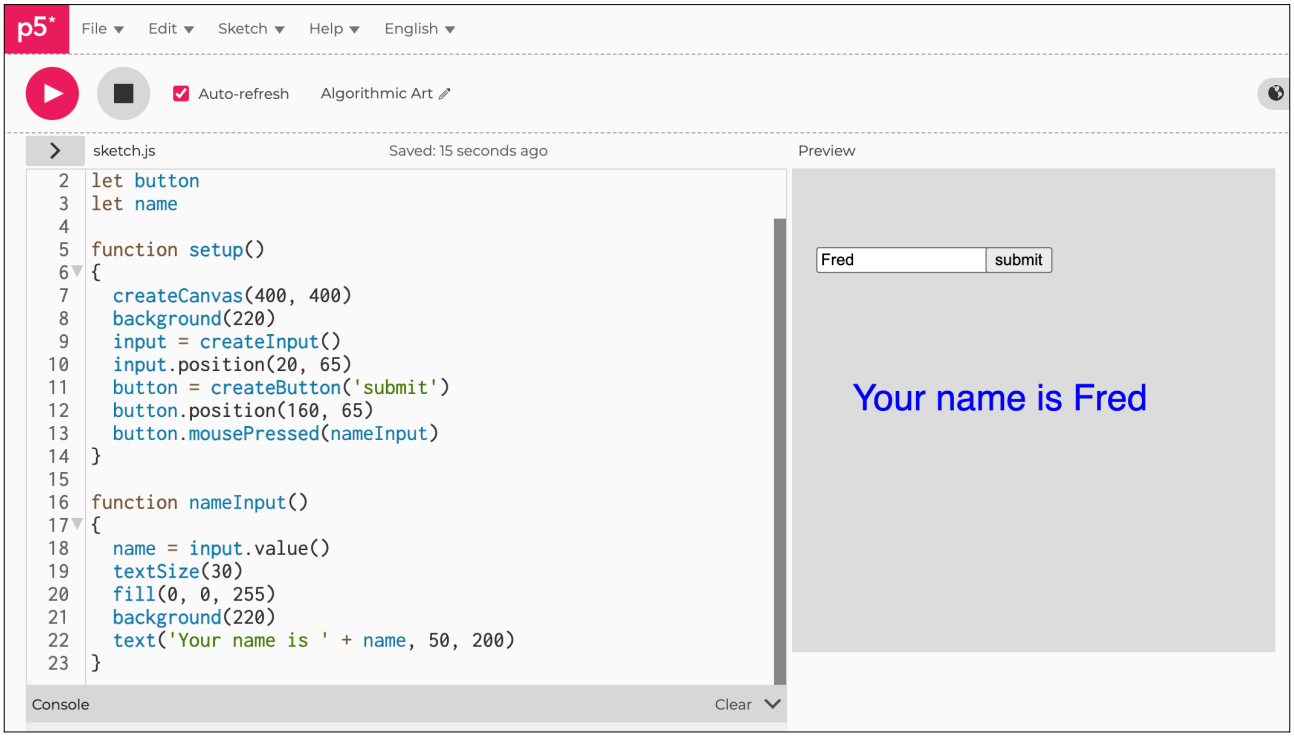
sketch.js

```
let input
let button
let name

function setup()
{
  createCanvas(400, 400)
  background(220)
  input = createInput()
  input.position(20, 65)
  button = createButton('submit')
  button.position(160, 65)
  button.mousePressed(nameInput)
}

function nameInput()
{
  name = input.value()
  textSize(30)
  fill(0, 0, 255)
  background(220)
  text('Your name is ' + name, 50, 200)
}
```

🗒️ Notes

We draw the `background()` again to clear the canvas so you don't have to reload it each time.

Figure F2.17


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
let button
let name
function setup()
{
createCanvas(400, 400)
background(220)
input = createInput()
input.position(20, 65)
button = createButton('submit')
button.position(160, 65)
button.mousePressed(nameInput)
}
function nameInput()
{
name = input.value()
textSize(30)
fill(0, 0, 255)
background(220)
text('Your name is ' + name, 50, 200)
}
Fred
submit
Your name is Fred
Console
Clear

Sketch F2.18 have the question

Finally, we can put a question on the canvas as a heading.

