Scaling the Image

By default, images you display in the `canvas` are displayed at whatever size they were created at. You can change that easily by specifying the final image size as part of your `drawImage` call. That might be news to you, for the `drawImage` method we looked at earlier takes only an image source and position values as its arguments:

```
context.drawImage(image, x, y);
```

As it turns out, our `drawImage` method has a few variants it can dress up as. An expanded variant of it allows you to also specify the width and height values you would like to display the image in:

```
context.drawImage(image, x, y, width, height);
```

If we took our earlier example and wanted our image to be displayed as a tiny 50x50 icon, our code would look like this:

HTML JavaScript

javascript

```
var canvas = document.querySelector("#myCanvas");
var context = canvas.getContext("2d");

var myImage = new Image();
myImage.src = "/udata/4WzbwO9WLQq/87.PNG";
myImage.addEventListener("load", loadImage, false);

function loadImage(e) {
  context.drawImage(myImage, 0, 0, 50, 50);
}
```

output

Notice that our `drawImage` call specifies a width and height value of **50**.

Even though we are inside a `canvas` element, standard image scaling logic applies. Scaling down larger images is good. Scaling up smaller images is bad and might result in weird artifacts and blurry lines.

Speaking of blurry lines and weird artifacts, the `canvas` actually smoothens out your images when they are scaled. This smoothing is the default behavior, and you can disable this behavior by setting the `imageSmoothingEnabled` property on your drawing context to **false**:

```
context.mozImageSmoothingEnabled = false;
context.webkitImageSmoothingEnabled = false;
context.msImageSmoothingEnabled = false;
context.imageSmoothingEnabled = false;
```

This property is still pretty new, so you should specify the vendor-prefixed versions of this property (as shown) until the browser support catches up!