Displaying an Image

WE'LL COVER THE FOLLOWING

- Finding an Image
- Creating the Image Object
- Ensuring the Image Has Loaded
- Displaying the Image...for Realz!

Here is the situation. You have an image on your server that you'd like to display inside your `canvas` element. From the previous section, you kinda know that there are two steps involved. You also know that the `drawImage` method is going to be used somewhere. If you are with me so far, that's good! The next few sections will be a breeze.

Finding an Image

The first thing you need is an image to display. You can display almost any image format your browser supports such as GIF, JPEG, PNG, SVG, etc. For walking through all of this with you, the image I will be using looks as follows:

This image is a physical file that lives at the following location. The images you can use don't have to be actual files. I know that sounds crazy, but your canvas can work with "images" that are base-64 encoded. Your image can be a frame from a `video` element. Your image can even by another `canvas` element

where what is drawn there is the visual you have to work with.

For now, let's keep things simple and stick with a physical file. It will be the most common image type you'll be working with anyway, so it can't hurt to start with that :P

Creating the Image Object

To get our image to display on the `canvas`, we need a JavaScript representation of it. The way you do that is by creating an `Image` object and setting the `src` property to the path where the image lives.

In code, this will look as follows:

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

var myImage = new Image();
myImage.src = "images/orange.svg";
```

The first two lines are your standard `canvas` initialization stuff. If you aren't familiar with what these two lines do, refer back to the Getting Started Guide. The next two lines is where things start getting interesting:

```
var myImage = new Image();
myImage.src = "images/orange.svg";
```

In the first line we create a new `Image` object called `myImage`. In the second line, we set this object's `src` property to the path our image lives. This can be a relative path as I've shown. You can also specify an absolute path if you prefer, but ensure the domain you are specifying is the same as the domain your `canvas` element is in. There are some security issues you sorta kinda might run into otherwise.

Ensuring the Image Has Loaded

In the previous section, you may have thought that setting the `src` attribute is pretty much all you would need to do in order to get your image loaded and start working against that loaded image. As it turns out, it doesn't quite work that way. Because of bandwidth, network latency, and a bunch of other excuses, we need to actually ensure the image is loaded before we do anything

excuses, we need to actually ensure the image is loaded before we do anything

that relies on it such as writing some code to get that image drawn to our `canvas`.

The way we do that is by listening for the **load** event on our `Image` object and calling an event handler once that event is overheard. All of that translates into the lines 3-7 of code that you should add to your document:

```
var myImage = new Image();
myImage.src = "images/orange.svg";
myImage.addEventListener("load", loadImage, false);

function loadImage(e) {

}
```

What these lines of code do is ensure the `loadImage` event handler is called when the **load** event on our `Image` object is overheard. That solves our problem of ensuring we have a way of running code only after our image has loaded. With that done, all that remains is to...

Displaying the Image...for Realz!

We created our `Image` object. We specified the image source. We even wrote some code to ensure the image was loaded before we took another step forward. All that remains is to actually display our image inside our `canvas` element. This part is actually a bit anticlimatic.

Inside the `loadImage` function, make a call to `drawImage` with our `myImage` object as the image source and a value of **0** for the x and y position.

Your code should look similar to line 6:

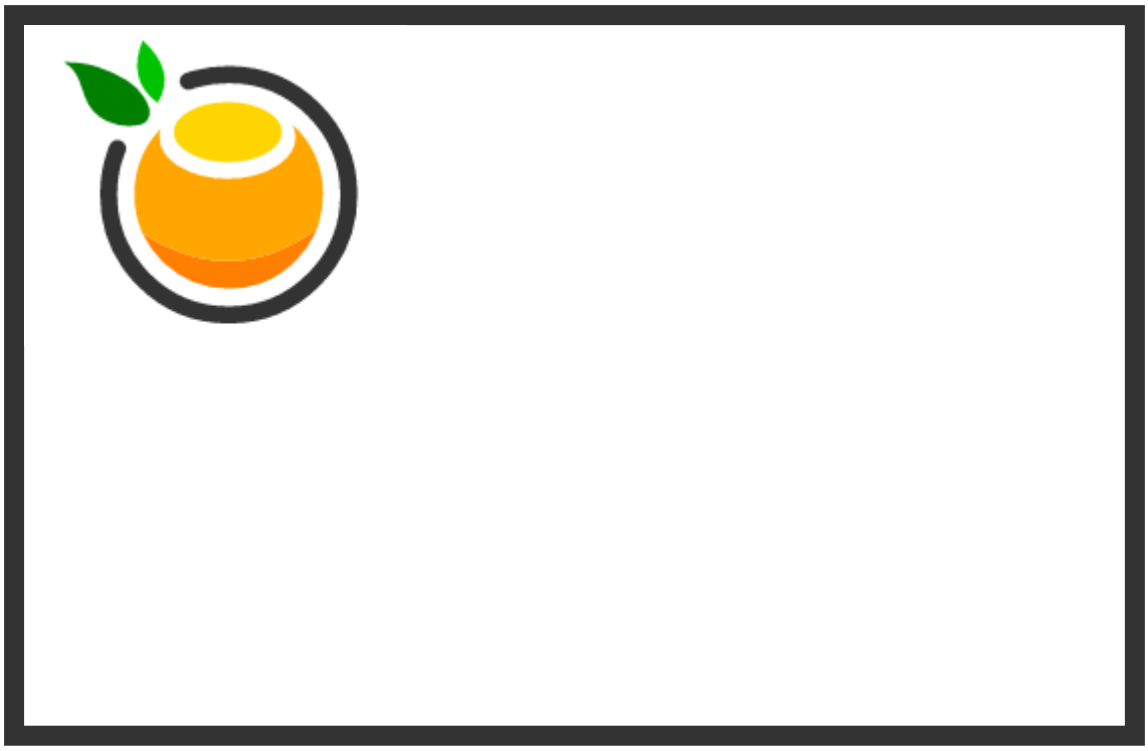
HTML JavaScript

javascript

```
1  var canvas = document.querySelector("#myCanvas");
2  var context = canvas.getContext("2d");
3
4  var myImage = new Image();
5  myImage.src = "/udata/4WzbwO9WLQq/87.PNG";
6  myImage.addEventListener("load", loadImage, false);
7
8  function loadImage(e) {
9    context.drawImage(myImage, 0, 0);
```

```
10 }
```

output



Pretty neat, right? What we've just looked at is the basic steps needed to go from having an image somewhere to having that image displayed inside a `canvas` element. At this point, you've learned the "80%" part of working with images on the canvas. You can safely step away and enjoy the outdoors (or play some video games in the great indoors!) as a celebration of what you've just learned.

With that said, the remaining "20%" is kinda cool as well. In the following sections, let's look at the other things you can do with images on the `canvas`.