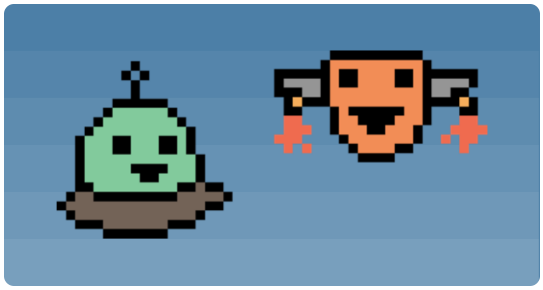
Projects

Pixel art

Create a pixel art editor using HTML, CSS, and JavaScript.

HTML / CSS

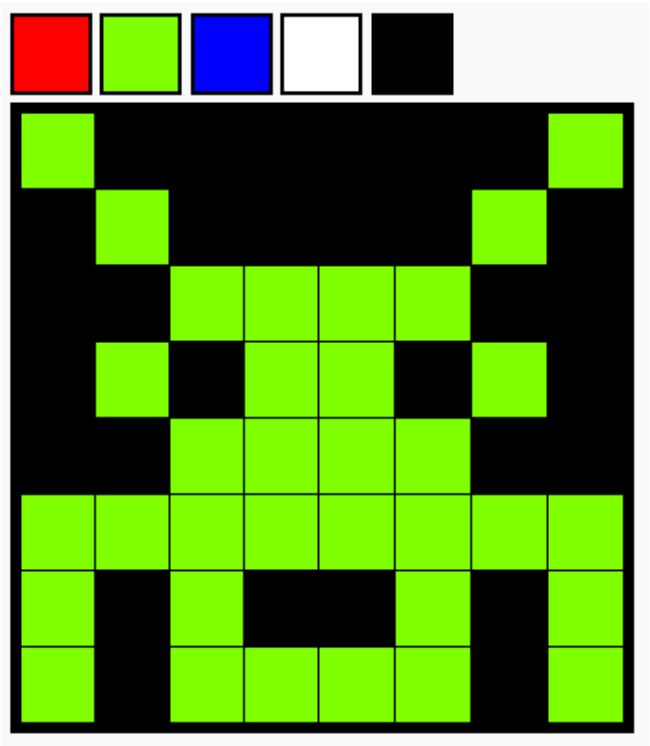


Step 1 Introduction

Create a pixel art editor. As well as using HTML and CSS, you'll learn how to use JavaScript to add interactivity to your project.

What you will make

Try out the project below. Click on a colour from the palette, then click on a pixel to change its colour.



What you will learn

This project covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (http://rpf.io/curriculum):

- **Design basic 2D and 3D assets** (https://www.raspberrypi.org/curriculum/design/creator).
- **Use basic programming constructs to create simple programs** (https://www.raspberrypi.org/curriculum/programming/creator)

Additional information for educators

If you need to print this project, please use the **printer-friendly version** (https://projects.raspberrypi.org/en/projects/pixel-art/print) .

Use the link in the footer to access the GitHub repository for this project, which contains all resources (including an example finished project) in the 'en/resources' folder.

Step 2 What you will need

Hardware

- A computer capable of accessing the **trinket.io** (https://trinket.io) website

Software

This project can be completed in a web browser using the **trinket.io** (https://trinket.io) online application.

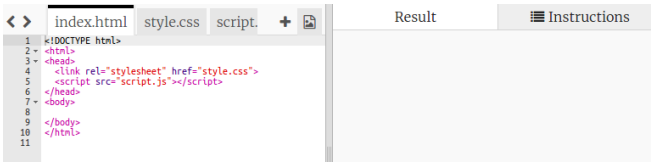
Step 3 Create a grid of pixels

Let's create a grid of pixels that you can use for creating pixel art.

The grid will look like a table. Tables contain rows, and rows contain cells which will represent the pixels.

- Open the **starter trinket** (http://jumpto.cc/web-pixel).

The project should look like this:



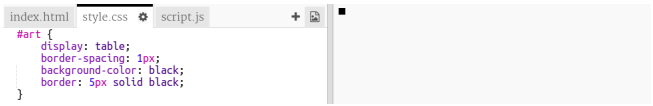
First, let's write some code to create a table with a black background and then put white pixels into it.

- Add this code into the `<body>` of your `index.html` file to create a `<div>`:

```
</head>
<body>
<div id="art"></div>
</body>
</html>
```

A `<div>` is an invisible box to which you can give a **style**. This `<div>` has the ID `art`, which you need so you can add styles to the box.