sketch.js

```
let input
let button
let name
let question

function setup()
{
  createCanvas(400, 400)
  background(220)
  input = createInput()
  input.position(20, 65)
  button = createButton('submit')
  button.position(160, 65)
  button.mousePressed(nameInput)
  question = createElement('h2', 'What is your name?')
  question.position(20, 5)
}

function nameInput()
{
  name = input.value()
  textSize(30)
  fill(0, 0, 255)
  background(220)
  text('Your name is ' + name, 50, 200)
}
```

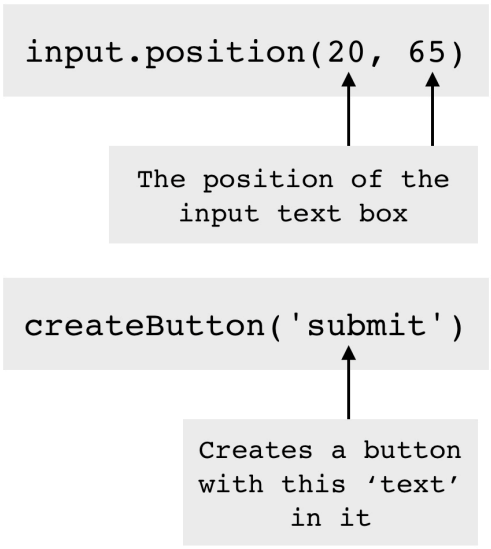
🗒️ Notes

The benefit of using the DOM element as the question is that it doesn't get removed when the background is refreshed.

There is a lot happening here, but to summarise: you are creating a text box and a submit button. A text box is created when you use the `createInput()` function. You create a variable called `input` and when you type in your name, it stores it as a

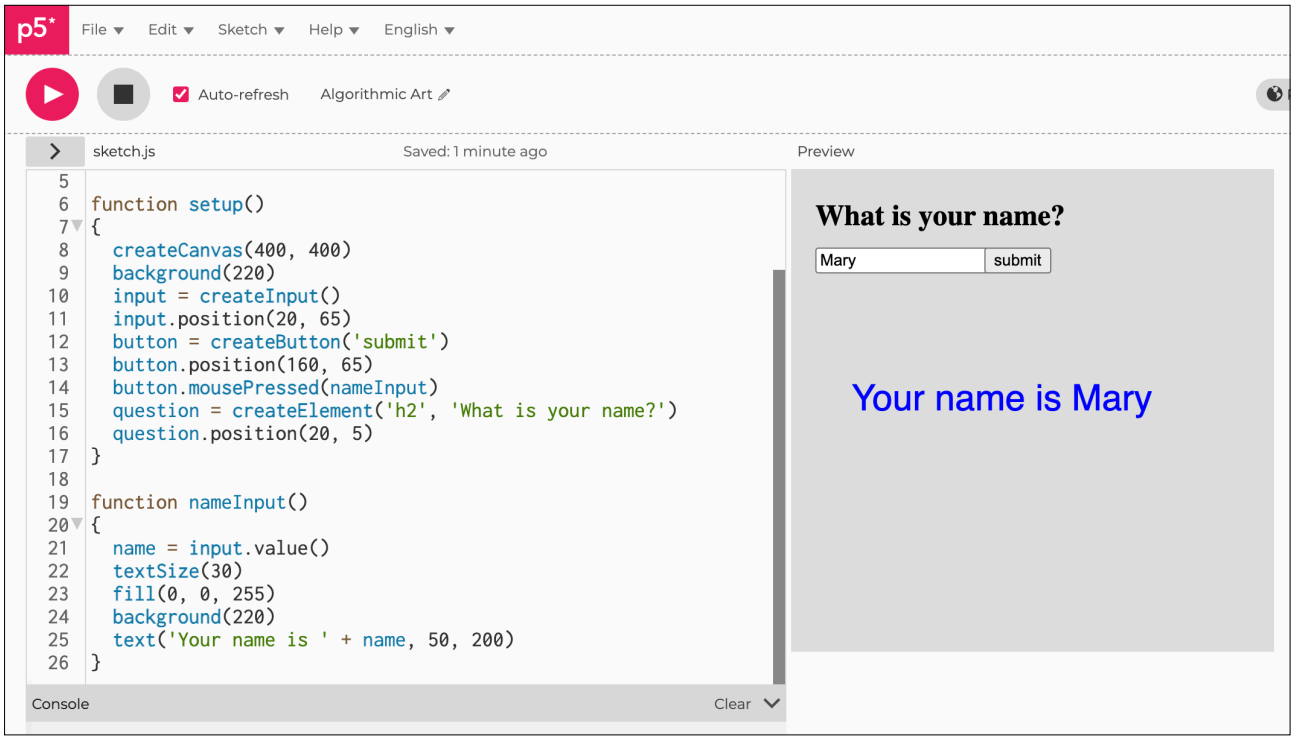
string. Because you are using an HTML element, you need to give it a position or else it appears at the bottom of the canvas. You create a button with the function `createButton()` and you put any text which you want to appear on the button inside the brackets with speech marks.

