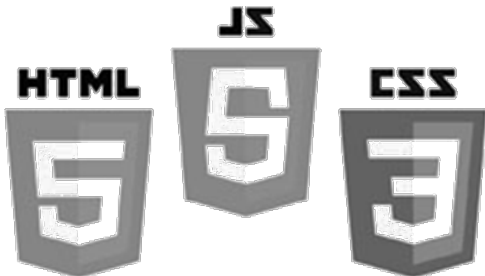


# Image Processing

HTML5 – Canvas

JavaScript

Pixels



# Lecture Objectives

- Provide Examples
  - Basic HTML5 canvas
  - Pixel Manipulation
  - “Loading” Images

# What this is Not

- To complete your projects
  - You must learn more about HTML5 and JavaScript than what is about to be shown
    - This is an “on-your-own” activity
      - Instructor can help, but you must try on your own
    - A prereq to this course is CS 244
      - So you have programmed before
      - This stuff is “easy” compared to that =)
      - Likewise on the math topics
- In Sum: The following is just a place to start
  - More examples will follow throughout the course

# Previously

- Digital Image Processing (DIP)
  - Is computer manipulation of pictures, or images, that have been converted into numeric form
- A Digital Image
  - Is a picture or image converted to numeric form
  - In grey-scale the image can be thought of as
    - 2D function  $f(x, y)$  or a matrix
    - $x$ ,  $y$ , and  $f(x, y)$  are discrete and finite
    - Image size =  $(x_{\max})$  by  $(y_{\max})$ , e.g. 1024 x 768
    - Pixel Intensity Value =  $f(x, y) \in [0, 255]$

# HTML5/JS: Direct Pixel Manipulation

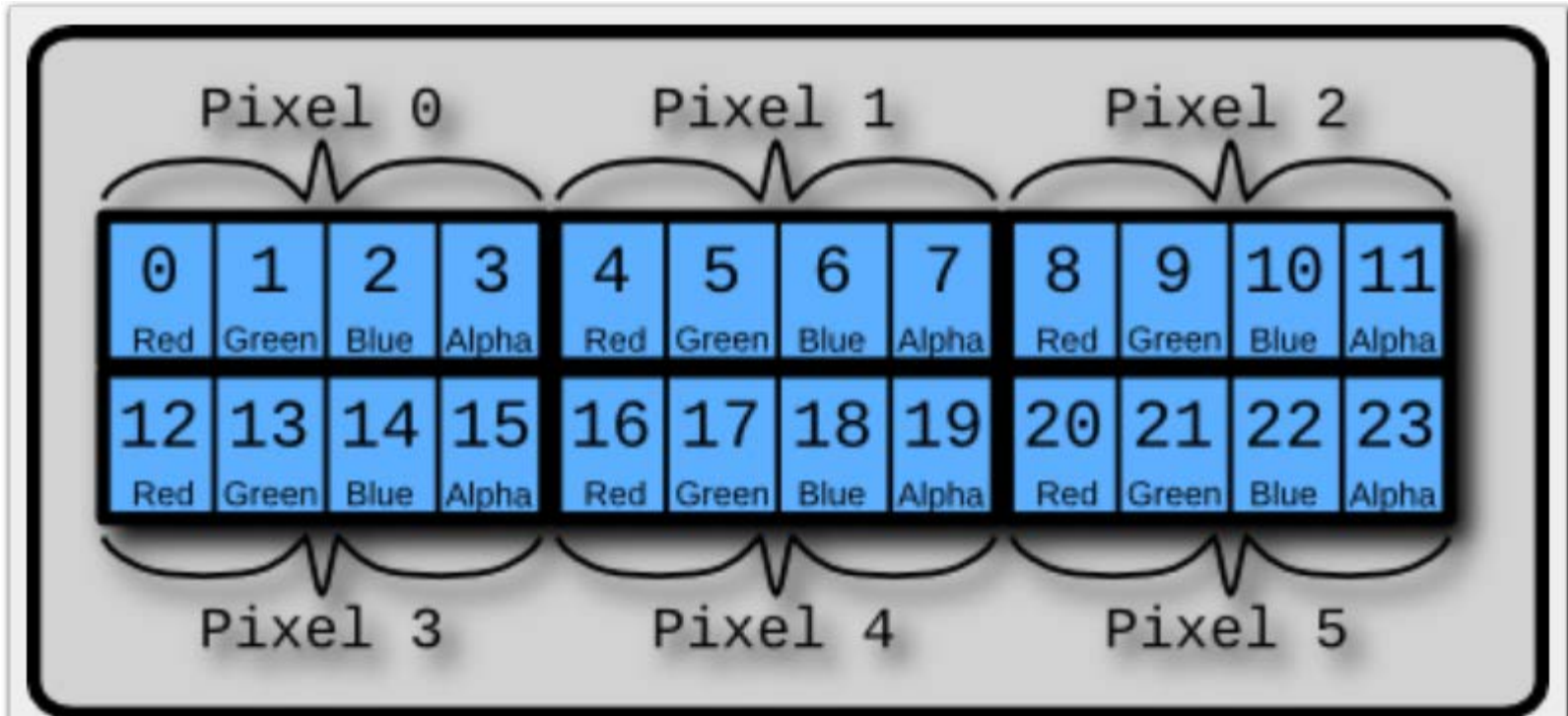
- Three basic things you can do
  - Create an array of empty pixels
  - Get an array of pixels from an existing canvas
  - Set the pixels of a canvas using a pixel array