- Now go to your `style.css` file and add the table styling for the `<div>` called `art`.



This creates a table with a border and sets the spacing inside the grid.

It doesn't look very interesting yet, so you need to put rows of pixels inside it.

- Go back to your `index.html` file and add a row of three pixels **inside** the `art` box. If you want to save time, you can type the first row and then copy and paste it to create the others.

```
<div id="art">
  <div class="row">
    <div class="pixel"></div>
    <div class="pixel"></div>
    <div class="pixel"></div>
  </div>
</div>
```

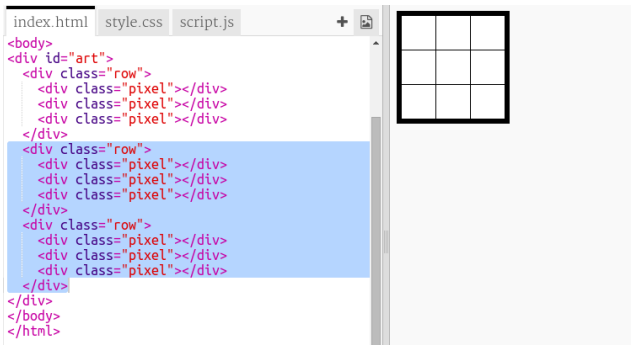
Notice that here you're using a **class** instead of an ID to style the divs. This is because there will be lots of them, so a class is more useful.

- Switch to the `style.css` file and add the following styles for the rows and the pixels within each row:

Now your pixels will line up in a grid with black lines around them.

- In your `index.html` file, add another two sections of pixels to create a 3×3 pixel grid. You can use copy and paste again to save time.

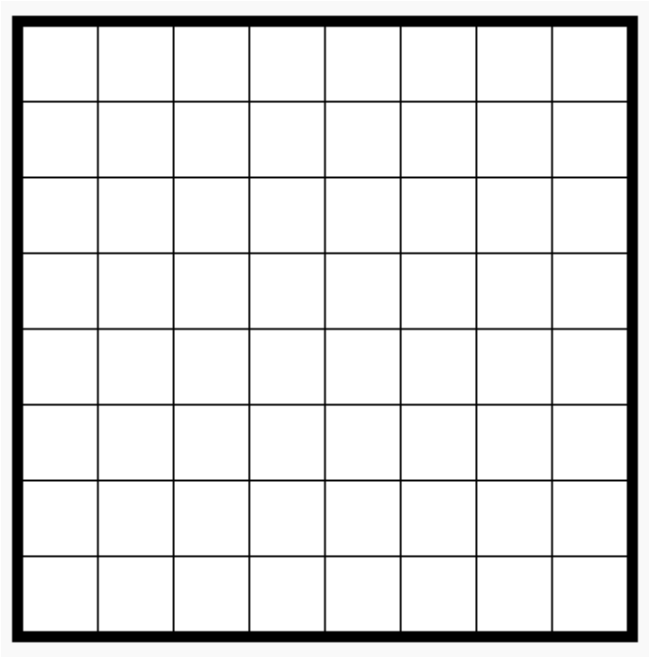
Here is how your code should look:



Step 4 Challenge: resize your grid

3×3 is quite a small grid for pixel art. Can you make the grid bigger? 8×8 is a good size for pixel art.

Try copying and pasting rather than typing everything out.



Step 5 Colour the pixels

This project uses three different languages:

- HTML is used to organise your content
- CSS tells the content what to look like with styles
- JavaScript is a programming language you can use to make a webpage respond when you interact with it

Let's add some JavaScript code to colour in a pixel automatically when you click on it.

We will create a **function**. Functions are named blocks of code which perform a particular task. We can **call** a function by its name when we want to run the code it contains.

- Inside the `script.js` file, create a function with the name `setPixelColour`. The `setPixelColour` function needs to take a `pixel` as an **input** so that it can change that pixel's colour.

```
index.html  style.css  script.js
1 function setPixelColour(pixel)
2 {
3
4 }
```

- Add this code inside the function to set the background colour of the pixel:

```
index.html  style.css  script.js
function setPixelColour(pixel)
{
  pixel.style.backgroundColor = 'black';
}
```

Notice that `backgroundColor` uses the American spelling of 'colour'.