When the button is pressed, it calls a function we have created called function `nameInput()`. This is the function where we do something with the name we have stored. We pull that information and put it in a variable called name and use it to write text on the canvas.



The `createElement('h2', 'Type you name')` in this case `h2` is heading 2 which is the second biggest and down to `h6`, the smallest heading.

Figure F2.18


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 1 minute ago
Preview
function setup()
{
createCanvas(400, 400)
background(220)
input = createInput()
input.position(20, 65)
button = createButton('submit')
button.position(160, 65)
button.mousePressed(nameInput)
question = createElement('h2', 'What is your name?')
question.position(20, 5)
}
function nameInput()
{
name = input.value()
textSize(30)
fill(0, 0, 255)
background(220)
text('Your name is ' + name, 50, 200)
}
Console
Clear
What is your name?
Mary
submit
Your name is Mary

Sketch F2.19 simple calculator

❗ Totally new sketch.
Input two numbers that are then multiplied.

sketch.js

```
let input1
let input2
let button
let total
let title
let num1
let num2

function setup()
{
  createCanvas(400, 400)
  background(220)
  input1 = createInput('')
  input1.position(20, 65)
  button = createButton('submit')
  button.position(60, 130)
  button.mousePressed(number)
  input2 = createInput('')
  input2.position(20, 100)

  title = createElement('h2', 'Multiplying two numbers')
  title.position(20, 5)
  textSize(20)
}

function number()
{
  background(220)
  num1 = input1.value()
  num2 = input2.value()
  total = num1 * num2
  text('The answer is ' + total, 100, 200)
```

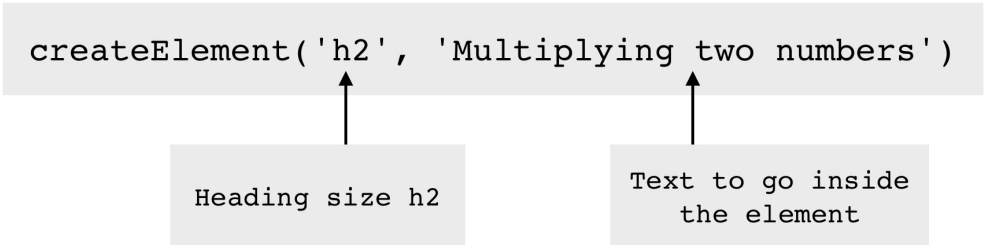
```
}
```

🗒️ Notes

The beauty of this application is that you can start to collect data, information, and manipulate it. Here we are making a very simple calculator. The two numbers are multiplied together. However, if you were to change the `*` (multiply) to a `+` (addition), you will notice that it treats them as strings and writes the one after the other. This is because we should've made sure that the variables are read as integers, not strings (by default). The solution is to change these two lines of code...

```
num1 = int(input1.value())
num2 = int(input2.value())
```