# Pixel Array

- JavaScript arrays work like C/C++/Java
  - Use the standard accessors to index into the array
    - EX:
      - `mya[0]` is the first element in the array named *mya*
      - `mya[k-1]` is the k-th element in the array named *mya*
  - Pixels in the array are in row-major order
    - with values of 0 to 255
    - where each four-integer group represents the four color channels: Red-Green-Blue-Alpha or RGBA
- *illustration next slide*

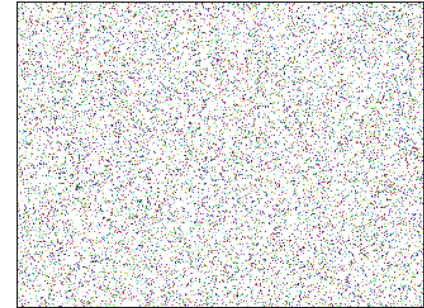
# Pixel Order



Pixel layout in the pixel array for a 3-by-2 image of 6 pixels. Each pixel takes 4 elements in the array for red, green, blue, and alpha, for a total of 24 array elements, 0-23.

# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- Use context's function: *putImageData()*



## createImage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="createImage.js" type="text/javascript"></script>
</head>

<body>
  <div>
    <canvas id="myCanvas" width="320" height="240">
      Your browser does NOT support canvas!
    </canvas>
  </div>
</body>
</html>
```

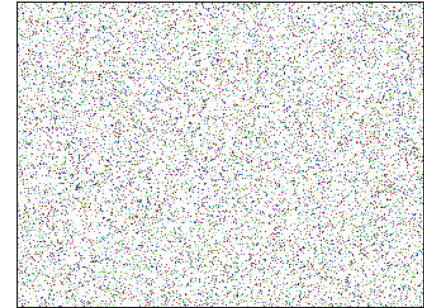
Same basic HTML as previous examples

JavaScript filename changed to createImage.js



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- Use context's function: *putImageData()*



## createImage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="createImage.js" type="text/javascript"></script>
</head>

<body>
  <div>
    <canvas id="myCanvas" width="320" height="240">
      Your browser does NOT support canvas!
    </canvas>
  </div>
</body>
</html>
```

Questions on the HTML file ?

