Lab 03-Lab 04
Image and color basics Continued

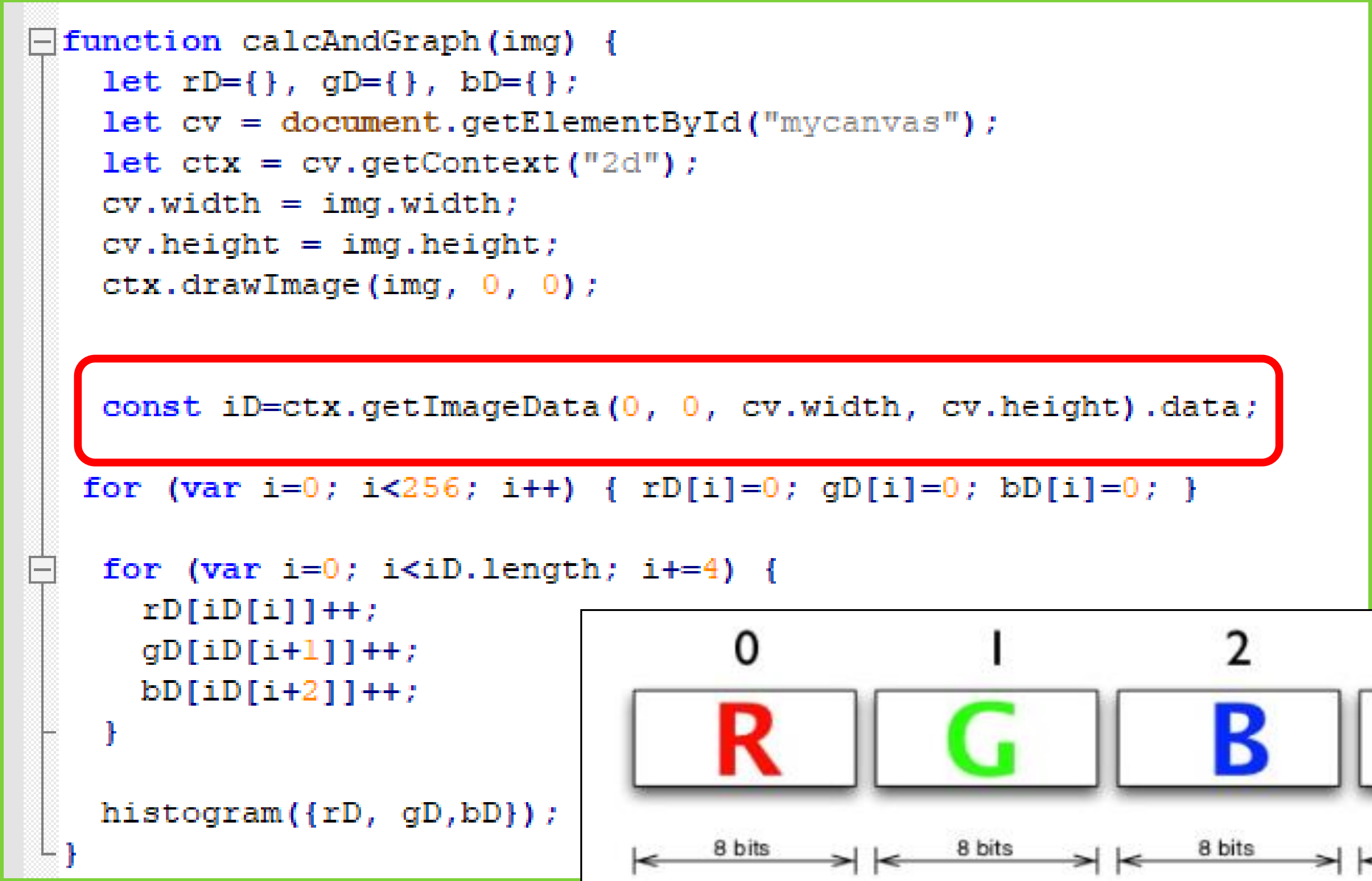
Discussion is based on F.S. Hill Chapter 02,03,10

BSCS – 514 Computer Graphics
Course Supervisor Dr. Humera Tariq

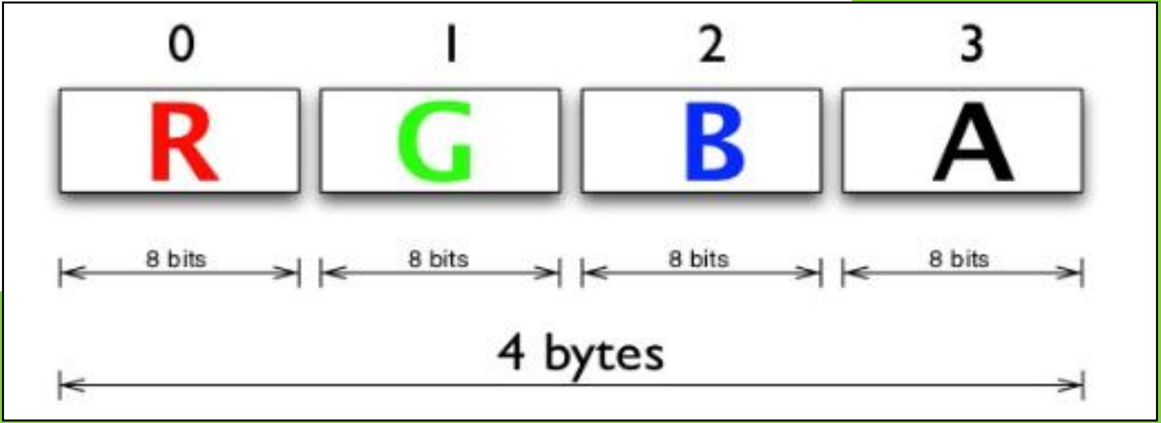
Lab 03, Lab 04 Objectives /Tasks

1. Resizing Rectangle (Chap 3)
2. Viewport and Tiling (Chap3)
3. Aspect Ratio Concept Practice (Chap 2, Chap 3)
3. Refactoring code into mutiple .js scripts
4. Build your own local server by installing node.js
5. Understanding getImagedata(…) (chap 10)
6. Working with Javascript dictionaries

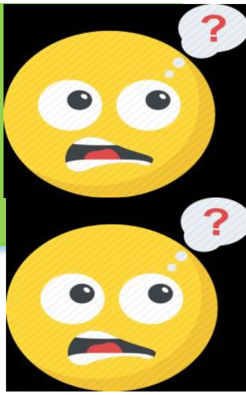
Recall function calcAndGraph(img)

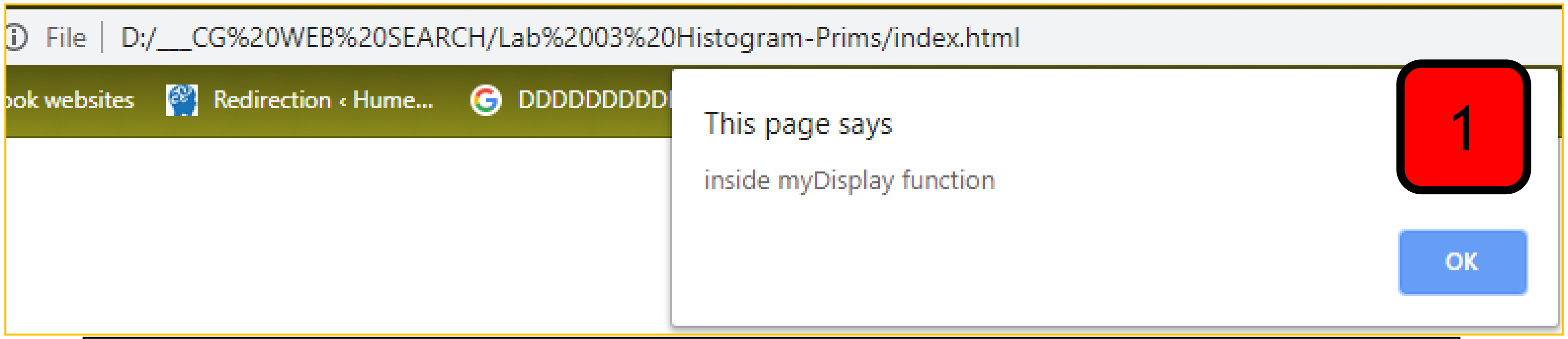


```
function calcAndGraph(img) {
  let rD={}, gD={}, bD={};
  let cv = document.getElementById("mycanvas");
  let ctx = cv.getContext("2d");
  cv.width = img.width;
  cv.height = img.height;
  ctx.drawImage(img, 0, 0);

  const iD=ctx.getImageData(0, 0, cv.width, cv.height).data;
 for (var i=0; i<256; i++) { rD[i]=0; gD[i]=0; bD[i]=0; }

  for (var i=0; i<iD.length; i+=4) {
    rD[iD[i]]++;
    gD[iD[i+1]]++;
    bD[iD[i+2]]++;
  }

  histogram({rD, gD,bD});
}
```