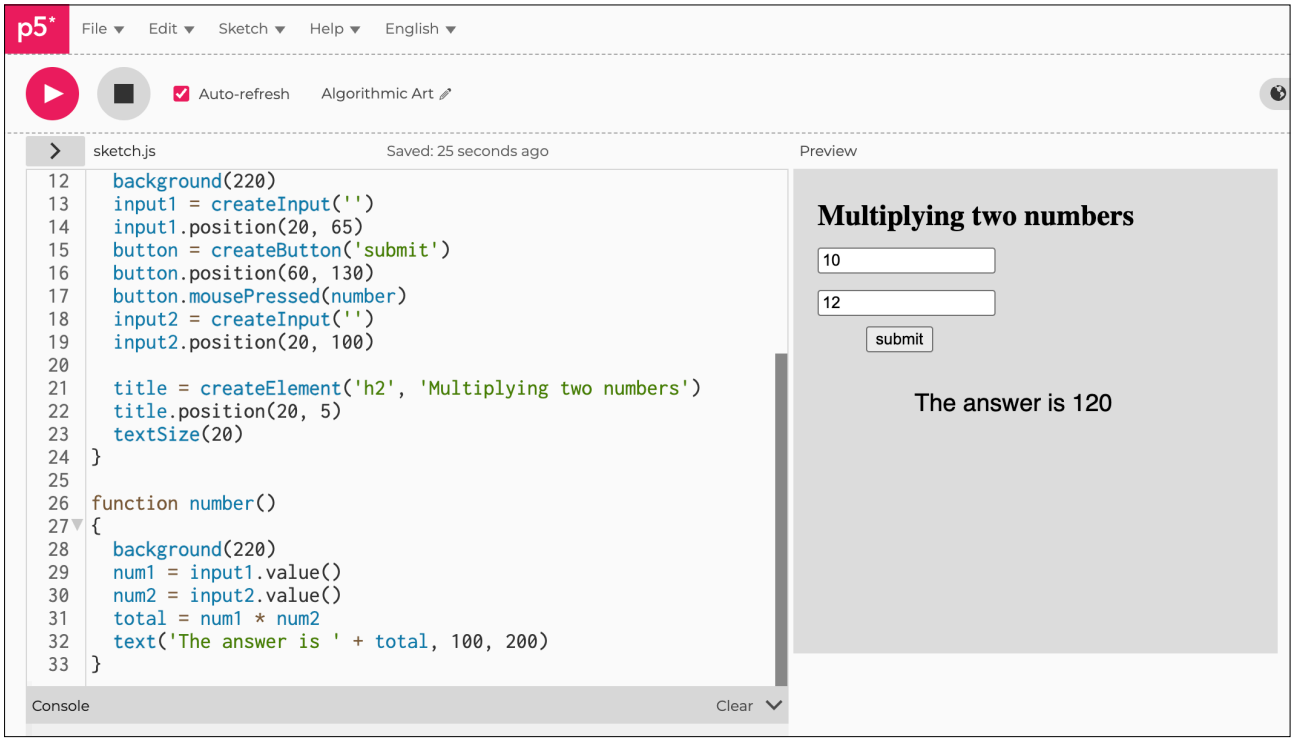
The `createElement()` has a heading `h2`. If there is a lot of text, then replace the `h2` with `p` for paragraph.



🌻 Challenge

Add more functionality with addition, subtraction, or division.

Figure F2.19


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 25 seconds ago
Preview
background(220)
input1 = createInput('')
input1.position(20, 65)
button = createButton('submit')
button.position(60, 130)
button.mousePressed(number)
input2 = createInput('')
input2.position(20, 100)
title = createElement('h2', 'Multiplying two numbers')
title.position(20, 5)
textSize(20)
}
function number()
{
background(220)
num1 = input1.value()
num2 = input2.value()
total = num1 * num2
text('The answer is ' + total, 100, 200)
}
Console
Clear
Multiplying two numbers
10
12
submit
The answer is 120

Sketch F2.20 slider colour circle

❗ Another new sketch.
Three RGB sliders so that you can create your own colour chart.

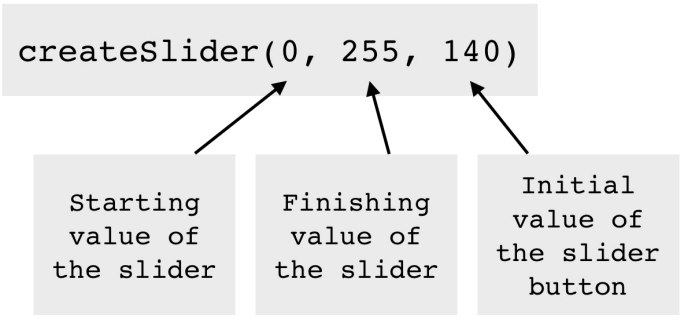
sketch.js

```
let sliderRed
let sliderGreen
let sliderBlue

function setup()
{
  createCanvas(400, 400)
  sliderRed = createSlider(0, 255, 140)
  sliderGreen = createSlider(0, 255, 140)
  sliderBlue = createSlider(0, 255, 140)
}

function draw()
{
  background(220)
  fill(sliderRed.value(), sliderGreen.value(), sliderBlue.value())
  textSize(50)
  circle(width/2, height/4, 100)
  textSize(30)
  fill(0)
  text('red: '+ sliderRed.value(), 200, 250)
  text('green: '+ sliderGreen.value(), 200, 300)
  text('blue: '+ sliderBlue.value(), 200, 350)
  sliderRed.position(50, 250)
  sliderGreen.position(50, 300)
  sliderBlue.position(50, 350)
}
```