# Creating a Noise Image

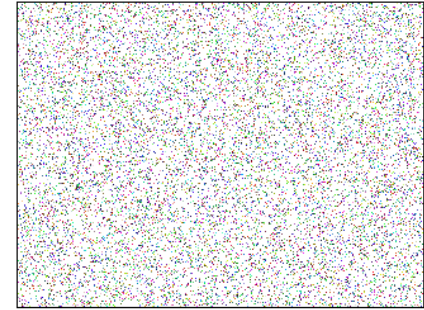
- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- Use context's function: *putImageData()*

createImage.js

```
var theProgram = {  
  Main: function() {  
    theCanvas = document.getElementById("myCanvas");  
    ctx = theCanvas.getContext("2d");
```

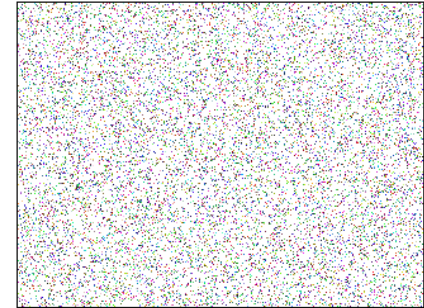
```
// Get the size of the canvas as declared in the HTML  
var width = theCanvas.width;  
var height = theCanvas.height;
```

```
// Create an array of pixels the same size as the canvas  
imageData = ctx.createImageData(width, height);
```



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- Use context's function: *putImageData()*



createImage.js

```
var theProgram = {  
  Main: function() {  
    theCanvas = document.getElementById("myCanvas");  
    ctx = theCanvas.getContext("2d");
```

```
// Get the size of the canvas as declared in the HTML  
var width = theCanvas.width;  
var height = theCanvas.height;
```

```
// Create an array of pixels the same size as the canvas  
imageData = ctx.createImageData(width, height);
```

Note the following

imageData.width → width of the image data in PIXELS

imageData.height → height of the image data in PIXELS

imageData.data → pixel data array of (width \* height \* 4) elements

# Creating a Noise Image

- Use context's function: *createImageData()*
- **Create a new function: *SetPixel()***
- Use context's function: *putImageData()*

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
```

## SetPixel Function

(x, y) is image coordinate

r = red, g = green, b = blue, a = alpha

for alpha --> 255 is opaque and 0 is transparent



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- **Use context's function: *putImageData()***

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

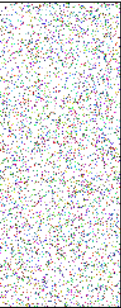
    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
```

```
// Draw random dots
for (var i = 0; i < 15000; i++) // 15000 is arbitrary
{
  var x = Math.random() * width | 0;
  var y = Math.random() * height | 0;
  var r = Math.random() * 256 | 0;
  var g = Math.random() * 256 | 0;
  var b = Math.random() * 256 | 0;
  theProgram.SetPixel(imageData, x, y, r, g, b, 255);
}

// Put the image data onto the canvas
ctx.putImageData(imageData, 0, 0);
},
```



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- **Use context's function: *putImageData()***

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

    // Get the size
    var width = theCanvas.width;
    var height = theCanvas.height;

    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

The " | 0 " is to truncate to integer value

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
```

```
// Draw random dots
for (var i = 0; i < 15000; i++) // 15000 is arbitrary
{
  var x = Math.random() * width | 0;
  var y = Math.random() * height | 0;
  var r = Math.random() * 256 | 0;
  var g = Math.random() * 256 | 0;
  var b = Math.random() * 256 | 0;
  theProgram.SetPixel(imageData, x, y, r, g, b, 255);
}

// Put the image data onto the canvas
ctx.putImageData(imageData, 0, 0);
},
```



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- **Use context's function: *putImageData()***

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

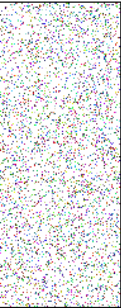
    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

255 is opaque

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
```

```
// Draw random dots
for (var i = 0; i < 15000; i++) // 15000 is arbitrary
{
  var x = Math.random() * width | 0;
  var y = Math.random() * height | 0;
  var r = Math.random() * 256 | 0;
  var g = Math.random() * 256 | 0;
  var b = Math.random() * 256 | 0;
  theProgram.SetPixel(imageData, x, y, r, g, b, 255);
}

// Put the image data onto the canvas
ctx.putImageData(imageData, 0, 0);
},
```