Run index.js
Sample Image and observed problem

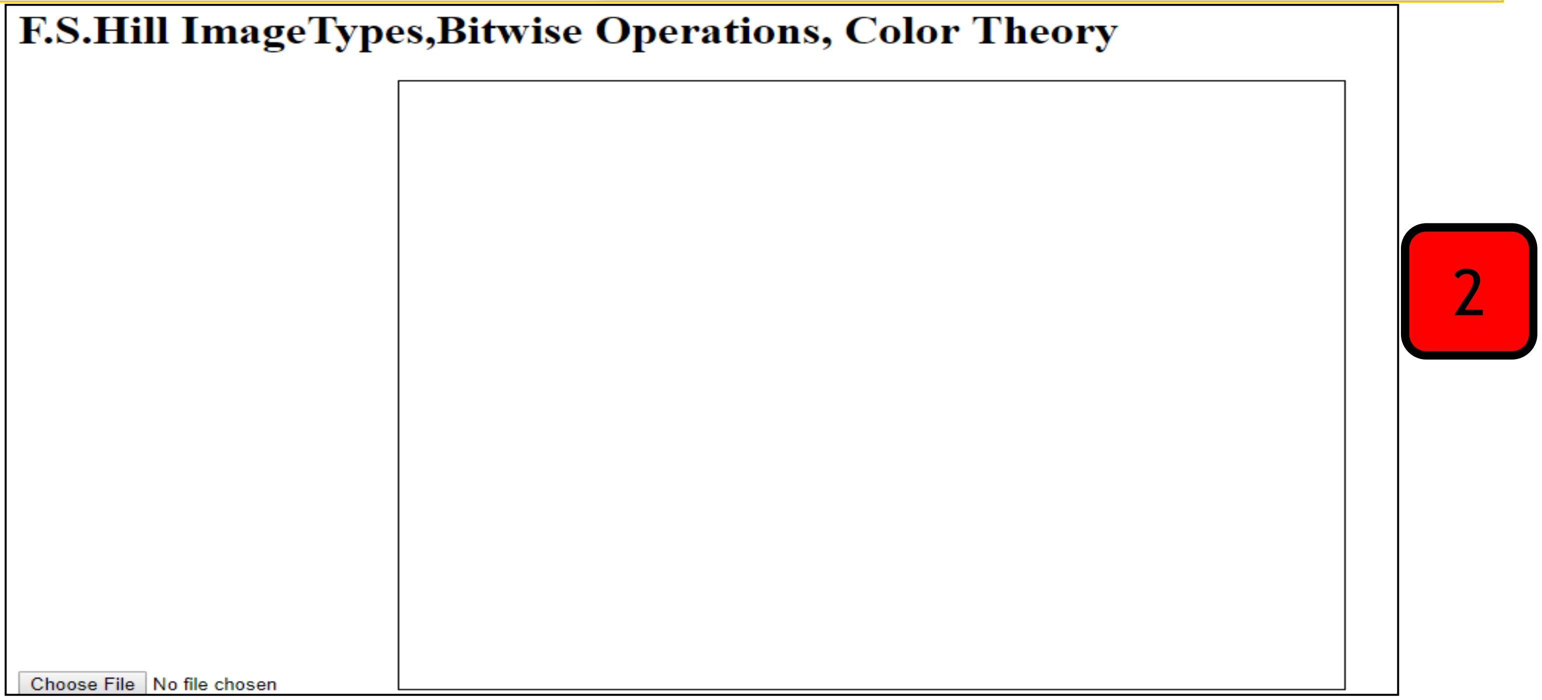




File | D:/___CG%20WEB%20SEARCH/Lab%2003%20Histogram-Prims/index.html

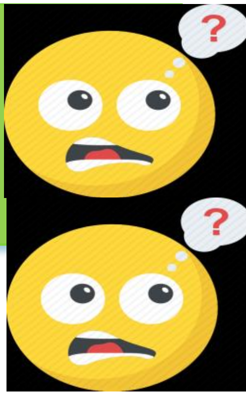
This page says

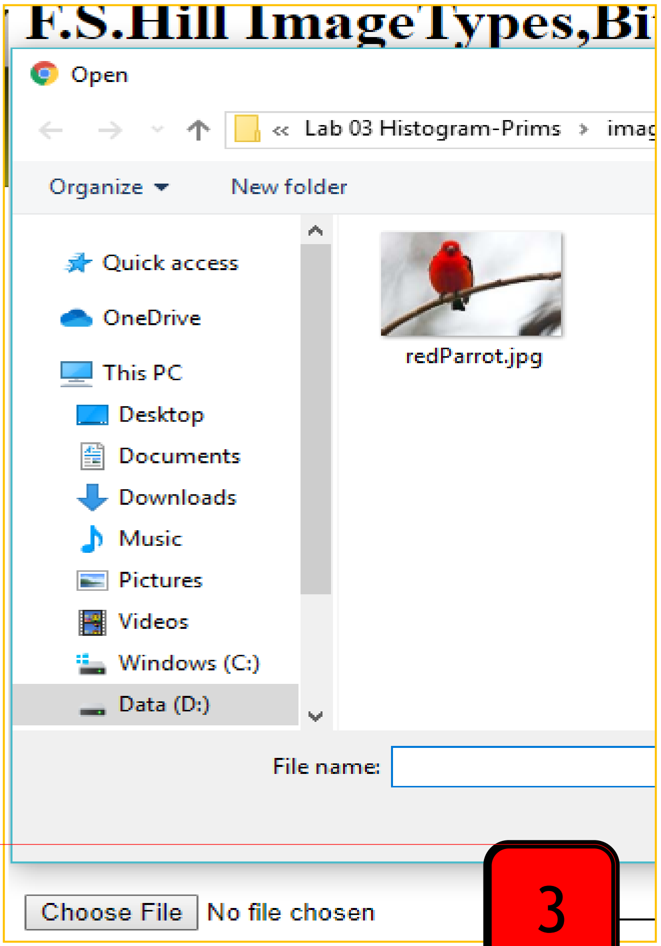
inside myDisplay function

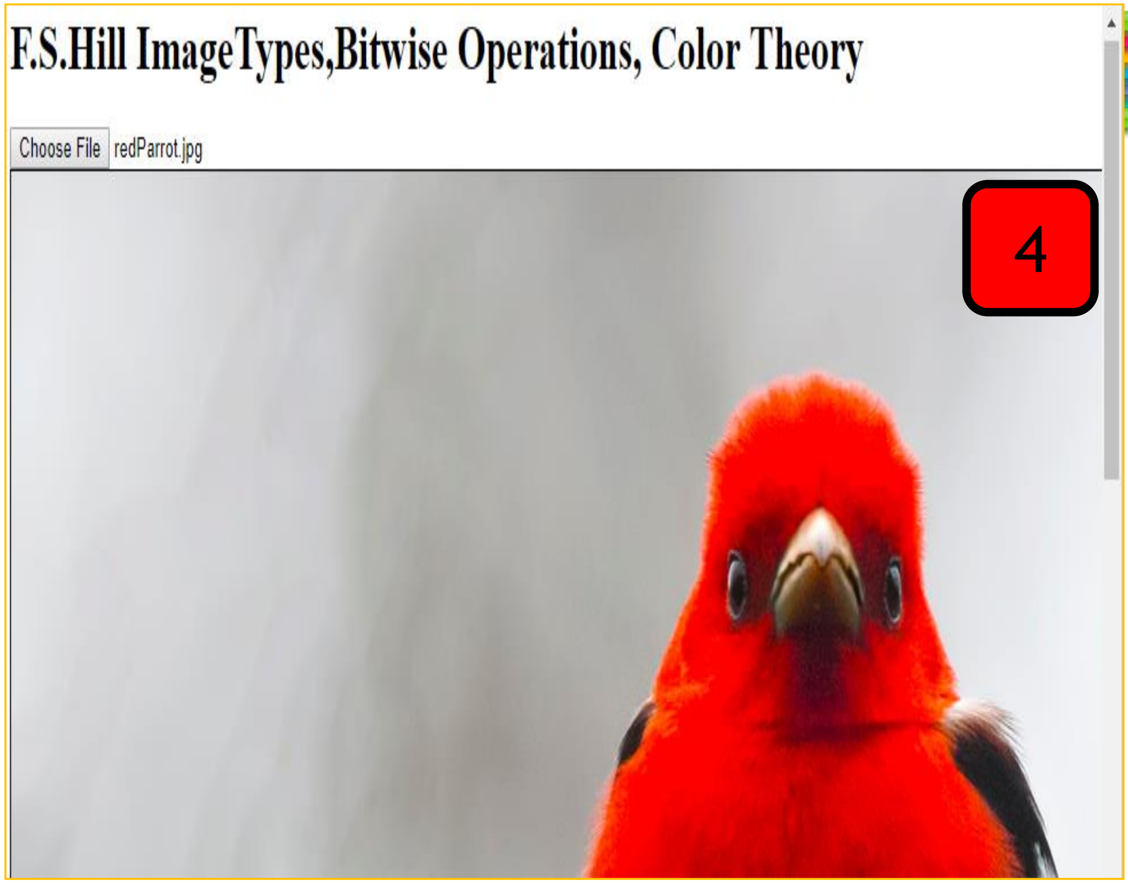


F.S.Hill ImageTypes,Bitwise Operations, Color Theory

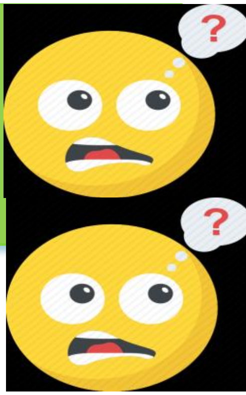
Sample Image and observed problem

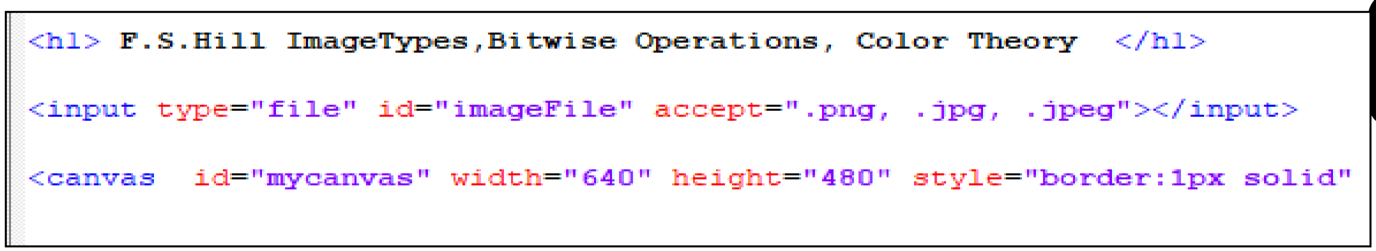



F.S.Hill ImageTypes,Bi


F.S.Hill ImageTypes,Bitwise Operations, Color Theory

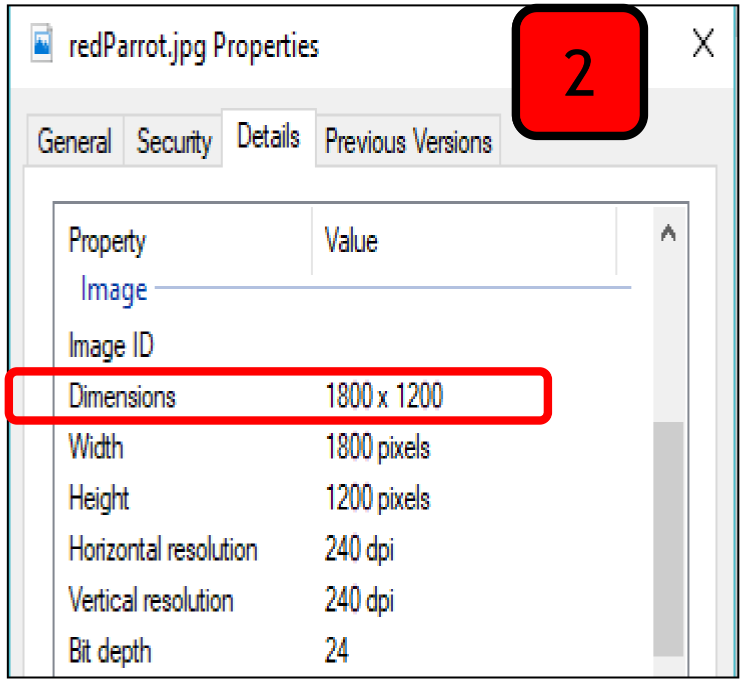
Solving issue 1: Canvas Resizing or Image Resizing





1

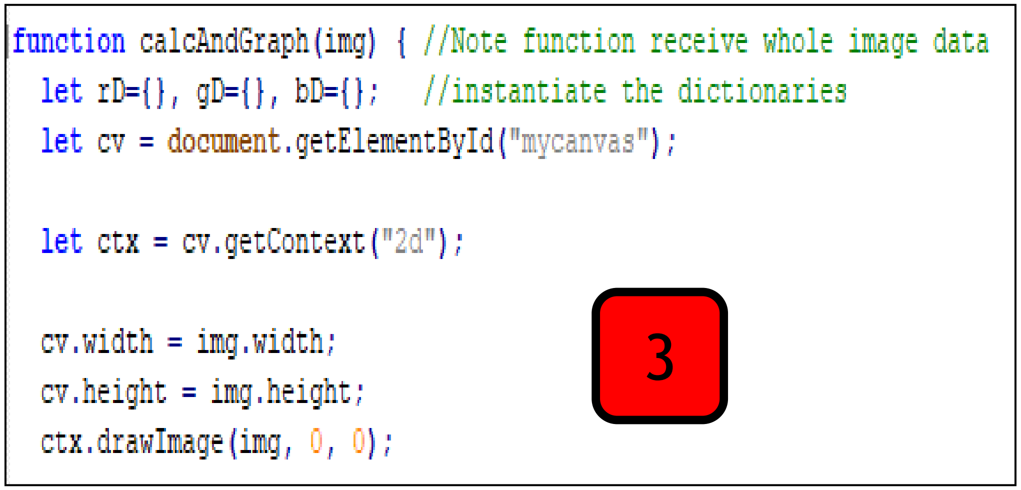
```
<h1> F.S.Hill ImageTypes,Bitwise Operations, Color Theory  </h1>

<input type="file" id="imageFile" accept=".png, .jpg, .jpeg"></input>

<canvas  id="mycanvas" width="640" height="480" style="border:1px solid"
```



2

redParrot.jpg Properties

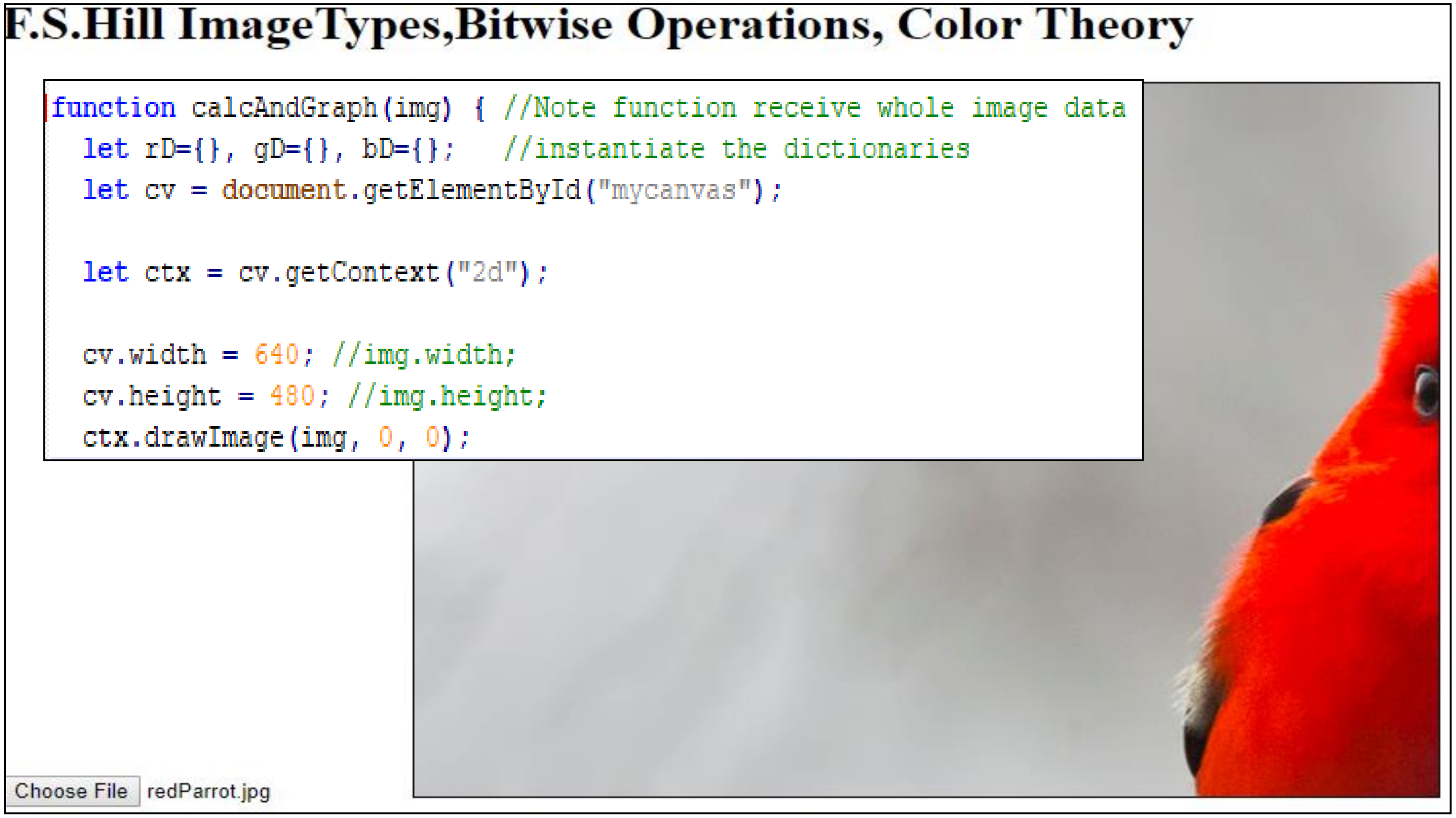
Property	Value
Image	
Image ID	
Dimensions	1800 x 1200
Width	1800 pixels
Height	1200 pixels
Horizontal resolution	240 dpi
Vertical resolution	240 dpi
Bit depth	24



3

```
function calcAndGraph(img) { //Note function receive whole image data
  let rD={}, gD={}, bD={};   //instantiate the dictionaries
  let cv = document.getElementById("mycanvas");

  let ctx = cv.getContext("2d");

  cv.width = img.width;
  cv.height = img.height;
  ctx.drawImage(img, 0, 0);
```

I attempt to change canvas width and height Still I am unhappy


F.S.Hill ImageTypes,Bitwise Operations, Color Theory

```
function calcAndGraph(img) { //Note function receive whole image data
  let rD={}, gD={}, bD={};   //instantiate the dictionaries
  let cv = document.getElementById("mycanvas");

  let ctx = cv.getContext("2d");

  cv.width = 640; //img.width;
  cv.height = 480; //img.height;
  ctx.drawImage(img, 0, 0);
```

Looking for Resize Rectangle problem

1800 x 1200 -----→ 640 x 480

See chapter 03 for aspect Ratio details of World window and Viewport

<u>Learning Practices</u>
Aligned Rectangle,
Rounded Rectangle,
Aspect Ratio problems

Suppose the aspect ratio of the world window is know to be *R*, and the screen window has width *W* and height *H*. There are two distinct situations: the world window may have a larger aspect ratio than the screen window ($R > W/H$), or it may have a smaller aspect ratio ($R < W/H$). The two situations are shown in Figure 3.16.

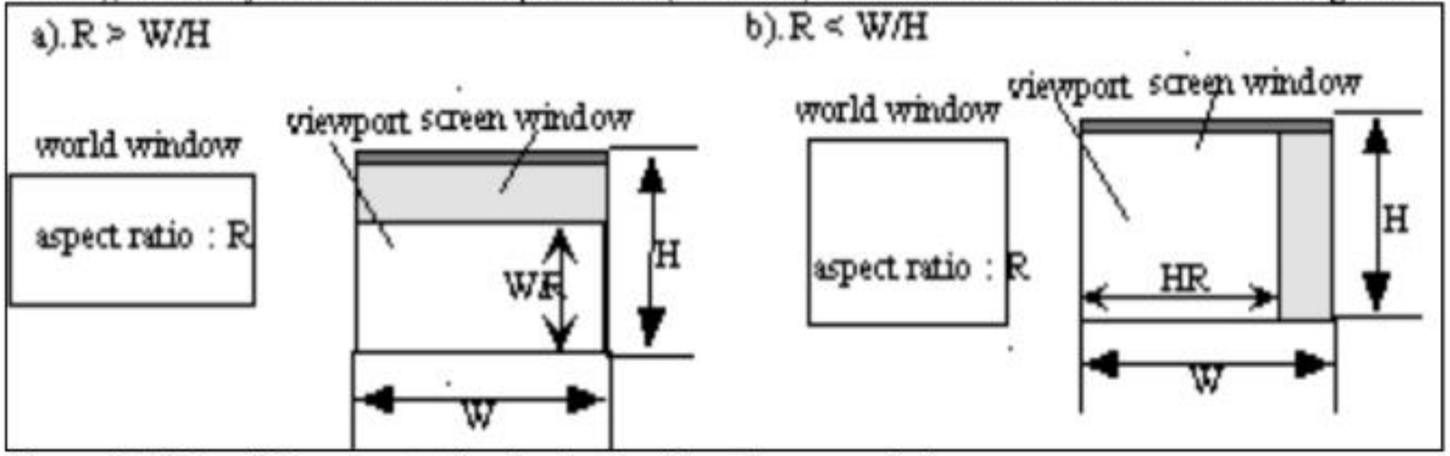
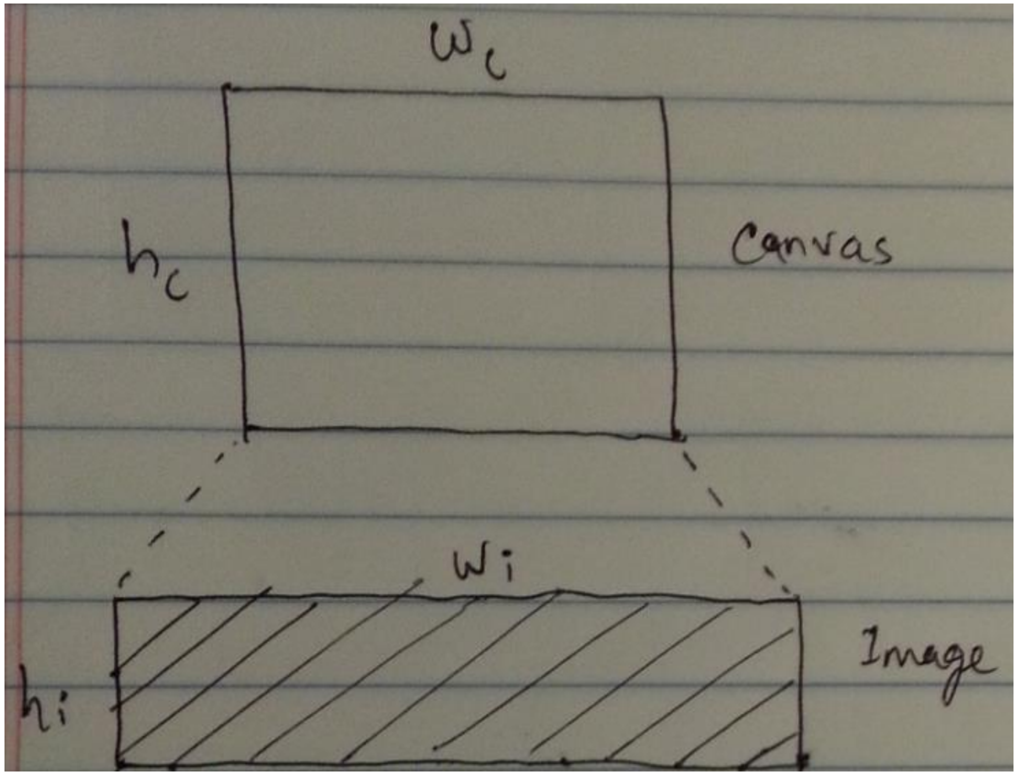


Figure 3.16. Possible aspect ratios for the world and screen windows.