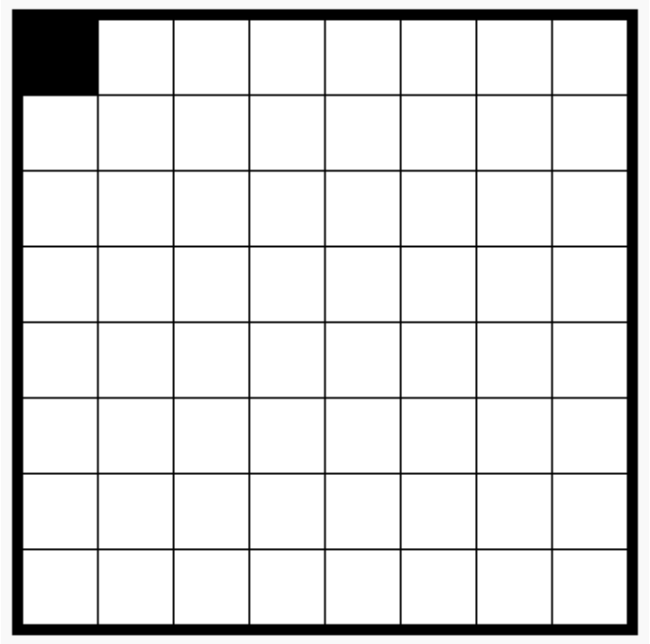
At the moment this code doesn't have any effect.

- Go to `index.html` and add the following code to the first pixel so that when you click on this pixel, the `setPixelColour` function is called:

```
<div id="art">
  <div class="row">
    <div class="pixel" onclick="setPixelColour(this)"></div>
    <div class="pixel"></div>
    <div class="pixel"></div>
```

The `this` in the brackets is the input for the `setPixelColour` function, which lets it know which pixel to set the colour for — `this` pixel!

- Test your code by clicking on the first pixel. It should turn black.

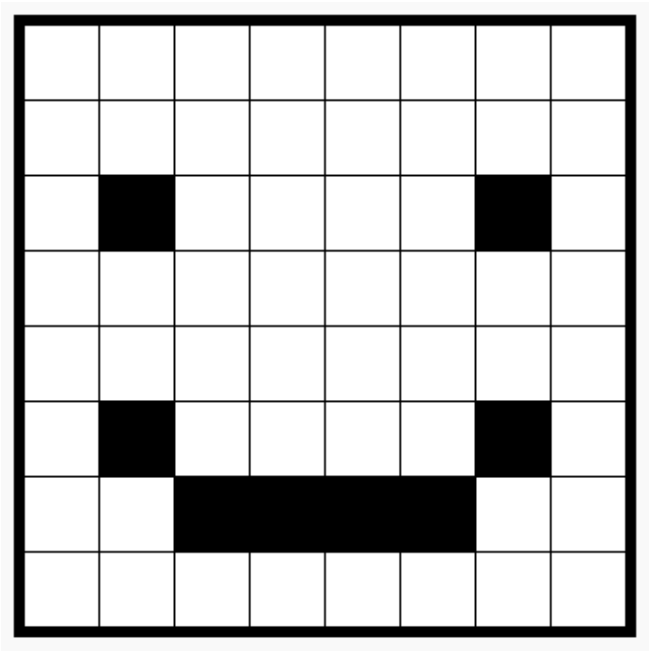


You've only added `onclick` code to the **first** pixel, so clicking on the other pixels won't do anything yet.

Step 6 Challenge: make all pixels clickable

Can you make all the pixels clickable? To save time, you can copy and paste the code you need.

Test your code by creating a quick piece of pixel art.



Tip: you can click **Autorun** to clear all of the pixels.

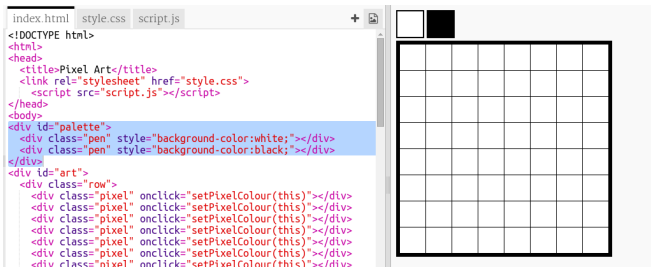
Step 7 Add a colour palette

Did you find it annoying that you couldn't change a pixel's colour back to white if you made a mistake? Let's fix that by creating a colour palette so that you can choose between pen colours with a click.