# Creating a Noise Image

- Use context's function: *createImageData()*
- Create a new function: *SetPixel()*
- **Use context's function: *putImageData()***

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

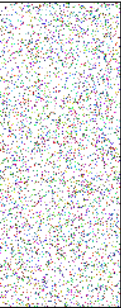
    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

place data on the canvas starting at the upper left corner => (0, 0)

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
```

```
// Draw random dots
for (var i = 0; i < 15000; i++) // 15000 is arbitrary
{
  var x = Math.random() * width | 0;
  var y = Math.random() * height | 0;
  var r = Math.random() * 256 | 0;
  var g = Math.random() * 256 | 0;
  var b = Math.random() * 256 | 0;
  theProgram.SetPixel(imageData, x, y, r, g, b, 255);
}
```

```
// Put the image data onto the canvas
ctx.putImageData(imageData, 0, 0);
},
```





# Creating a Noise Image

## createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);
```

```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
}; // end theProgram Variable
```

```
// Draw random dots
for (var i = 0; i < 15000; i++) // 15000 is arbitrary
{
  var x = Math.random() * width | 0;
  var y = Math.random() * height | 0;
  var r = Math.random() * 256 | 0;
  var g = Math.random() * 256 | 0;
  var b = Math.random() * 256 | 0;
  theProgram.SetPixel(imageData, x, y, r, g, b, 255);
}

// Put the image data onto the canvas
ctx.putImageData(imageData, 0, 0);
},
```



# Creating a Noise Image

createImage.js

```
var theProgram = {
  Main: function() {
    theCanvas = document.getElementById("myCanvas");
    ctx = theCanvas.getContext("2d");

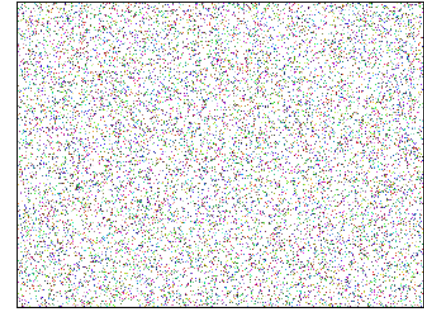
    // Get the size of the canvas as declared in the HTML
    var width = theCanvas.width;
    var height = theCanvas.height;

    // Create an array of pixels the same size as the canvas
    imageData = ctx.createImageData(width, height);

    // Draw random dots
    for (var i = 0; i < 15000; i++) // 15000 is arbitrary
    {
      var x = Math.random() * width | 0;
      var y = Math.random() * height | 0;
      var r = Math.random() * 256 | 0;
      var g = Math.random() * 256 | 0;
      var b = Math.random() * 256 | 0;
      theProgram.SetPixel(imageData, x, y, r, g, b, 255);
    }

    // Put the image data onto the canvas
    ctx.putImageData(imageData, 0, 0);
  },
```

Questions on the JavaScript?



```
SetPixel: function(imageData, x, y, r, g, b, a)
{
  var index = (x + y * imageData.width) * 4;
  imageData.data[index + 0] = r;
  imageData.data[index + 1] = g;
  imageData.data[index + 2] = b;
  imageData.data[index + 3] = a;
},
}; // end theProgram Variable
```

```
window.onload = function()
{
  theProgram.Main();
};
```

# Challenge

- Modify the previous program to display an image that is not so random
  - make it make circles
  - or alternating colored lines
  - or...

# Load Image

- We will now walk through a program that starts like:



Click/tap to see the  
altered image

# Turn it Red

- And when the user clicks on the right side...