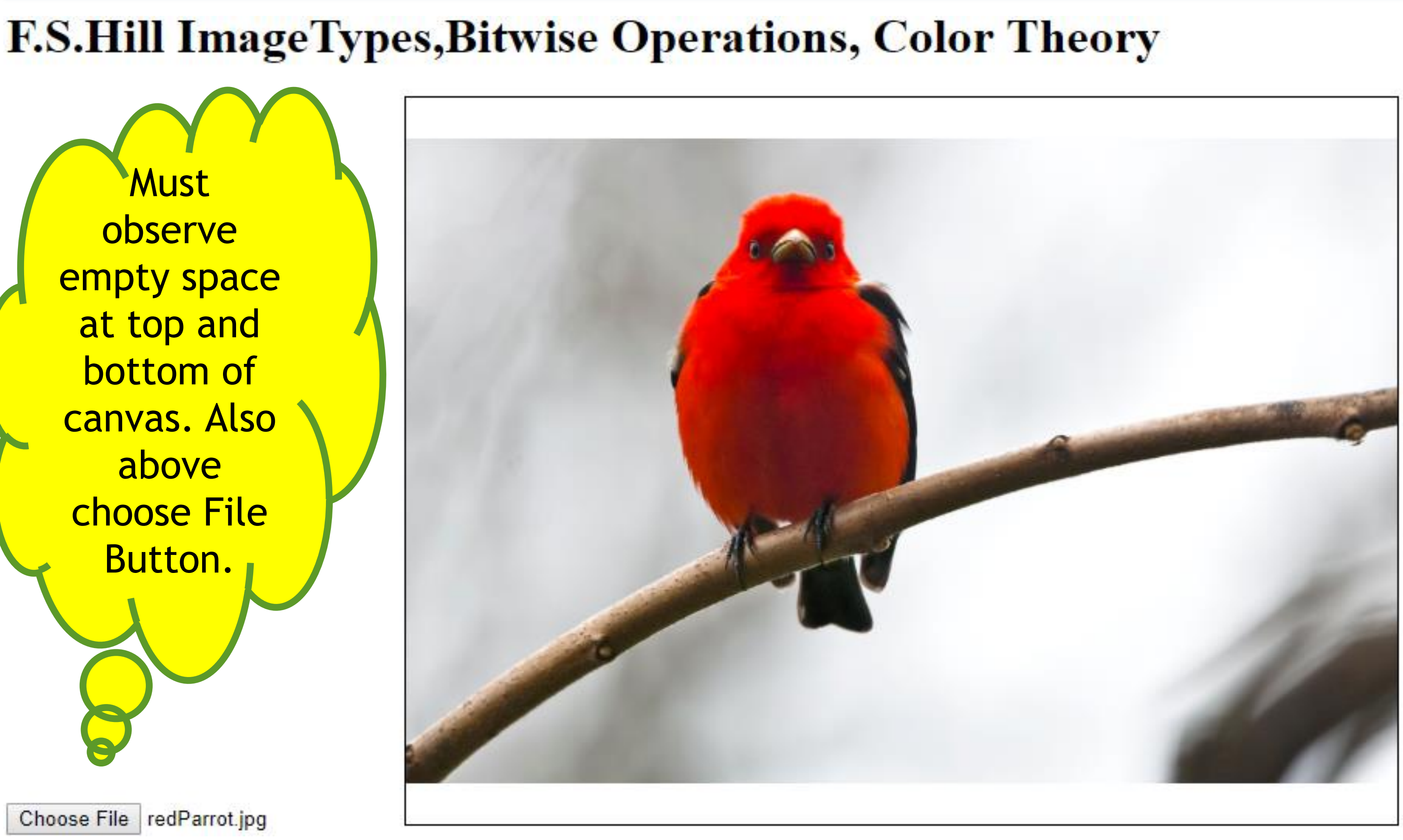
https://sadique.io/blog/2013/10/03/fitting-an-image-in-to-a-canvas-object/

Resizing in Java script

1800 x 1200 -----→ 640 x 480

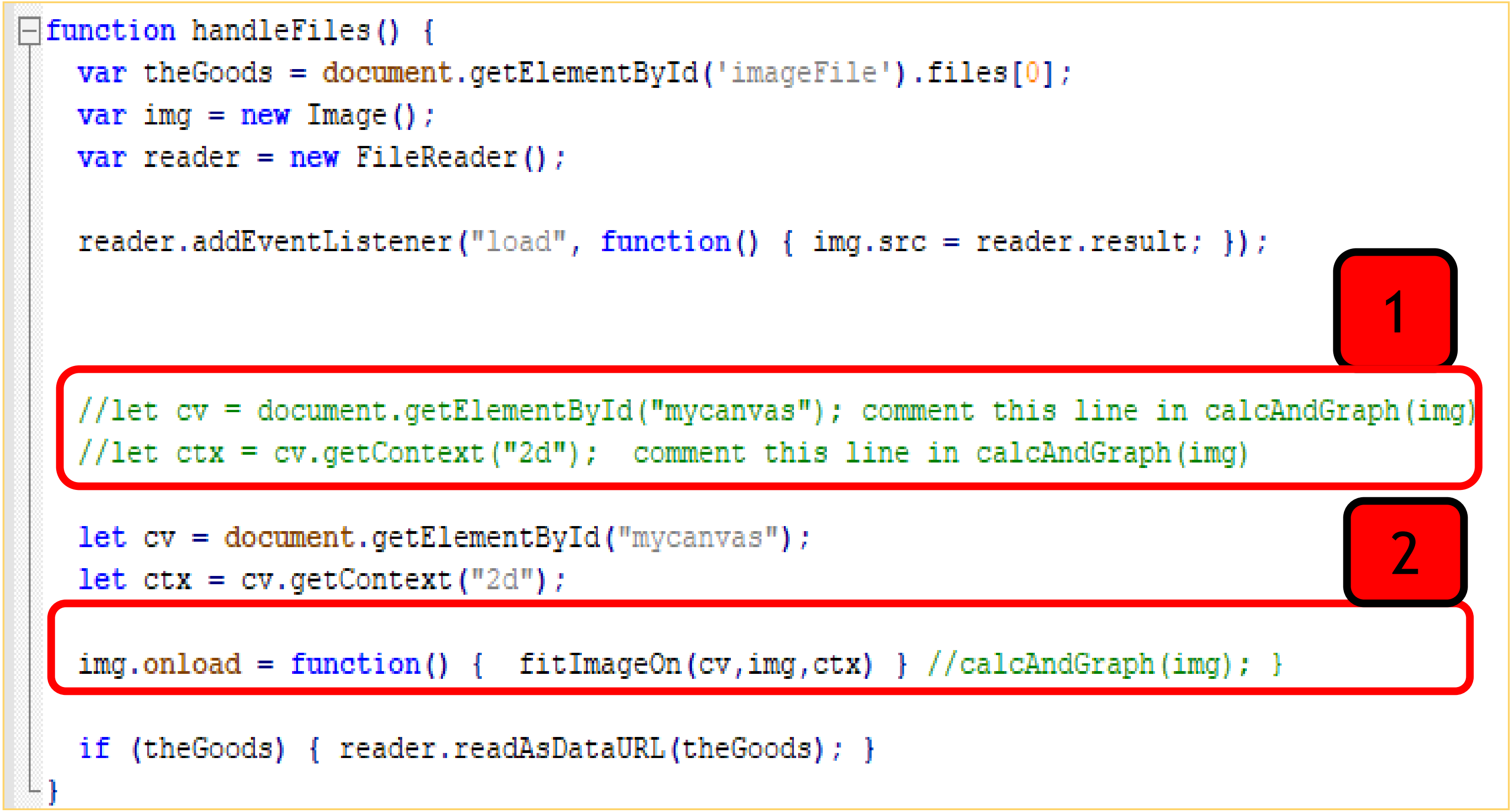


My resizing output (Screen Mgmt)


F.S.Hill ImageTypes,Bitwise Operations, Color Theory
Must observe empty space at top and bottom of canvas. Also above choose File Button.
Choose File
redParrot.jpg

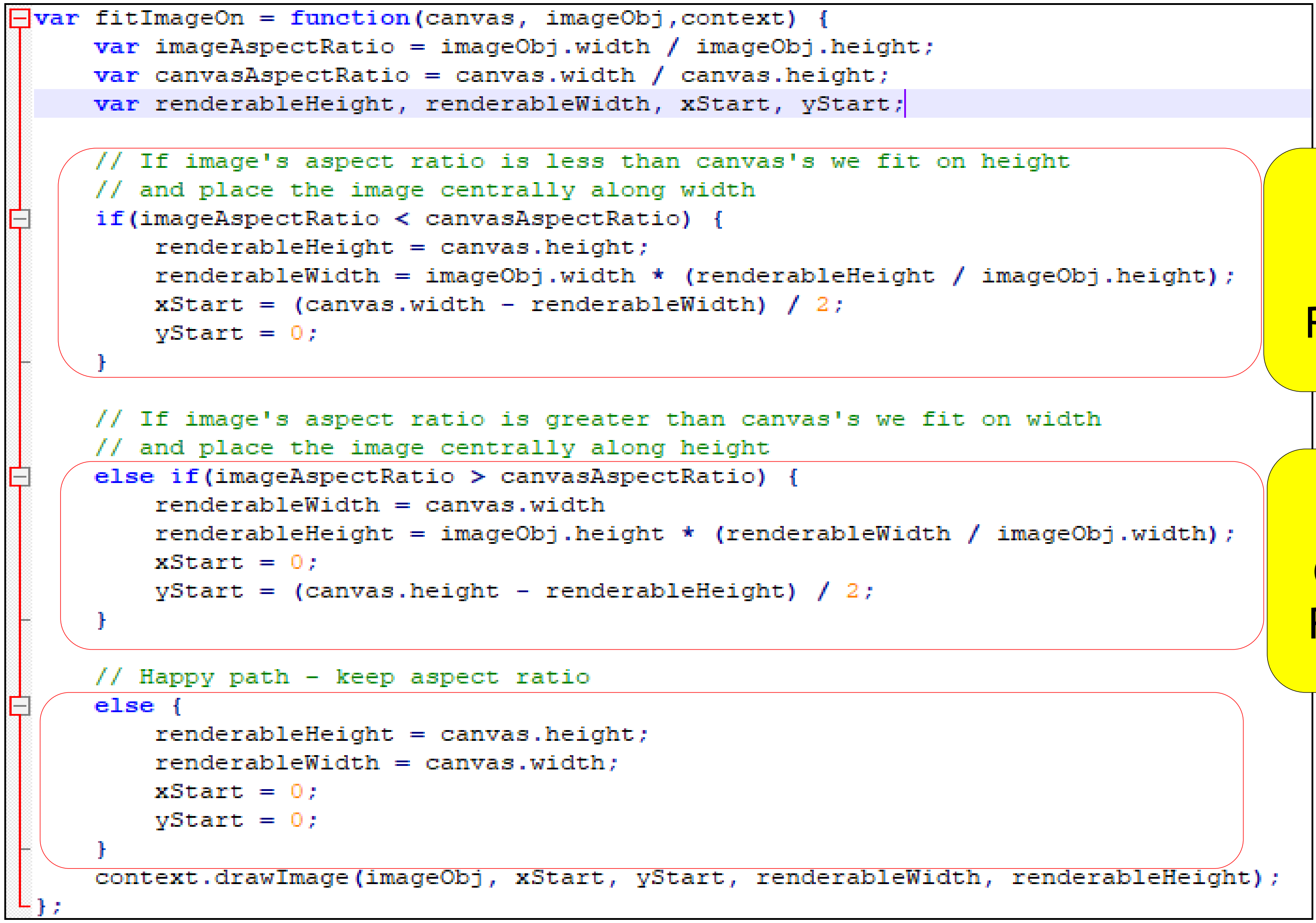
My code changes for resizing

1800 x 1200 -----→ 640 x 480



```
function handleFiles() {
   var theGoods = document.getElementById('imageFile').files[0];
   var img = new Image();
   var reader = new FileReader();

   reader.addEventListener("load", function() { img.src = reader.result; });


   //let cv = document.getElementById("mycanvas"); comment this line in calcAndGraph(img)
   //let ctx = cv.getContext("2d");  comment this line in calcAndGraph(img)

   let cv = document.getElementById("mycanvas");
   let ctx = cv.getContext("2d");

   img.onload = function() {  fitImageOn(cv,img,ctx) } //calcAndGraph(img); }

   if (theGoods) { reader.readAsDataURL(theGoods); }
}
```