🗒️ Notes

Here we can create a slider to change the colour of the circle. The line `createSlider(0, 255, 140)` gives us three arguments. The first one `(0)` is the minimum value, the second `(255)` is the maximum value, and the third `(140)` is the starting position of the slider button between the max and the min. The

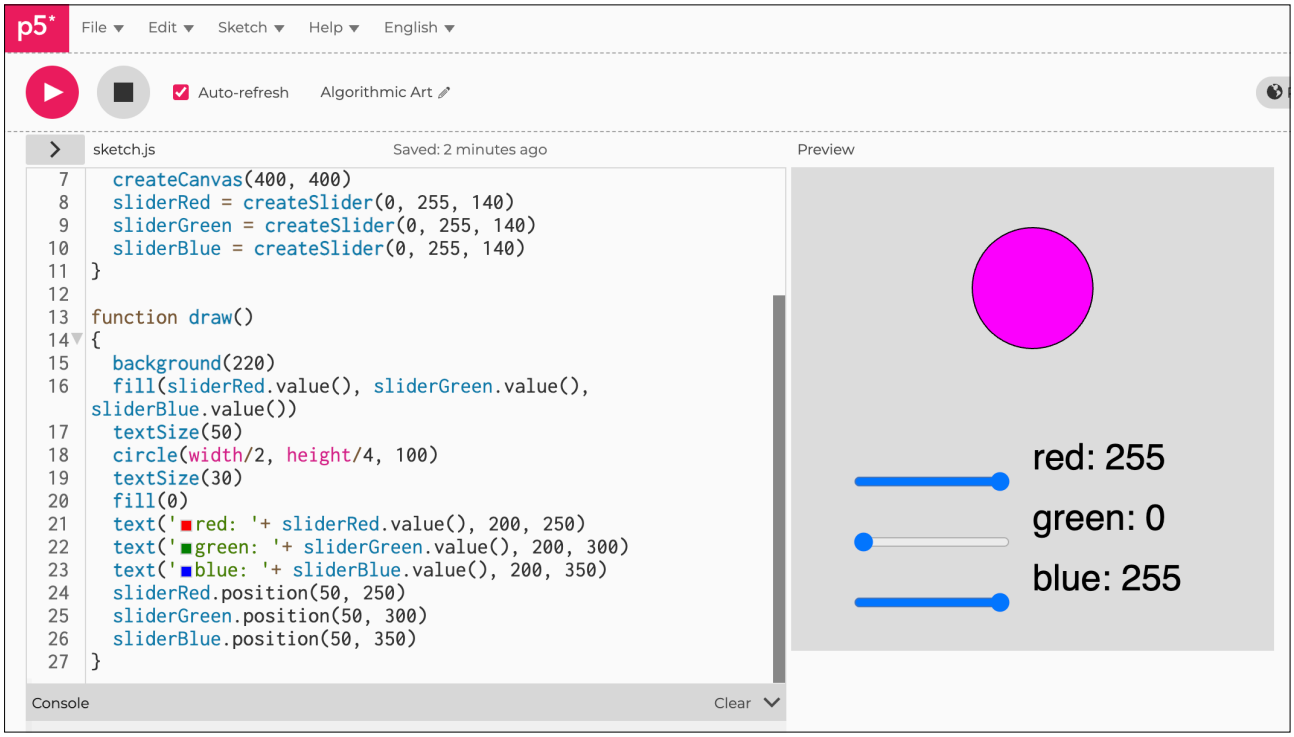
`sliderRed.value()` gets the value of the slider element and is then used for the amount of red colour in this instance; the value is also printed on the canvas.



🌻 Challenge

Add an extra slider for diameter.

Figure F2.20



Sketch F2.21 slider rotate

❗ Yet another new sketch.
Rotating a baton with the slider, a nice demonstration.