## loadImage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta content="text/html; charset=utf-8" http-equiv="content-type">
  <meta name="author" content="Brent Dingle">
  <meta name="description" content="Image Data Example in HTML5 and JavaScript by Brent M Dingle">
  <title>Example: Load Image Data</title>
  <script src="loadImage.js" type="text/javascript"></script>
</head>

<body>
<div>
  <canvas id="mainImageCanvas" width="1000" height="500" style="border:1px solid #000000;">
    Your browser does not support the HTML5 canvas tag.
  </canvas>
</div>
</body>
</html>
```

Same as previous examples... but using a more detailed id name for the canvas

## Questions on the HTML file ?

## loadImage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta content="text/html; charset=utf-8" http-equiv="content-type">
  <meta name="author" content="Brent Dingle">
  <meta name="description" content="Image Data Example in HTML5 and JavaScript by Brent M Dingle">
  <title>Example: Load Image Data</title>
  <script src="loadImage.js" type="text/javascript"></script>
</head>

<body>
<div>
  <canvas id="mainImageCanvas" width="1000" height="500" style="border:1px solid #000000;">
    Your browser does not support the HTML5 canvas tag.
  </canvas>
</div>
</body>
</html>
```

## loadImage.js

```
// -----  
// theProgram (singleton) Object  
// -----  
// -----  
var theProgram =  
{  
  // -----  
  // Pseudo-constants  
  // -----  
  IMAGE_CANVAS_ID: "mainImageCanvas", // canvas id used in html  
  
  // -----  
  // Variables  
  // -----  
  width: 400, // canvas width... likely will be reset  
  height: 400, // canvas height... likely will be reset  
  imageObj: null,  
  
  xOffset: 5,  
  yOffset: 5,  
}
```

Declare and initialize the  
Member variables  
of the  
singleton object *theProgram*



## loadImage.js

```

// Variables
// -----
width:    400,    // canvas width... likely will be reset
height:   400,    // canvas height... likely will be reset
imageObj: null,

xOffset:  5,
yOffset:  5,

// -----
// Functions
// -----
//                               Main
// -----
Main: function()
{
    var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
    canvas.addEventListener('click', theProgram.onClick, false);
    canvas.addEventListener('touch', theProgram.onClick, false);

    this.imageObj = new Image();
    this.imageObj.onload = function()
        {
            theProgram.drawOrigImage();
        };
    this.imageObj.src = 'bobcatCactus.png';
},

// -----
// EventHandler                    onClick

```

Our desired Entry-point function

## loadImage.js

```

// Variables
// -----
width:    400,    // canvas width... likely will be reset
height:   400,    // canvas height... likely will be reset
imageObj: null,

xOffset:  5,
yOffset:  5,

// -----
// Functions
// -----
//                               Main
// -----
Main: function()
{
    var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
    canvas.addEventListener('click', theProgram.onClick, false);
    canvas.addEventListener('touch', theProgram.onClick, false);

    this.imageObj = new Image();
    this.imageObj.onload = function()
        {
            theProgram.drawOrigImage();
        };
    this.imageObj.src = 'bobcatCactus.png';
},

// -----
// EventHandler                    onClick

```

Register a call-back function for click and touch events

## loadImage.js

```
// Variables
// -----
width:    400,    // canvas width... likely will be reset
height:   400,    // canvas height... likely will be reset
imageObj: null,

xOffset:  5,
yOffset:  5,

// -----
// Functions
// -----
//                               Main
// -----
Main: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  canvas.addEventListener('click', theProgram.onClick, false);
  canvas.addEventListener('touch', theProgram.onClick, false);

  this.imageObj = new Image();
  this.imageObj.onload = function()
  {
    theProgram.drawOrigImage();
  };
  this.imageObj.src = 'bobcatCactus.png';
}

// -----
// EventHandler                   onClick
```

Load the image file named:  
bobcatCactus.png

Register a callback function  
to draw it to the canvas when the  
file is loaded into the browser

*Yes it is important to set the onload function  
BEFORE setting the src*

## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawBWImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("black and white image", this.imageObj.width+20, 100);
},

// -----
// drawBWImage -- must call drawOrigImage before this
```

Function that is called when  
the image file named:

*bobcatCactus.png*

is loaded into the browser

## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawBWImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("black and white image", this.imageObj.width+20, 100);
},

// -----
// drawBWImage -- must call drawOrigImage before this
```

Gets the canvas and context

## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawBWImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("black and white image", this.imageObj.width+20, 100);
},