My code changes for resizing

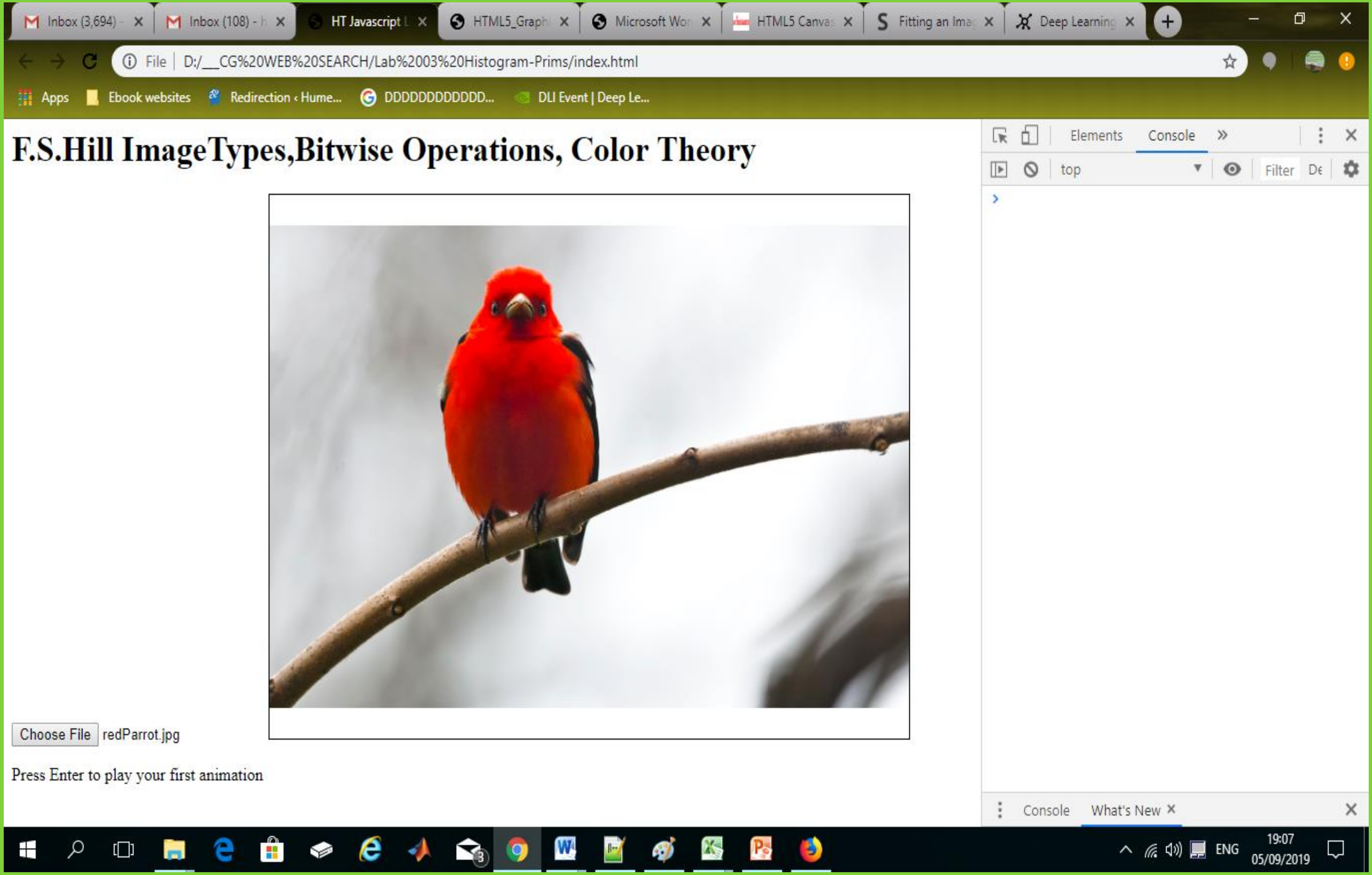


```
var fitImageOn = function(canvas, imageObj,context) {
    var imageAspectRatio = imageObj.width / imageObj.height;
    var canvasAspectRatio = canvas.width / canvas.height;
    var renderableHeight, renderableWidth, xStart, yStart;

    // If image's aspect ratio is less than canvas's we fit on height
    // and place the image centrally along width
    if(imageAspectRatio < canvasAspectRatio) {
        renderableHeight = canvas.height;
        renderableWidth = imageObj.width * (renderableHeight / imageObj.height);
        xStart = (canvas.width - renderableWidth) / 2;
        yStart = 0;
    }

    // If image's aspect ratio is greater than canvas's we fit on width
    // and place the image centrally along height
    else if(imageAspectRatio > canvasAspectRatio) {
        renderableWidth = canvas.width
        renderableHeight = imageObj.height * (renderableWidth / imageObj.width);
        xStart = 0;
        yStart = (canvas.height - renderableHeight) / 2;
    }

    // Happy path - keep aspect ratio
    else {
        renderableHeight = canvas.height;
        renderableWidth = canvas.width;
        xStart = 0;
        yStart = 0;
    }
    context.drawImage(imageObj, xStart, yStart, renderableWidth, renderableHeight);
};
```

Chap 3
Case I:
R > w/h

Chap 3
Case II:
R < w/h

Problem 2: Image Tiling


D:/__CG%20WEB%20SEARCH/Lab%2003%20Histogram-Prims/index.html

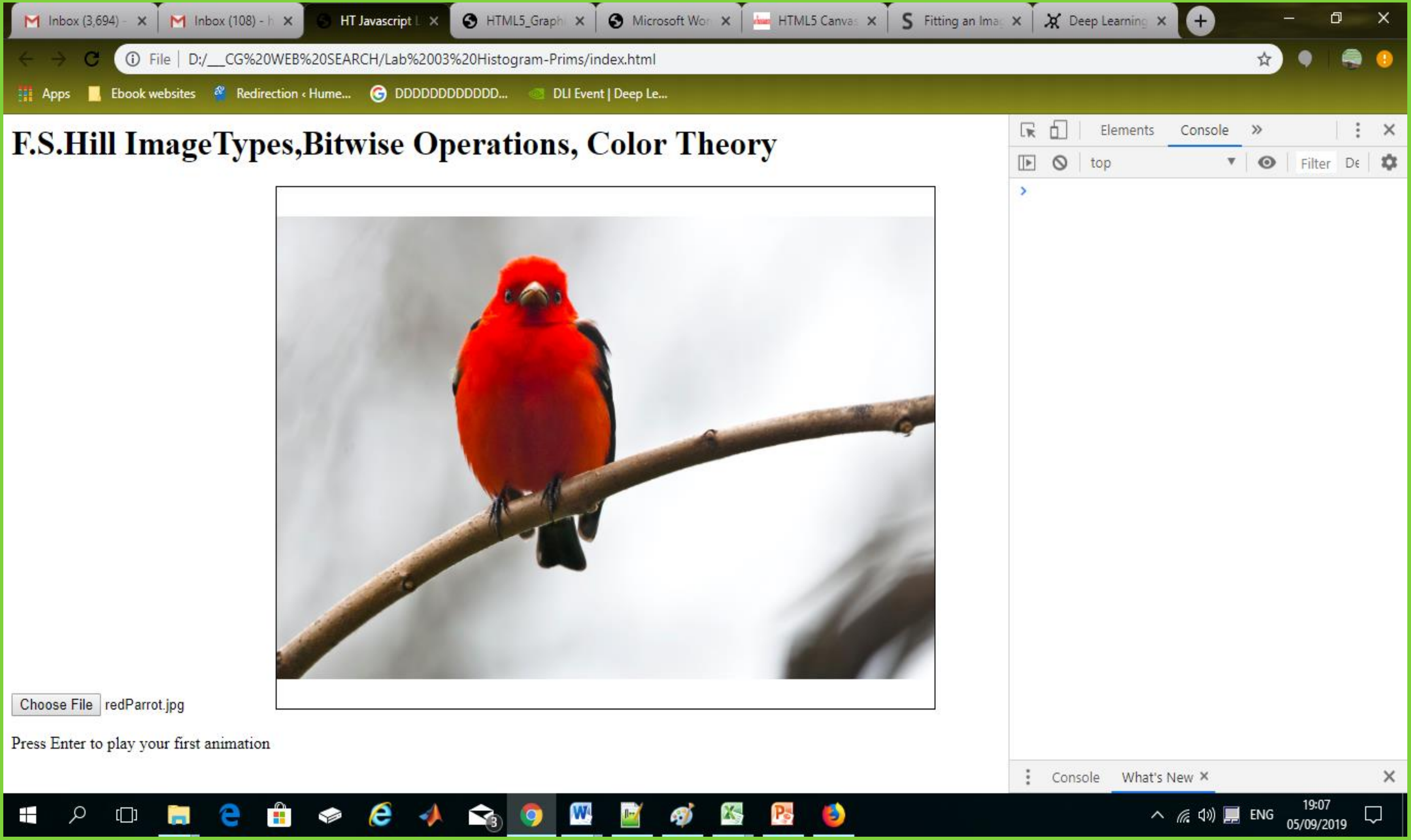
F.S.Hill ImageTypes,Bitwise Operations, Color Theory

Choose File redParrot.jpg

Press Enter to play your first animation

Image Histogram

1- Where I should set my viewport on canvas??



F.S.Hill ImageTypes,Bitwise Operations, Color Theory

Choose File redParrot.jpg

Press Enter to play your first animation

Viewport in Vanilla Java Script

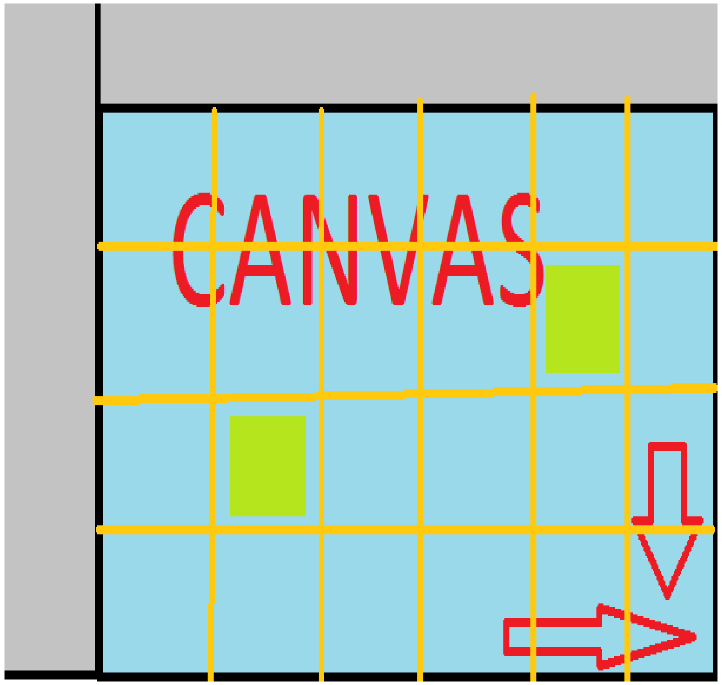
Note: viewport discussion is part of chap 3.

https://gomakethings.com/breakpoint-conditional-javascript-in-vanilla-js/

Note: The <canvas> element has no drawing abilities of its own (it is only a container for graphics) - you must use a script to actually draw the graphics.

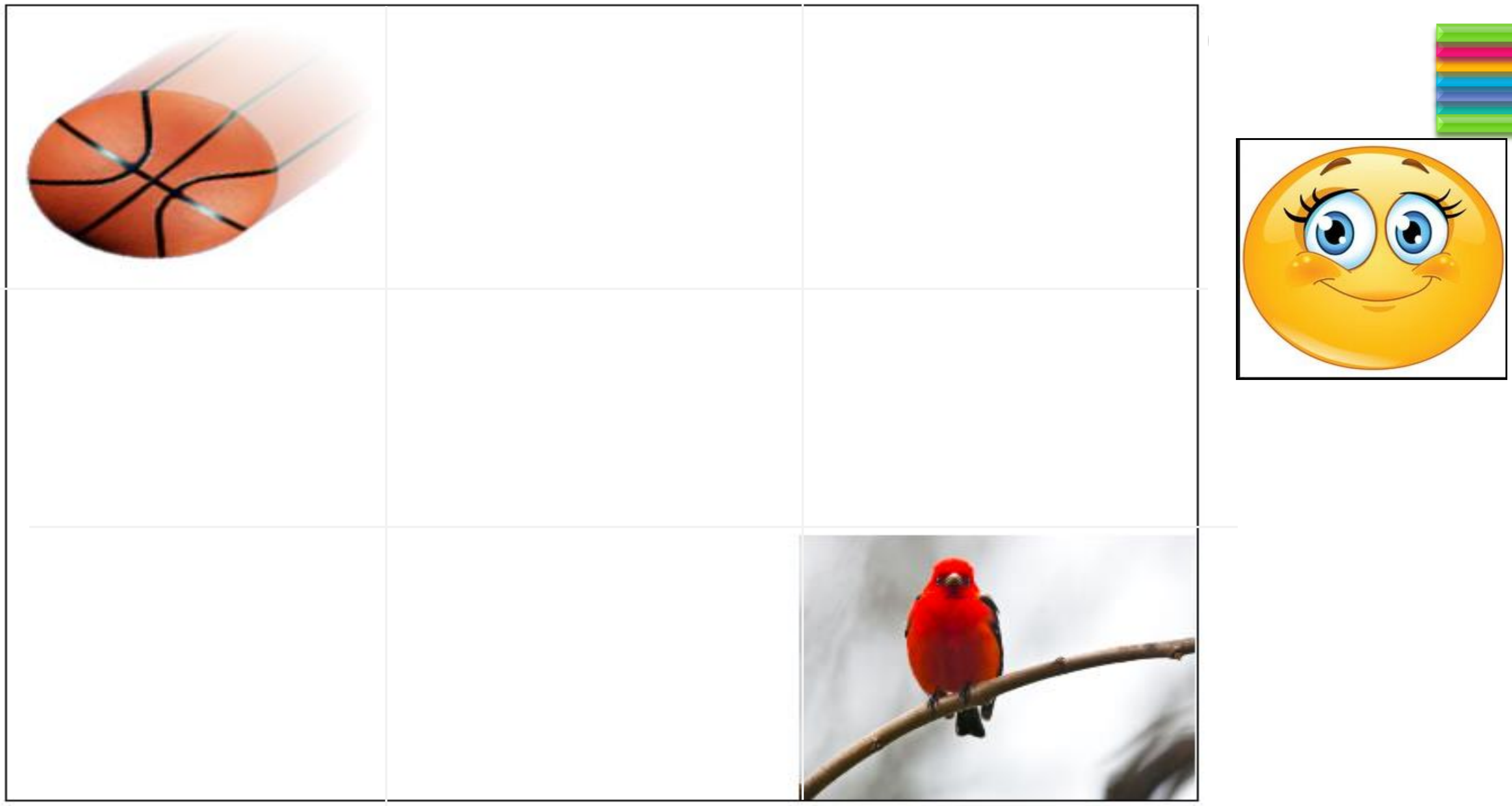
The getContext() method returns an object that provides methods and properties for drawing on the canvas.

This reference will cover the properties and methods of the getContext("2d") object, which can be used to draw text, lines, boxes, circles, and more - on the canvas.