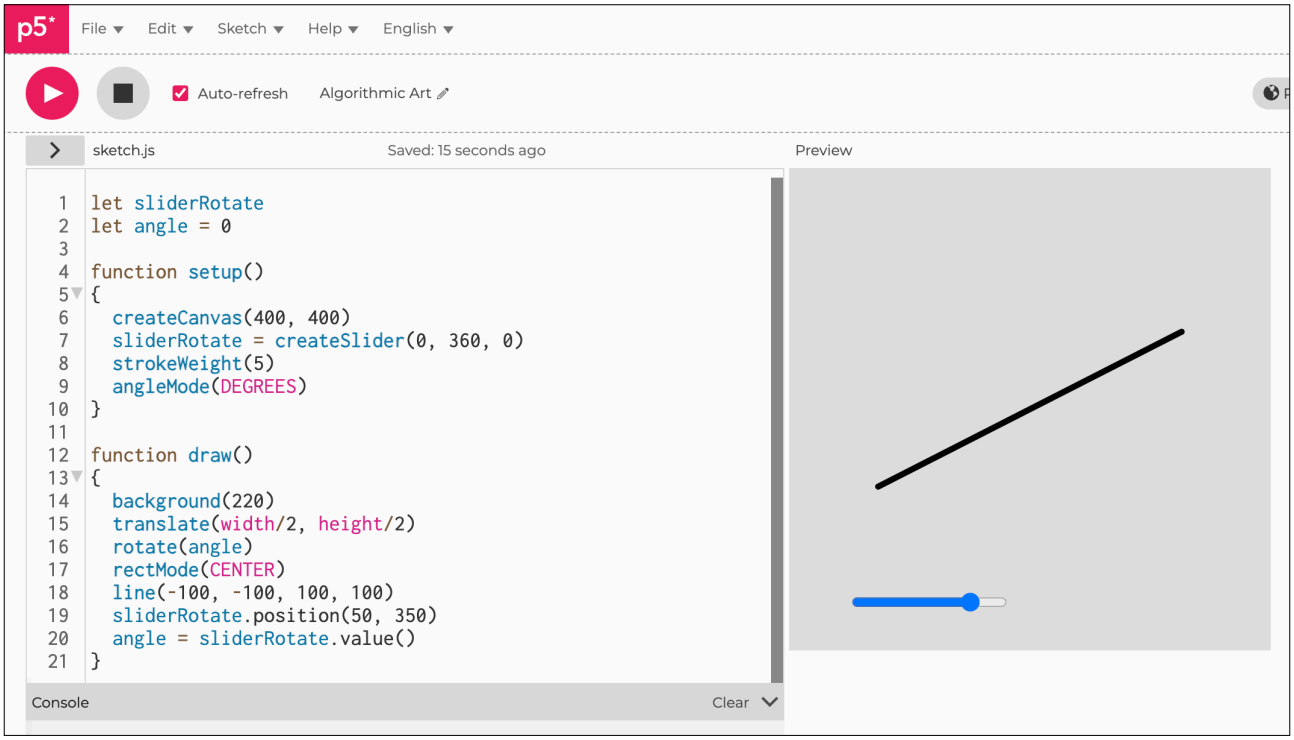
sketch.js

```
let sliderRotate
let angle = 0

function setup()
{
  createCanvas(400, 400)
  sliderRotate = createSlider(0, 360, 0)
  strokeWeight(5)
  angleMode(DEGREES)
}

function draw()
{
  background(220)
  translate(width/2, height/2)
  rotate(angle)
  rectMode(CENTER)
  line(-100, -100, 100, 100)
  sliderRotate.position(50, 350)
  angle = sliderRotate.value()
}
```

🗒️ Notes
Another use of the slider value, in this instance to spin a line.

🌻 Challenge
Can you think of any other applications or fun things you could control?

Figure F2.21


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
let sliderRotate
let angle = 0
function setup()
{
createCanvas(400, 400)
sliderRotate = createSlider(0, 360, 0)
strokeWeight(5)
angleMode(DEGREES)
}
function draw()
{
background(220)
translate(width/2, height/2)
rotate(angle)
rectMode(CENTER)
line(-100, -100, 100, 100)
sliderRotate.position(50, 350)
angle = sliderRotate.value()
}
Console
Clear

Sketch F2.22 radio buttons colour

Using a predefined button to change the colour of the circle.

sketch.js

```
let radio
let val = 255

function setup()
{
  createCanvas(400, 400)
  radio = createRadio()
  radio.position(10, 10)
  radio.option('1', 'red')
  radio.option('2', 'green')
  radio.option('3', 'blue')
}

function draw()
{
  background(220)
  circle(width/2, height/2, 100)
  val = radio.value()
  if (val == 1)
  {
    fill(200, 0, 0)
  }
  if (val == 2)
  {
    fill(0, 200, 0)
  }
  if (val == 3)
  {
    fill(0, 0, 200)
  }
}
```

🗒️ Notes

Three buttons to change the colour of the circle. Very simple but also very powerful if you want to use the data. Adds to the interactivity of your canvas, page, or website.

🌻 Challenges

1. Remove `radio.position(10, 10)`
2. Create shapes instead of colours
3. Have an event happen, e.g. `10` random bubbles
4. Use the to create a paint package

Figure F2.22