// -----
// drawBWImage -- must call drawOrigImage before this
```

Clears the canvas

## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawBWImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("black and white image", this.imageObj.width+20, 100);
},

// -----
// drawBWImage -- must call drawOrigImage before this
```

Draws the loaded image

## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawBWImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("altered image", this.imageObj.width+20, 100);
},

// -----
// drawBWImage -- must call drawOrigImage before this
```

Displays message telling user to click or tap on the area to see the altered version of the image



## loadImage.js

```
// EventHandler                                onClick
// -----
onClick: function(e)
{
  theProgram.drawOrigImage();
  theProgram.drawAlteredImage();
},

// -----
// drawOrigImage
// -----
drawOrigImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  ctx.drawImage(this.imageObj, this.xOffset, this.yOffset);

  ctx.font="16px Georgia";
  ctx.fillText("Click/tap to see the", this.imageObj.width+20, 80);
  ctx.fillText("altered image", this.imageObj.width+20, 100);
},

// -----
// drawAlteredImage -- must call drawOrigImage before this
```

This function is called when the user clicks or taps on the area

It will draw the original image and then the altered image

## loadImage.js

```
// drawAlteredImage -- must call drawOrigImage before this
// -----
drawAlteredImage: function()
{
    var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
    var ctx = canvas.getContext('2d');

    // Get the image data from the canvas context
    var imageData = ctx.getImageData(this.xOffset, this.yOffset,
        this.imageObj.width, this.imageObj.height);
    var data = imageData.data;

    // This gives us the image data in RGBA format - one byte for each
    // So to iterate over all pixels we would do:
    //for(var i = 0, n = data.length; i < n; i += 4)
    // {
    //     // var red = data[i];
    //     // var green = data[i + 1];
    //     // var blue = data[i + 2];
    //     // var alpha = data[i + 3];
    // }

    // We must 'create' a new image
```

Function to draw the altered image

## loadImage.js

```
// drawAlteredImage -- must call drawOrigImage before this
// -----
drawAlteredImage: function()
{
  var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
  var ctx = canvas.getContext('2d');
```

```
// Get the image data from the canvas context
var imageData = ctx.getImageData(this.xOffset, this.yOffset,
                                this.imageObj.width, this.imageObj.height);
var data = imageData.data;
```

```
// This gives us the image data in RGBA format - one byte for each
// So to iterate over all pixels we would do:
//for(var i = 0, n = data.length; i < n; i += 4)
// {
//   // var red = data[i];
//   // var green = data[i + 1];
//   // var blue = data[i + 2];
//   // var alpha = data[i + 3];
// }

// We must 'create' a new image
```

Get the original image  
pixel data array

**CAUTION:**

This assumes the image is drawn on the canvas already, at (xOffset, yOffset) and is size *width* x height

Hence the need for drawOrigImage to be called before this function

There are other ways to do this and avoid this assumption

# loadImage.js

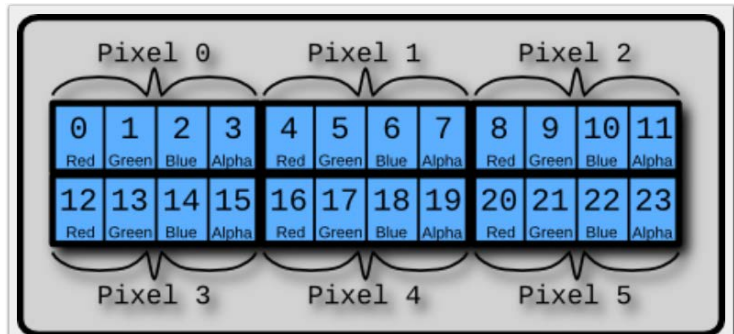
```
// drawAlteredImage -- must call drawOrigImage before this
// -----
drawAlteredImage: function()
{
    var canvas = document.getElementById(this.IMAGE_CANVAS_ID);
    var ctx = canvas.getContext('2d');