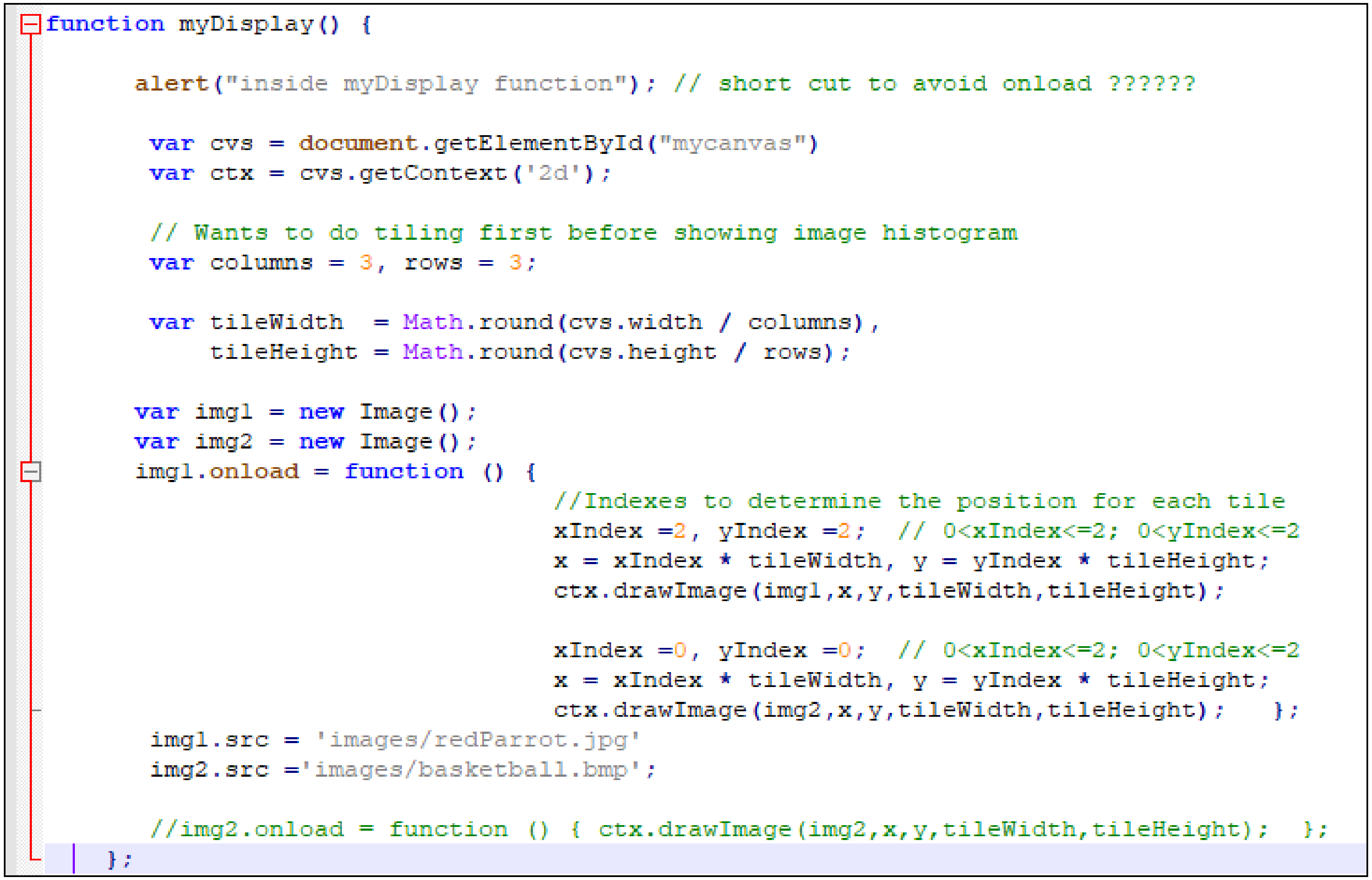
Tiling in Vanilla Java Script

Must read/study tiling loop from chapter 3 as we want to divide canvas into spaces of same width and height.



https://stackoverflow.com/questions/20711328/divide-a-canvas-into-spaces-of-same-width-and-height

Changes in code to achieve Tiling
Note: viewport discussion is part of chap 3.



```
function myDisplay() {

      alert("inside myDisplay function"); // short cut to avoid onload ??????

       var cvs = document.getElementById("mycanvas")
       var ctx = cvs.getContext('2d');

       // Wants to do tiling first before showing image histogram
       var columns = 3, rows = 3;

       var tileWidth  = Math.round(cvs.width / columns),
           tileHeight = Math.round(cvs.height / rows);

      var img1 = new Image();
      var img2 = new Image();
      img1.onload = function () {
                            //Indexes to determine the position for each tile
                            xIndex =2, yIndex =2;  // 0<xIndex<=2; 0<yIndex<=2
                            x = xIndex * tileWidth, y = yIndex * tileHeight;
                            ctx.drawImage(img1,x,y,tileWidth,tileHeight);

                            xIndex =0, yIndex =0;  // 0<xIndex<=2; 0<yIndex<=2
                            x = xIndex * tileWidth, y = yIndex * tileHeight;
                            ctx.drawImage(img2,x,y,tileWidth,tileHeight);   };
       img1.src = 'images/redParrot.jpg'
       img2.src ='images/basketball.bmp';

       //img2.onload = function () { ctx.drawImage(img2,x,y,tileWidth,tileHeight);  };
    };
```

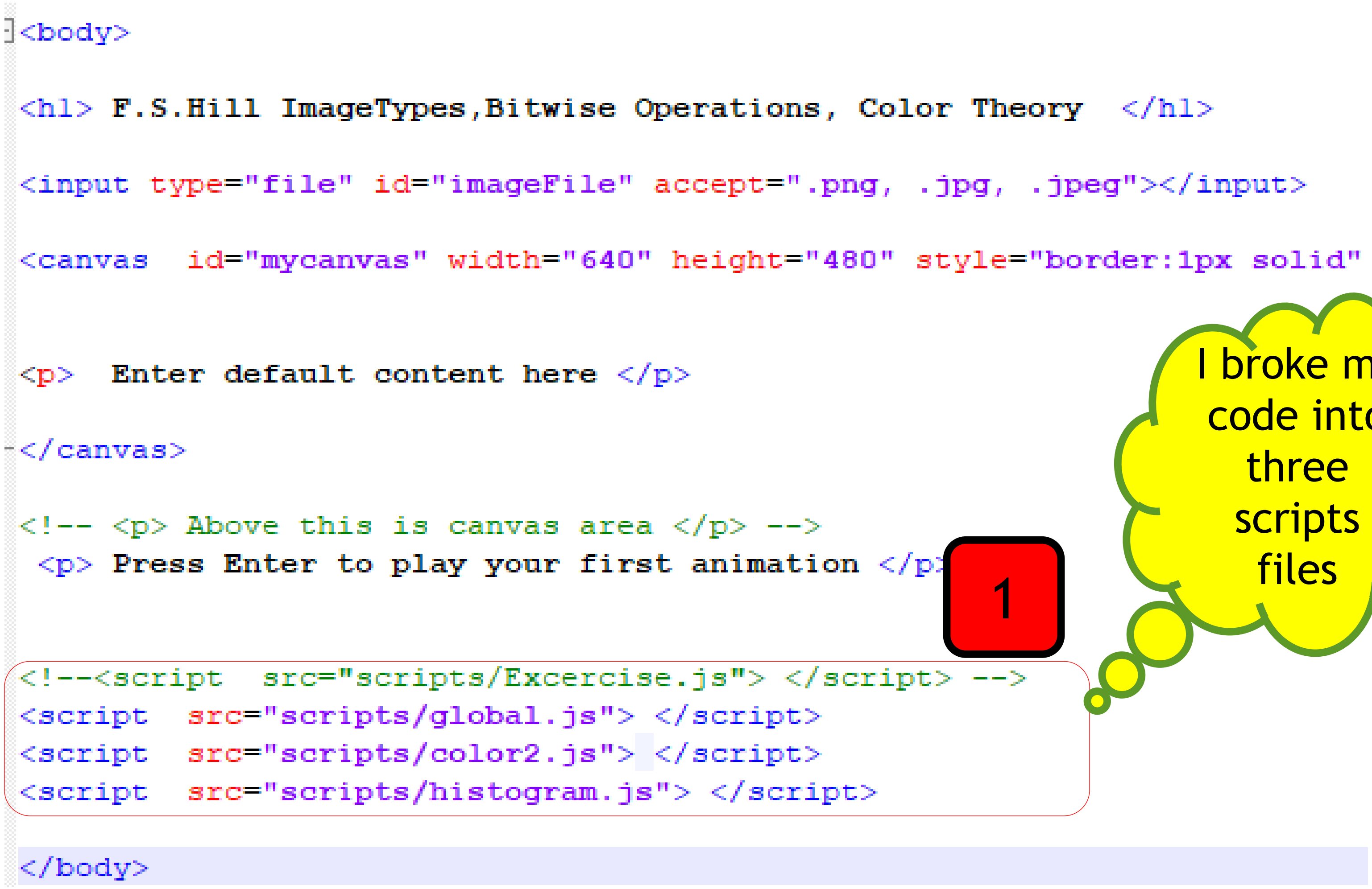
Problem 3: Passing argument between two .js files to reuse existing code

When you add a <script> element in a page, the code within that tag is executed by the browser and all variables you declare are added to the special “window” object. The window object is a global object and as such it can be referenced from every piece of code on that page. That is why and how a variable x will be there when you try to use it in the second JS file.

At this point I must warn you that this is an extremely bad idea.

It may strike you as a nice solution to a specific problem you have but in the long run it is going to cause you a lot of headaches. Your code will be extremely fragile (what happens if you switch order of the <script> tags?) and it will be a nightmare to maintain, especially after your code has reached a couple hundreds of lines.

Understanding global and shared in java script

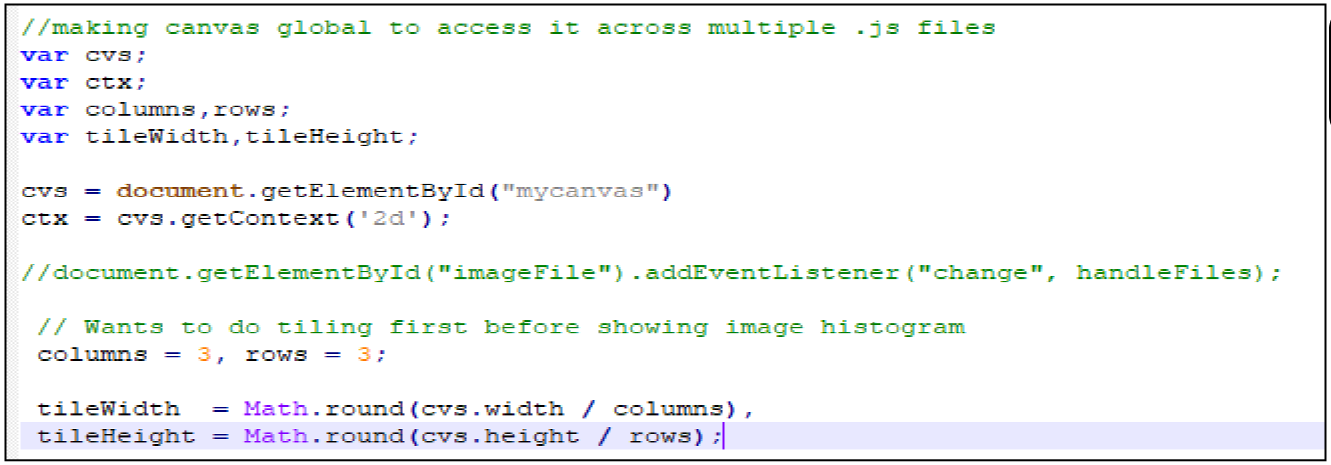


```
<body>

<h1> F.S.Hill ImageTypes,Bitwise Operations, Color Theory  </h1>

<input type="file" id="imageFile" accept=".png, .jpg, .jpeg"></input>

<canvas  id="mycanvas" width="640" height="480" style="border:1px solid"


<p>  Enter default content here </p>

</canvas>

<!-- <p> Above this is canvas area </p> -->
 <p> Press Enter to play your first animation </p>


<!--<script  src="scripts/Excercise.js"> </script> -->
<script  src="scripts/global.js"> </script>
<script  src="scripts/color2.js"> </script>
<script  src="scripts/histogram.js"> </script>

</body>
```

1

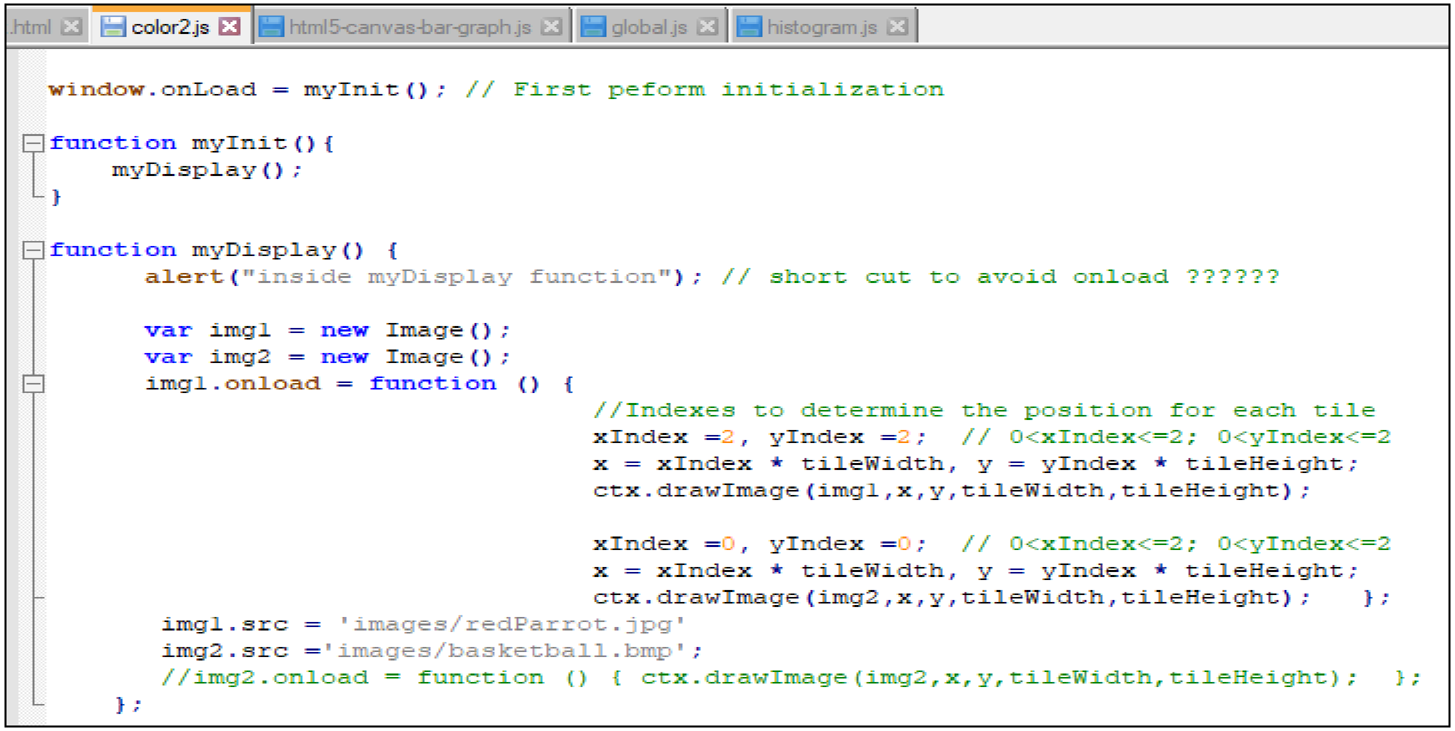
I broke my code into three scripts files

Scripts: global.js ; color2.js;



```
//making canvas global to access it across multiple .js files
var cvs;
var ctx;
var columns,rows;
var tileWidth,tileHeight;

cvs = document.getElementById("mycanvas")
ctx = cvs.getContext('2d');

//document.getElementById("imageFile").addEventListener("change", handleFiles);

 // Wants to do tiling first before showing image histogram
 columns = 3, rows = 3;

 tileWidth  = Math.round(cvs.width / columns),
 tileHeight = Math.round(cvs.height / rows);
```