- Add this code at the bottom of your `style.css` file to create a pen style:

```
.pen {
  display: inline-block;
  width: 40px;
  height: 40px;
  border: 2px solid black;
}
```

- Now create a palette with black and white pen colours using the pen style you just created. Add the following code to your `index.html` below the `<body>` tag:



`style=` allows you to add CSS code inside your HTML file, which is convenient here.

We need to add code so that when one of the colours in the palette is clicked on, the colour of the pen changes.

- Switch to `script.js` and create a variable called `penColour` at the very top of the file. Set the value of the variable to `'black'`.

Create a variable in JavaScript

A variable allows you to store data within a program. Variables have a name and a value.

This variable has the name `animal` and the value `"cat"`:

```
var animal = "cat";
```

This variable has the name `score` and the value `30`:

```
var score = 30;
```

To create a variable, give it a name and set it equal to a value. The name of the variable always goes after `var`, and it must not contain any spaces.

- Below the variable, create a new function called `setPenColour` with an input of `pen`. Look at the function `setPixelColour` that you already created to help you.

Create a function in JavaScript

You can create a function using the following code. Replace `name_of_function` with the name of your function - make sure you only include letters, numbers and underscores in the name you choose.

```
function name_of_function(){

}
```

The code you want to be part of the function should go between the braces `{` and `}`.

You can call the function by typing its name:

```
name_of_function();
```

- Inside the `setPenColour` function, add code to set the `penColour` variable to the `pen` colour provided as the input.

```
var penColour = 'black';

function setPenColour(pen)
{
  penColour = pen;
}
```

You'll also need use the `penColour` variable when you change the colour of a pixel.

- Change the `setPixelColour` function to use the `penColour` variable instead of `black`:

```
function setPixelColour(pixel)
{
  pixel.style.backgroundColor = penColour;
}
```

- In the `index.html` file, add some code to call the `setPenColour` function when a colour in the palette is clicked.

```
<body>
<div id="palette">
  <div class="pen" style="background-color:white;" onclick="setPenColour('white')"></div>
  <div class="pen" style="background-color:black;" onclick="setPenColour('black')"></div>
</div>
<div id="art">
```

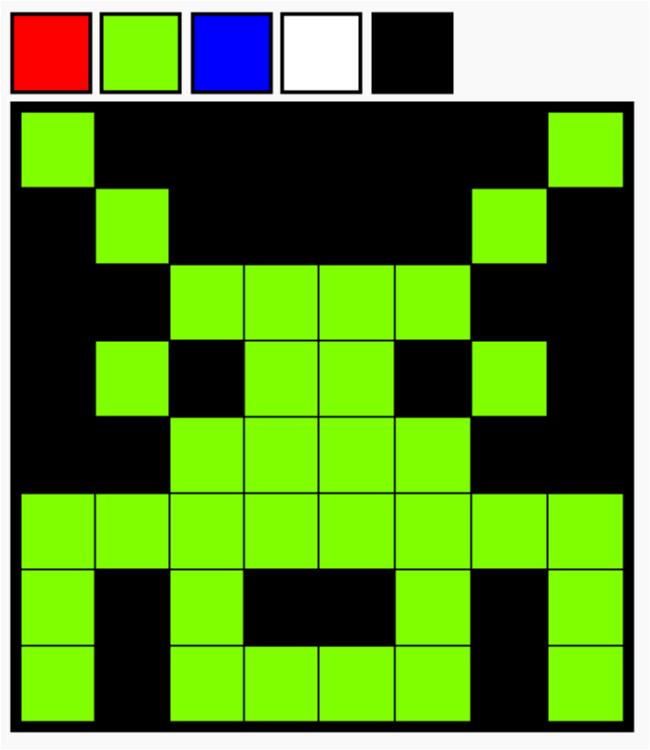
- Test that you can switch the pen colour between black and white to fill in or delete pixels.

Step 8 Challenge: add more colours to the palette

Can you add more colours to the palette?

- Choose the colours you want to use in your pixel art, and add them to your code. Then create some cool pixel images.

Hint: The bright green colour is called `chartreuse`. Here is a **list of colour names** (https://www.w3schools.com/colors/colors_names.asp) from which you can pick your favourites.



You can use the Snipping Tool in Windows (or an alternative if you're not using Windows) to save a copy of your pixel art as an image file.