    // Get the image data from the canvas context
    var imageData = ctx.getImageData(this.xOffset, this.yOffset,
        this.imageObj.width, this.imageObj.height);
    var data = imageData.data;
```

```
// This gives us the image data in RGBA format - one byte for each
// So to iterate over all pixels we would do:
//for(var i = 0, n = data.length; i < n; i += 4)
//{
    // var red = data[i];
    // var green = data[i + 1];
    // var blue = data[i + 2];
    // var alpha = data[i + 3];
//}
```

```
// We must 'create' a new image
```

## Recall the ordering of the pixel data



Pixel layout in the pixel array for a 3-by-2 image of 6 pixels. Each pixel takes 4 elements in the array for red, green, blue, and alpha, for a total of 24 array elements, 0-23.

## loadImage.js

```

    // var alpha = data[i + 3];
    //}

    // We must 'create' a new image
    var SecondImageData = ctx.createImageData(this.imageObj.width, this.imageObj.height);

    for (var pixelIndex=0; pixelIndex < SecondImageData.data.length; pixelIndex += 4)
    {
        SecondImageData.data[pixelIndex  ] = data[pixelIndex] * 3;    // More red
        SecondImageData.data[pixelIndex + 1] = data[pixelIndex +1] / 2; // Less green
        SecondImageData.data[pixelIndex + 2] = data[pixelIndex +2] / 2; // Less blue
        SecondImageData.data[pixelIndex + 3] = data[pixelIndex +3];    // alpha unchanged
    }

    // Want to draw the altered image to the right of original
    // Offset gives spacing between
    var SecondImageX = 2*this.xOffset + this.imageObj.width;
    var SecondImageY = this.yOffset;
    ctx.putImageData(SecondImageData, SecondImageX, SecondImageY);

},

}; // end theProgram variable

//-----
//                window.ONLOAD

```

Create a new pixel data array  
of the same size as the original

## loadImage.js

```

    // var alpha = data[i + 3];
    // }

    // We must 'create' a new image
    var SecondImageData = ctx.createImageData(this.imageObj.width, this.imageObj.height);

```

```

for (var pixelIndex=0; pixelIndex < SecondImageData.data.length; pixelIndex += 4)
{
    SecondImageData.data[pixelIndex  ] = data[pixelIndex] * 3;    // More red
    SecondImageData.data[pixelIndex + 1] = data[pixelIndex + 1] / 2; // Less green
    SecondImageData.data[pixelIndex + 2] = data[pixelIndex + 2] / 2; // Less blue
    SecondImageData.data[pixelIndex + 3] = data[pixelIndex + 3];    // alpha unchanged
}

```

```

// Want to draw the altered image to the right of original
// Offset gives spacing between
var SecondImageX = 2*this.xOffset + this.imageObj.width;
var SecondImageY = this.yOffset;
ctx.putImageData(SecondImageData, SecondImageX, SecondImageY);

```

```

},

```

```

}; // end theProgram variable

```

```

// -----
//                               window.ONLOAD

```

Loop through the pixel data  
 Increasing the red values  
 Decreasing the green and blue  
  
 Alpha remains the same

## loadImage.js

```

    // var alpha = data[i + 3];
    // }

    // We must 'create' a new image
    var SecondImageData = ctx.createImageData(this.imageObj.width, this.imageObj.height);

    for (var pixelIndex=0; pixelIndex < SecondImageData.data.length; pixelIndex += 4)
    {
        SecondImageData.data[pixelIndex  ] = data[pixelIndex] * 3;    // More red
        SecondImageData.data[pixelIndex + 1] = data[pixelIndex + 1] / 2; // Less green
        SecondImageData.data[pixelIndex + 2] = data[pixelIndex + 2] / 2; // Less blue
        SecondImageData.data[pixelIndex + 3] = data[pixelIndex + 3];    // alpha unchanged
    }

    // Want to draw the altered image to the right of original
    // Offset gives spacing between
    var SecondImageX = 2*this.xOffset + this.imageObj.width;
    var SecondImageY = this.yOffset;
    ctx.putImageData(SecondImageData, SecondImageX, SecondImageY),