2

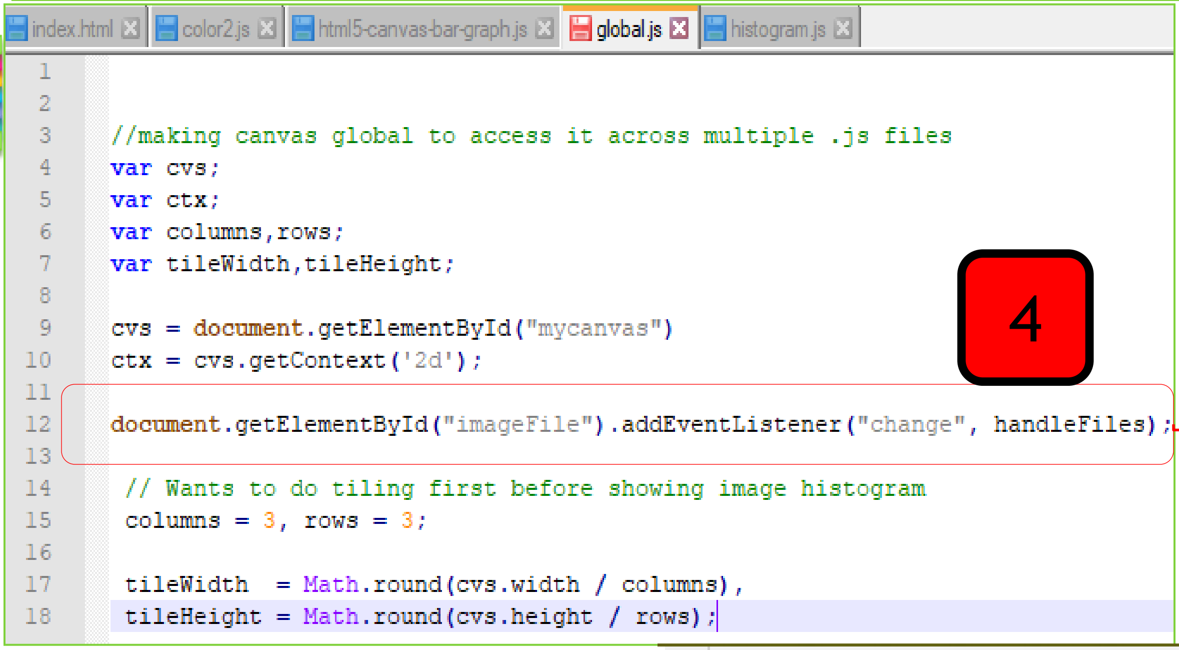


.html | color2.js | html5-canvas-bar-graph.js | global.js | histogram.js

```
window.onLoad = myInit(); // First peform initialization

function myInit(){
    myDisplay();
}

function myDisplay() {
      alert("inside myDisplay function"); // short cut to avoid onload ??????

      var img1 = new Image();
      var img2 = new Image();
      img1.onload = function () {
                              //Indexes to determine the position for each tile
                              xIndex =2, yIndex =2;  // 0<xIndex<=2; 0<yIndex<=2
                              x = xIndex * tileWidth, y = yIndex * tileHeight;
                              ctx.drawImage(img1,x,y,tileWidth,tileHeight);

                              xIndex =0, yIndex =0;  // 0<xIndex<=2; 0<yIndex<=2
                              x = xIndex * tileWidth, y = yIndex * tileHeight;
                              ctx.drawImage(img2,x,y,tileWidth,tileHeight);   };
       img1.src = 'images/redParrot.jpg'
       img2.src ='images/basketball.bmp';
       //img2.onload = function () { ctx.drawImage(img2,x,y,tileWidth,tileHeight);  };
   };
```

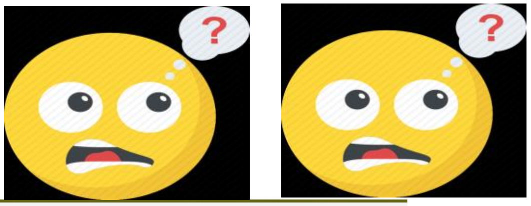
3

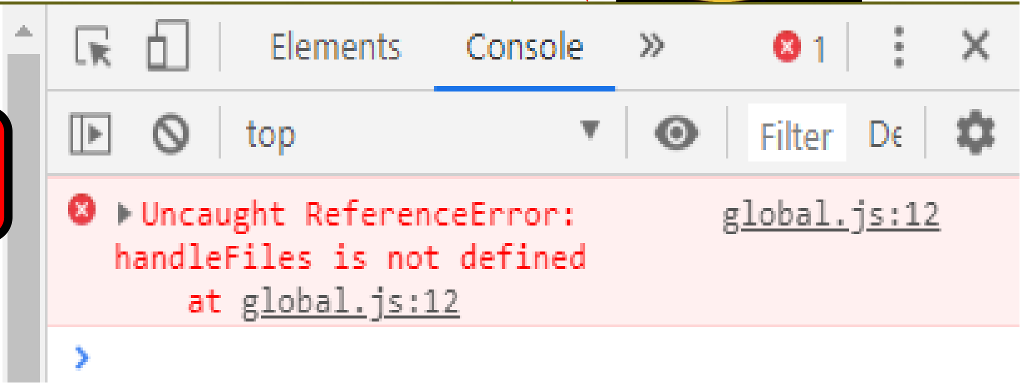
So far so good and my code is running fine and now I want to execute third script histogram.js on button and faced error



```
//making canvas global to access it across multiple .js files
var cvs;
var ctx;
var columns,rows;
var tileWidth,tileHeight;

cvs = document.getElementById("mycanvas")
ctx = cvs.getContext('2d');

document.getElementById("imageFile").addEventListener("change", handleFiles);

 // Wants to do tiling first before showing image histogram
 columns = 3, rows = 3;

 tileWidth  = Math.round(cvs.width / columns),
 tileHeight = Math.round(cvs.height / rows);
```

4





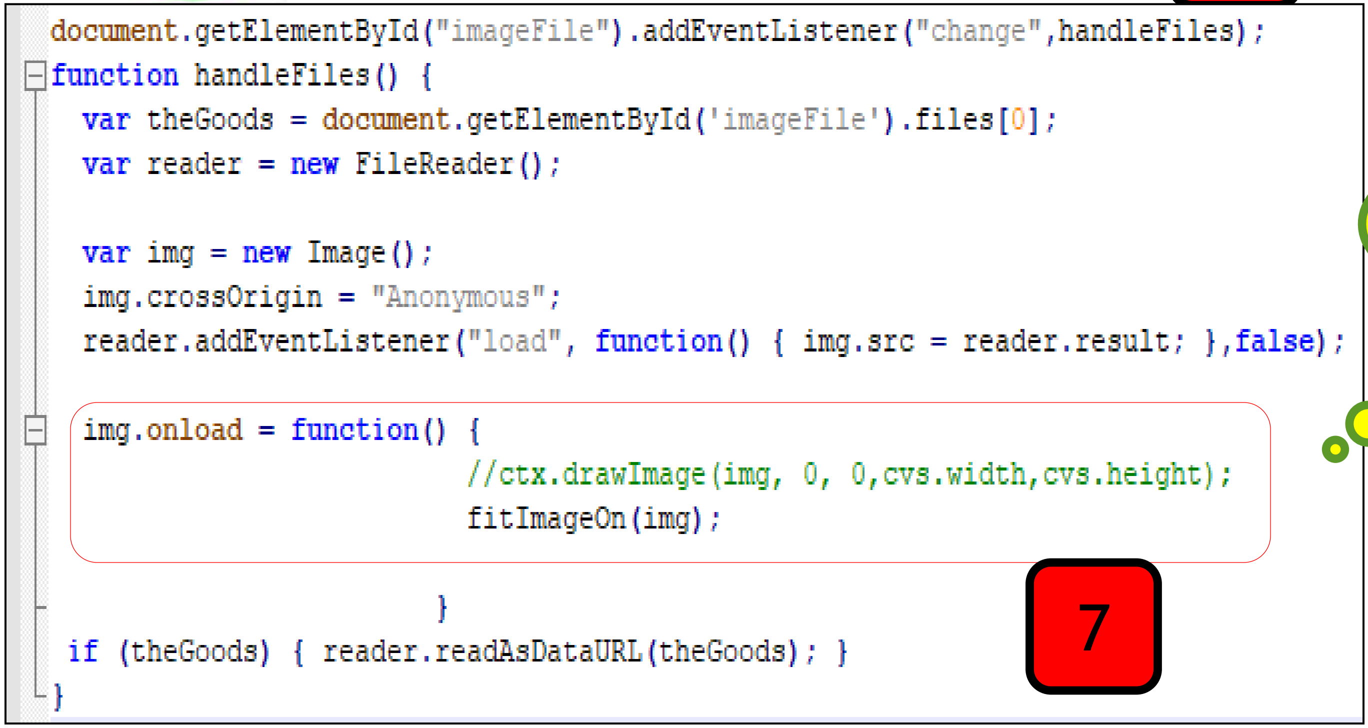
```
Uncaught ReferenceError:        global.js:12
handleFiles is not defined
    at global.js:12
```

5

I put both declaration and handleFiles() or body into histogram.js for time being.

Everything in JS is bound to containing scope. Therefore, if you define a function directly in file, it will be bound to window object, i.e. it will be global.

6



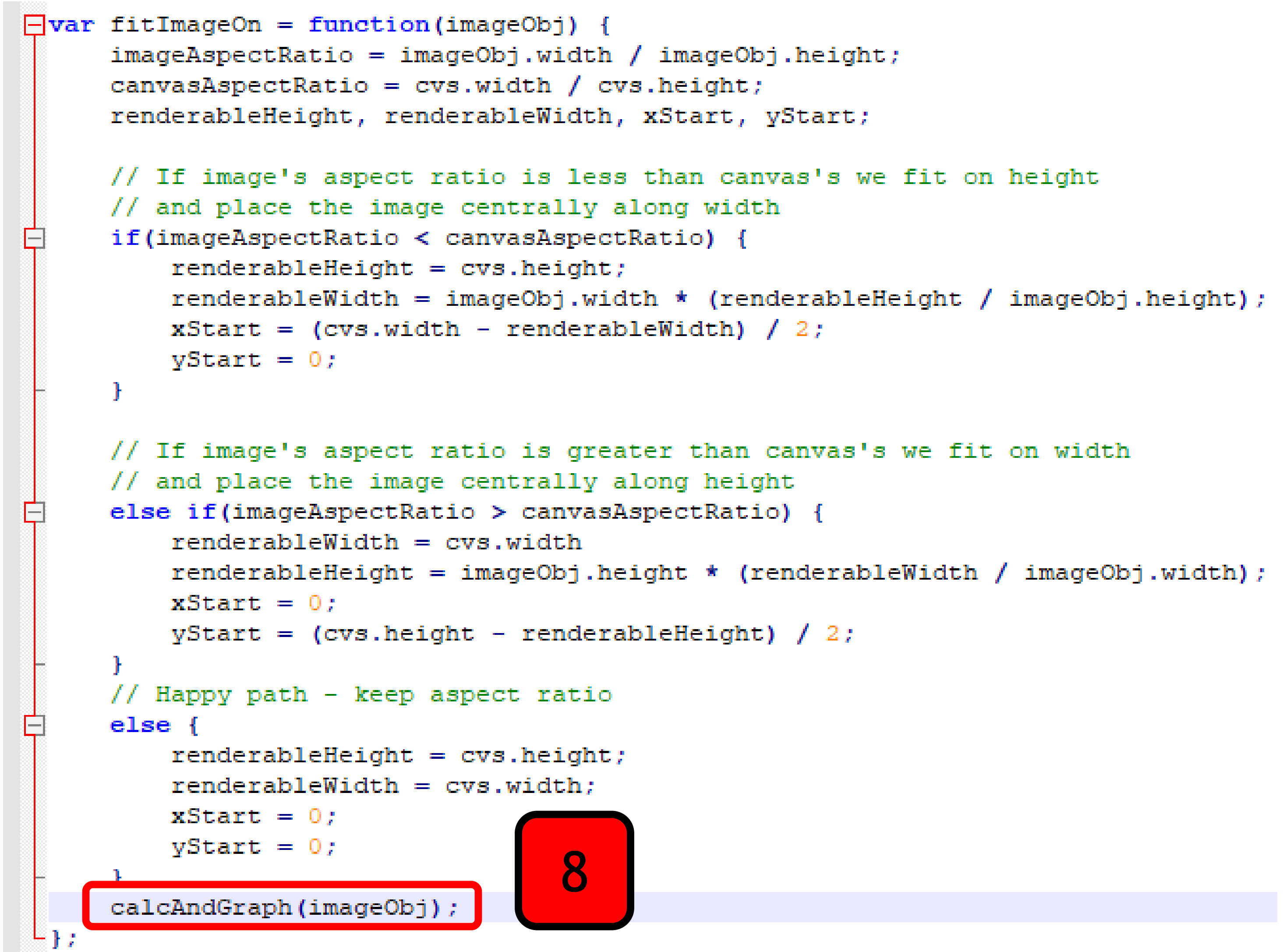
```
document.getElementById("imageFile").addEventListener("change",handleFiles);
function handleFiles() {
  var theGoods = document.getElementById('imageFile').files[0];
  var reader = new FileReader();

  var img = new Image();
  img.crossOrigin = "Anonymous";
  reader.addEventListener("load", function() { img.src = reader.result; },false);

  img.onload = function() {
                          //ctx.drawImage(img, 0, 0,cvs.width,cvs.height);
                          fitImageOn(img);

                       }
 if (theGoods) { reader.readAsDataURL(theGoods); }
}
```

7

Script histogram.js

Note we do not need to pass cvs and ctx as now they are global.

histogram.js → fitImageOn(….)

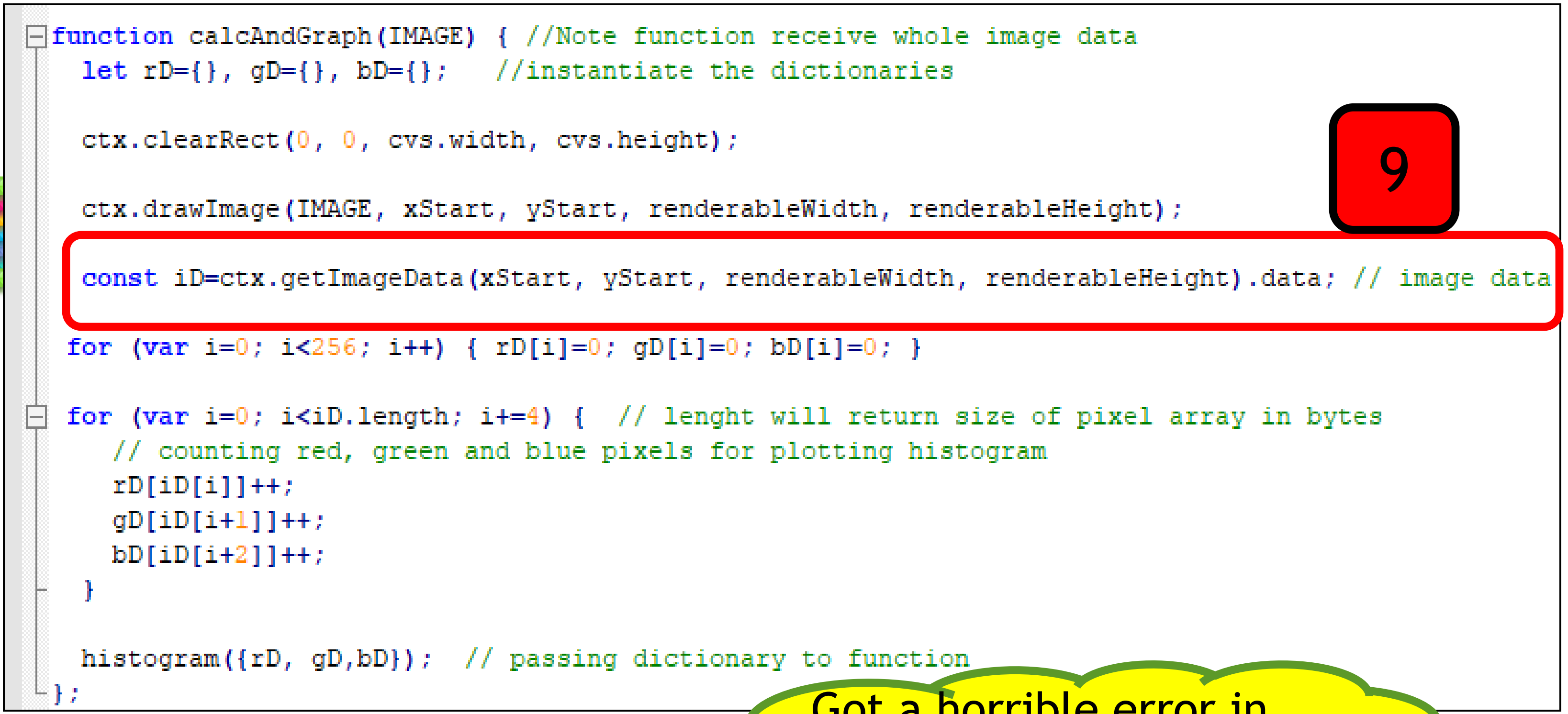


```
var fitImageOn = function(imageObj) {
    imageAspectRatio = imageObj.width / imageObj.height;
    canvasAspectRatio = cvs.width / cvs.height;
    renderableHeight, renderableWidth, xStart, yStart;

    // If image's aspect ratio is less than canvas's we fit on height
    // and place the image centrally along width
    if(imageAspectRatio < canvasAspectRatio) {
        renderableHeight = cvs.height;
        renderableWidth = imageObj.width * (renderableHeight / imageObj.height);
        xStart = (cvs.width - renderableWidth) / 2;
        yStart = 0;
    }

    // If image's aspect ratio is greater than canvas's we fit on width
    // and place the image centrally along height
    else if(imageAspectRatio > canvasAspectRatio) {
        renderableWidth = cvs.width
        renderableHeight = imageObj.height * (renderableWidth / imageObj.width);
        xStart = 0;
        yStart = (cvs.height - renderableHeight) / 2;
    }
    // Happy path - keep aspect ratio
    else {
        renderableHeight = cvs.height;
        renderableWidth = cvs.width;
        xStart = 0;
        yStart = 0;
    }
    calcAndGraph(imageObj);
};
```

8

Note we do not need to pass cvs and ctx as now they are global.

histogram.js → calAndGraph(….)



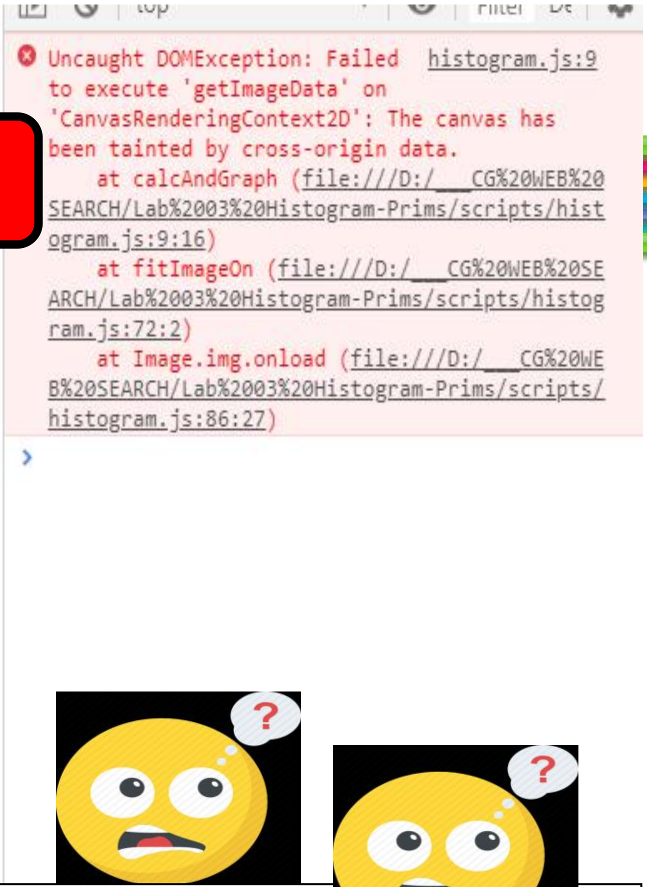
```
function calcAndGraph(IMAGE) { //Note function receive whole image data
  let rD={}, gD={}, bD={};   //instantiate the dictionaries

  ctx.clearRect(0, 0, cvs.width, cvs.height);

  ctx.drawImage(IMAGE, xStart, yStart, renderableWidth, renderableHeight);

  const iD=ctx.getImageData(xStart, yStart, renderableWidth, renderableHeight).data; // image data

 for (var i=0; i<256; i++) { rD[i]=0; gD[i]=0; bD[i]=0; }

 for (var i=0; i<iD.length; i+=4) {  // lenght will return size of pixel array in bytes
   // counting red, green and blue pixels for plotting histogram
   rD[iD[i]]++;
   gD[iD[i+1]]++;
   bD[iD[i+2]]++;
  }

  histogram({rD, gD,bD});  // passing dictionary to function
};
```

9

Got a horrible error in histogram.js (see next slide)

histogram.js → calAndGraph(….)

9



```
Uncaught DOMException: Failed   histogram.js:9
to execute 'getImageData' on
'CanvasRenderingContext2D': The canvas has
been tainted by cross-origin data.
    at calcAndGraph (file:///D:/__CG%20WEB%20
SEARCH/Lab%2003%20Histogram-Prims/scripts/hist
ogram.js:9:16)
    at fitImageOn (file:///D:/__CG%20WEB%20SE
ARCH/Lab%2003%20Histogram-Prims/scripts/histog
ram.js:72:2)
    at Image.img.onload (file:///D:/__CG%20WE
B%20SEARCH/Lab%2003%20Histogram-Prims/scripts/
histogram.js:86:27)
```

```
ctx.drawImage(IMAGE, xStart, yStart, renderableWidth, renderableHeight);

const iD=ctx.getImageData(xStart, yStart, renderableWidth, renderableHeight).data; // image data
```

Problem 4: The canvas has been tainted by cross origin data

Reason: Security Exception

Solution: We need to run client side script through server or http protocol.

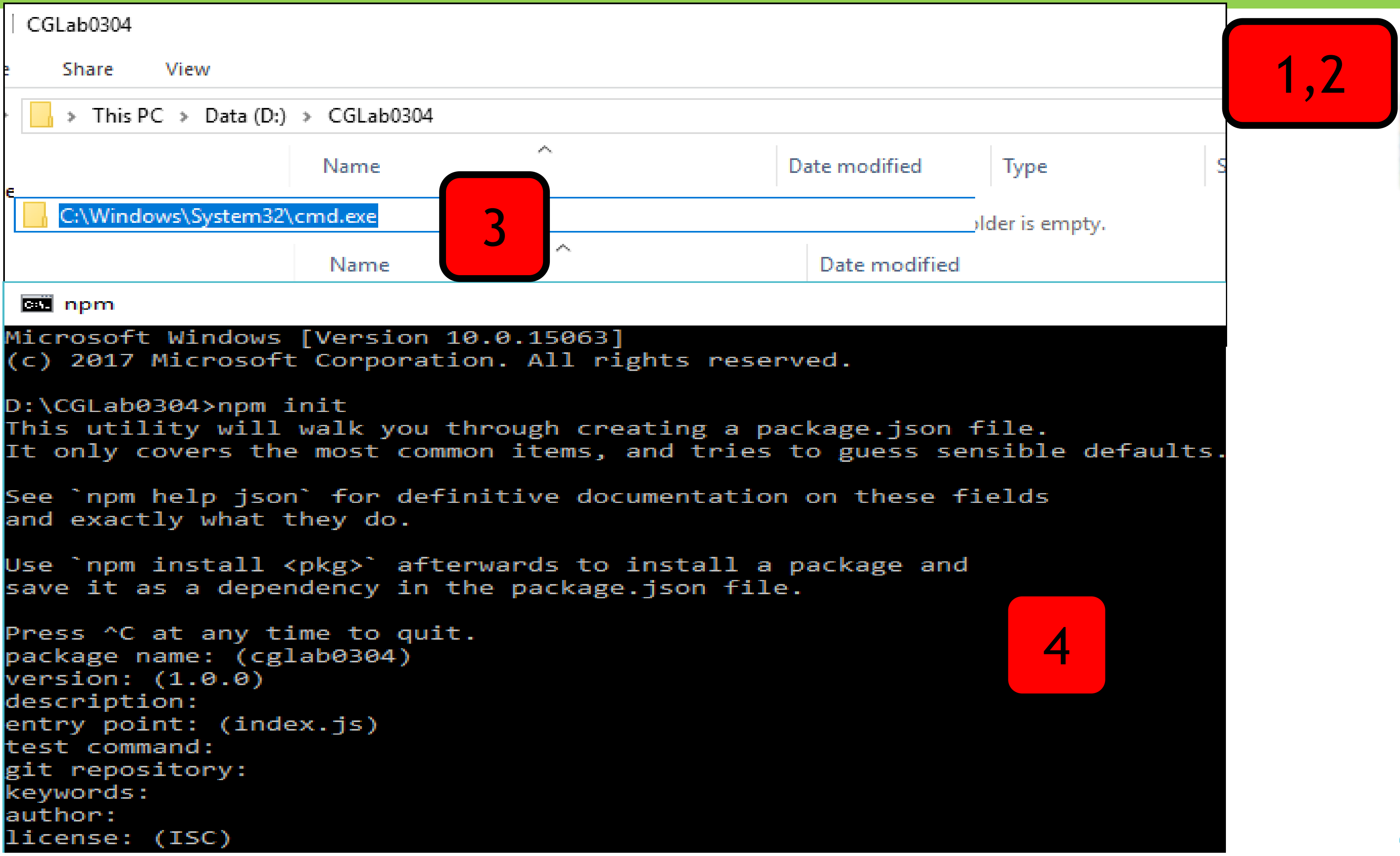
https://www.opencodez.com/java-script/static-website-with-node-js-webserver.htm

Solution: Running code through http protocol (step 1 till step 9)

1) Download node.js from web. Then install it
2) Create a directory for the code let say D:/Lab0304 (make sure directory name dosen't have space)
3) Now go to Lab0304 Directory
4) Run the command npm init then press enter until it will write .json file.
5) Then run the command npm install express --save
6) Create a folder named public in directory Lab01
7) Create three folders named (css, images, js) in public folder. Move the images and js files in corresponding directories
8) Create index.html file in public folder
9) Change the js files path in index.html to the following

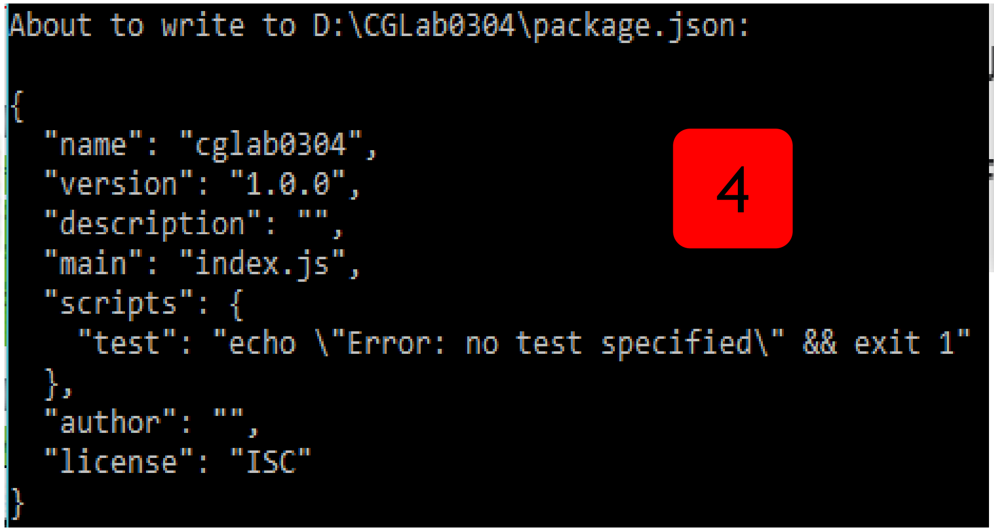
```
<script type="text/javascript"  src="/js/global.js"> </script>
<script type="text/javascript" src="/js/color2.js"> </script>
<script type="text/javascript" src="/js/histogram.js"> </script>
```

Output of step 1 till step 3

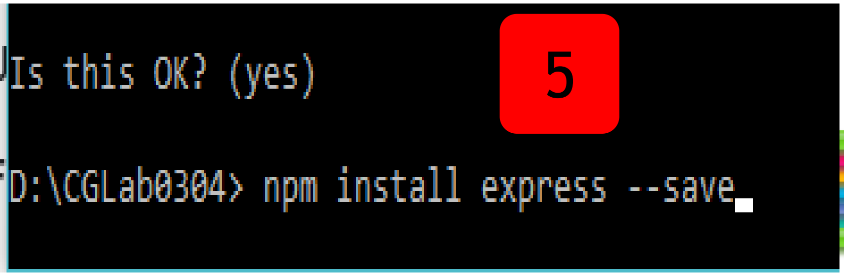


```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

D:\CGLab0304>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cglab0304)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
```

Output of step 4 till step 6

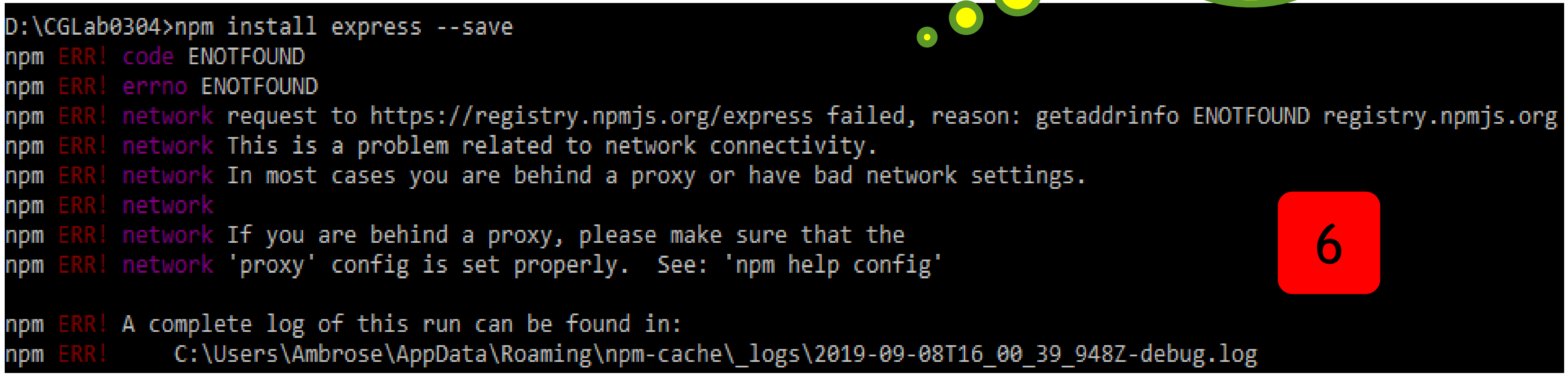


```
About to write to D:\CGLab0304\package.json:

{
  "name": "cglab0304",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```



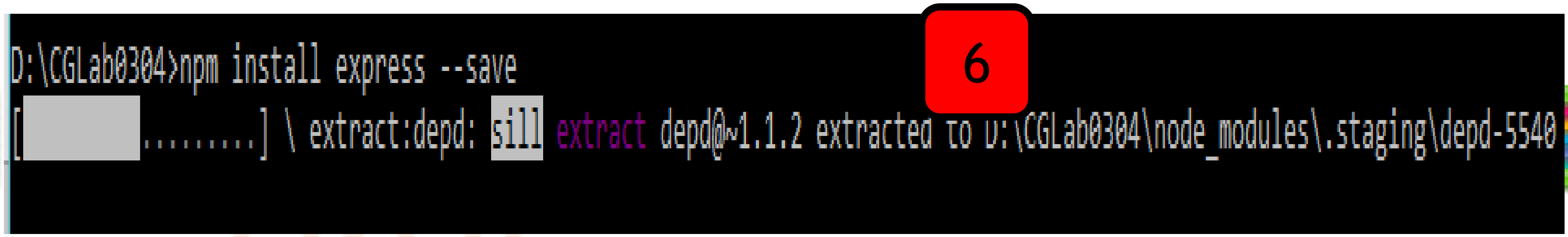
```
Is this OK? (yes)