    },

}; // end theProgram variable

// -----
//                               window.ONLOAD

```

Calculate the offset position to draw the second image

→ calculate where the upper left corner of the new image is to be placed on the canvas

Answer: at position  
(*secondImageX*, *secondImageY*)  
in canvas coordinates

## loadImage.js

```

    // var alpha = data[i + 3];
    // }

    // We must 'create' a new image
    var SecondImageData = ctx.createImageData(this.imageObj.width, this.imageObj.height);

    for (var pixelIndex=0; pixelIndex < SecondImageData.data.length; pixelIndex += 4)
    {
        SecondImageData.data[pixelIndex  ] = data[pixelIndex] * 3;    // More red
        SecondImageData.data[pixelIndex + 1] = data[pixelIndex + 1] / 2; // Less green
        SecondImageData.data[pixelIndex + 2] = data[pixelIndex + 2] / 2; // Less blue
        SecondImageData.data[pixelIndex + 3] = data[pixelIndex + 3];    // alpha unchanged
    }

    // Want to draw the altered image to the right of original
    // Offset gives spacing between
    var SecondImageX = 2*this.xOffset + this.imageObj.width;
    var SecondImageY = this.yOffset;
    ctx.putImageData(SecondImageData, SecondImageX, SecondImageY);
},

}; // end theProgram variable

// -----
//                               window.ONLOAD

```

Draw the altered image  
on the canvas  
at the calculated offset



## loadImage.js

```
SecondImageData.data[pixelIndex + 2] = data[pixelIndex + 2] / 2; // Less blue
SecondImageData.data[pixelIndex + 3] = data[pixelIndex + 3]; // alpha unchanged
}

// Want to draw the altered image to the right of original
// Offset gives spacing between
var SecondImageX = 2*this.xOffset + this.imageObj.width;
var SecondImageY = this.yOffset;
ctx.putImageData(SecondImageData, SecondImageX, SecondImageY);

},

}; // end theProgram variable

// -----
//                               window.ONLOAD
// -----
window.onload = function()
{
    // Initialize and Start the game
    theProgram.Main();
};

// -----
// -----
// end of file
```

And make sure when all the contents of the HTML page are loaded into the browser that our program begins

i.e. call *theProgram.Main()*

## loadImage.js

```

SecondImageData.data[pixelIndex + 2] = data[pixelIndex + 2] / 2; // Less blue
SecondImageData.data[pixelIndex + 3] = data[pixelIndex + 3]; // alpha unchanged
}

```

```

// Want to draw the altered image to the right of original
// Offset gives spacing between
var SecondImageX = 2*this.xOffset + this.imageObj.width;
var SecondImageY = this.yOffset;
ctx.putImageData(SecondImageData, SecondImageX, SecondImageY);

```

```

},

```

```

}; // end theProgram variable

```

```

// -----
//                               window.ONLOAD
// -----

```

```

window.onload = function()

```

```

{

```

```

    // Initialize and Start the game
    theProgram.Main();

```

```

};

```

```

// -----
// -----
// end of file

```

Any questions on the JavaScript?



# Challenge

- Alter the above program to change the color image into a grey-scale image
- Apply a luminance algorithm of:
  - $0.3 * \text{red} + 0.59 * \text{green} + 0.11 * \text{blue}$
  - Also, come to the next class with an explanation of why the green would be such a higher weight

# Questions?



- Beyond D2L
  - Examples and information can be found online at:
    - <http://docdingle.com/teaching/cs.html>
  
- *Continue to more stuff as needed*

# Extra Reference Stuff Follows



# Credits

- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing*, Gonzalez and Woods
- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Bernd Girod, Stanford University, 2007
  - Shreekanth Mandayam, Rowan University, 2009
  - Igor Aizenberg, TAMUT, 2013
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015
  - Sinisa Todorovic, Oregon State, 2015
  - Beej's Bit Bucket / Tech and Programming Fun
    - <http://beej.us/blog/>
    - <http://beej.us/blog/data/html5s-canvas-2-pixel/>
  - w3schools.com