D:\CGLab0304> npm install express --save
```

If internet connectivity is missing, error will be observed

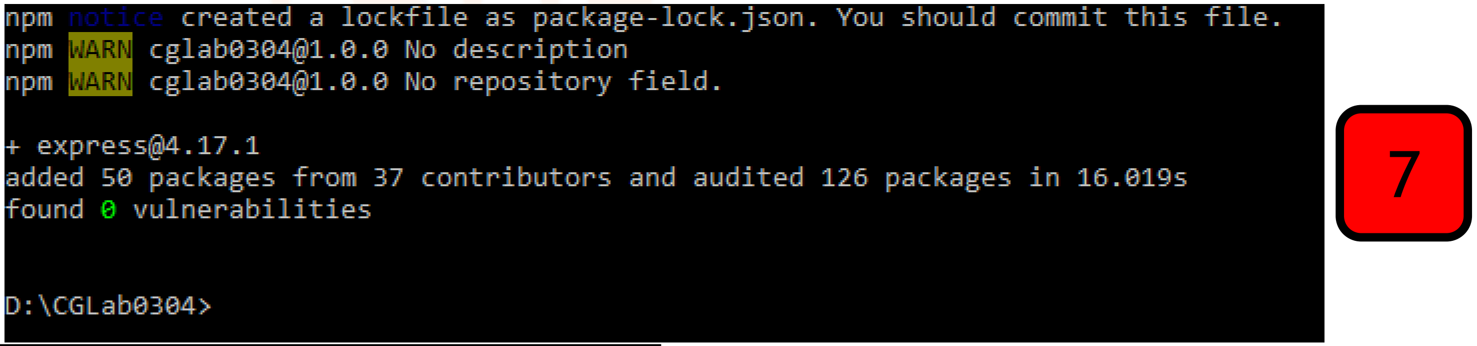


```
D:\CGLab0304>npm install express --save
npm ERR! code ENOTFOUND
npm ERR! errno ENOTFOUND
npm ERR! network request to https://registry.npmjs.org/express failed, reason: getaddrinfo ENOTFOUND registry.npmjs.org
npm ERR! network This is a problem related to network connectivity.
npm ERR! network In most cases you are behind a proxy or have bad network settings.
npm ERR! network
npm ERR! network If you are behind a proxy, please make sure that the
npm ERR! network 'proxy' config is set properly.  See: 'npm help config'

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\Ambrose\AppData\Roaming\npm-cache\_logs\2019-09-08T16_00_39_948Z-debug.log
```

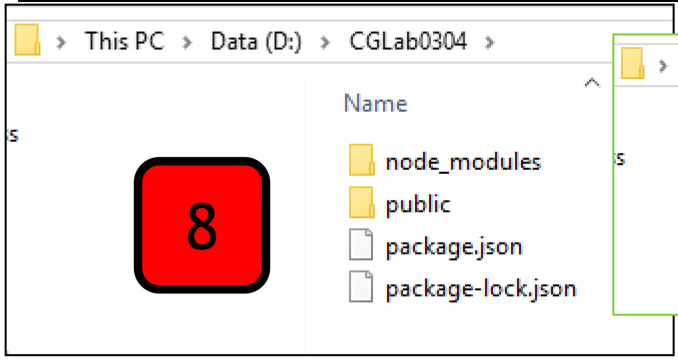
Output of step 6 till step 9


6

```
D:\CGLab0304>npm install express --save
[         .........] \ extract:depd: sill extract depd@~1.1.2 extracted to D:\CGLab0304\node_modules\.staging\depd-5540
```

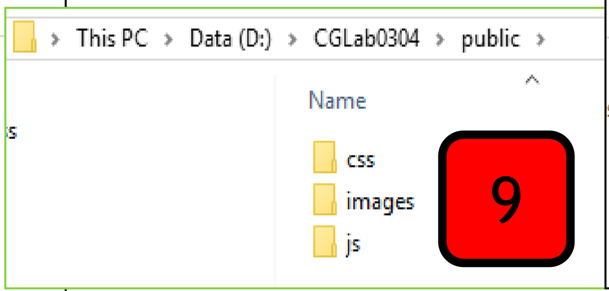

7

```
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN cglab0304@1.0.0 No description
npm WARN cglab0304@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 126 packages in 16.019s
found 0 vulnerabilities


D:\CGLab0304>
```


8

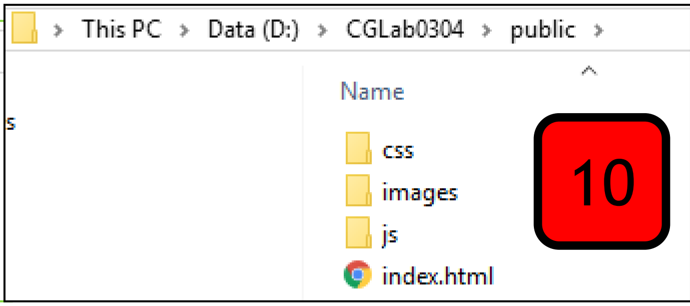
This PC › Data (D:) › CGLab0304 ›

Name
- node_modules
- public
- package.json
- package-lock.json


9

This PC › Data (D:) › CGLab0304 › public ›

Name
- css
- images
- js


10

This PC › Data (D:) › CGLab0304 › public ›

Name
- css
- images
- js
- index.html

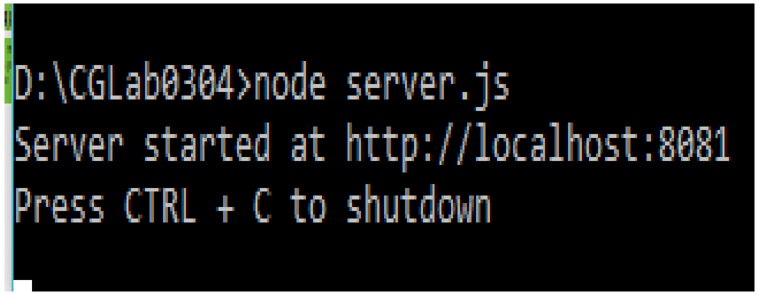
Problem 4: Running code through http protocol (Step 9 and 10)

9) Create server.js file in Lab01 directory
10) Place the following in server.js

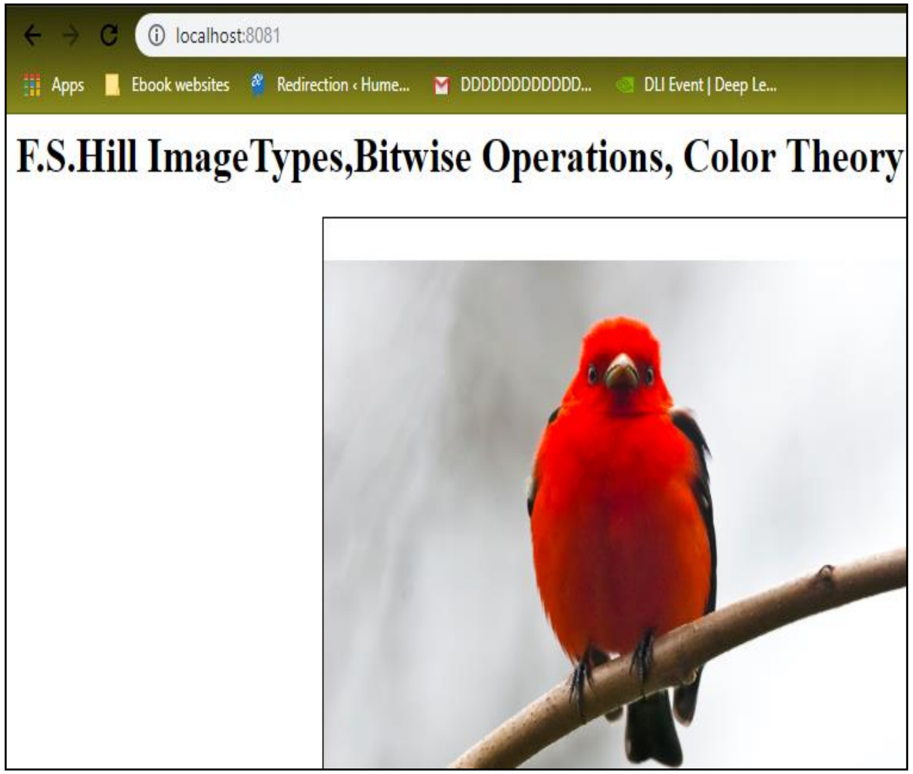
```
var express = require("express");
var app = express();
app.use(express.static('public'));

//make way for some custom css, js and images
app.use('/css', express.static(__dirname + '/public/css'));
app.use('/js', express.static(__dirname + '/public/js'));
app.use('/images', express.static(__dirname + '/public/images'));

var server = app.listen(8081, function(){
 var port = server.address().port;
console.log("Server started at http://localhost:%s \nPress CTRL + C to
                    shutdown", port);
});
```

Problem 4: Running code through http protocol (step 11-14)

11) Now open the terminal. Go to the D://Lab0304 Directory.
12) Run the command node server.js
13) Open the browser and type the URL http://localhost:8081.
14) App will open

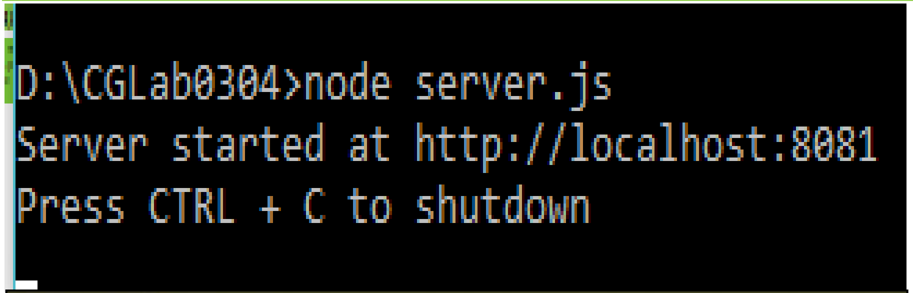


```
D:\CGLab0304>node server.js
Server started at http://localhost:8081
Press CTRL + C to shutdown
```

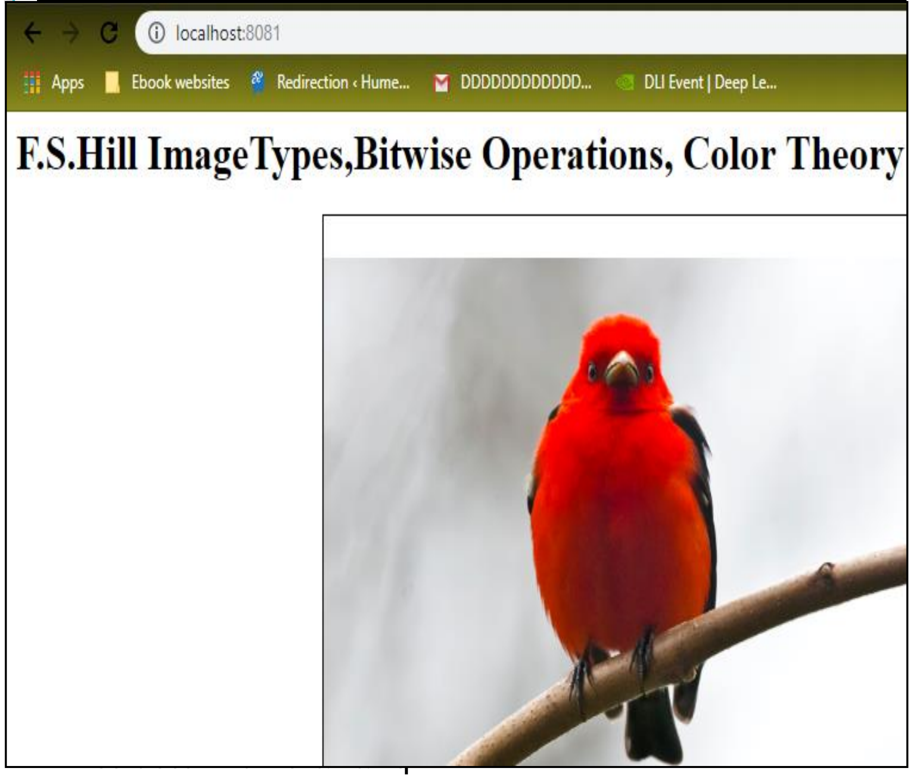


F.S.Hill ImageTypes,Bitwise Operations, Color Theory

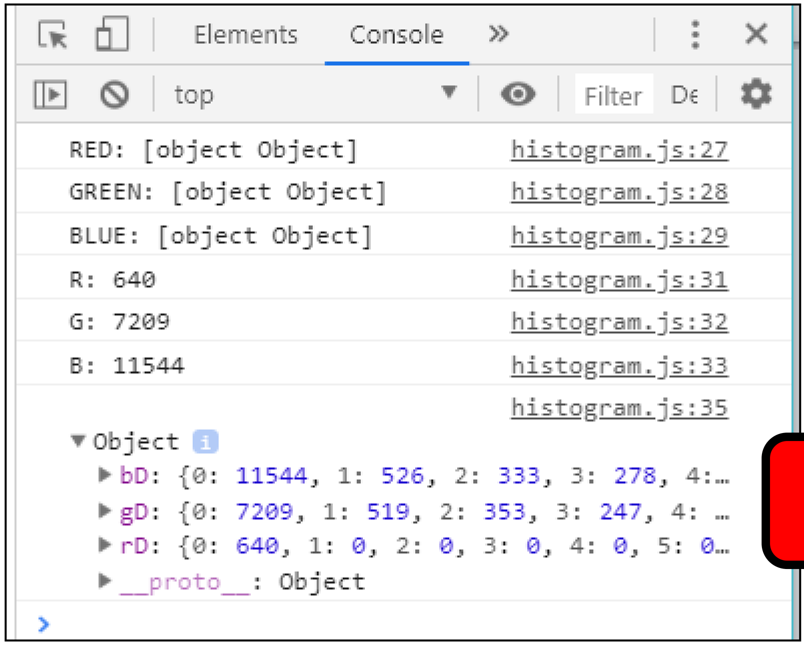
Now getImageData(…) is successful



```
D:\CGLab0304>node server.js
Server started at http://localhost:8081
Press CTRL + C to shutdown
```



F.S.Hill ImageTypes,Bitwise Operations, Color Theory



```
RED: [object Object]                 histogram.js:27
GREEN: [object Object]               histogram.js:28
BLUE: [object Object]                histogram.js:29
R: 640                               histogram.js:31
G: 7209                              histogram.js:32
B: 11544                             histogram.js:33
                                     histogram.js:35
▼Object
  ▶bD: {0: 11544, 1: 526, 2: 333, 3: 278, 4:…
  ▶gD: {0: 7209, 1: 519, 2: 353, 3: 247, 4: …
  ▶rD: {0: 640, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0…
  ▶__proto__: Object
>
```

1

The **getImageData**() function returns retrieve a set of pixel **data** from the canvas. The ImageData object represents a rectangle area of information and holds every pixel inside that rectangle. Every pixel in an ImageData object has four-element-array-like value, the RGBA values.

Next Lab 05-Lab 06

1. Pixel Counting for histogram(Chap 10)
2. Drawing X and Y axis using moveTo() lineTo() (Chap 3)
3. Working with classes in pure java script

   class Point, class Square
4. Tweening / In-Between/ lerp (Chap 4)
5. Finally we will draw histogram of Image (Chap 10)