

Assignment: [Tutorial] Web Workers

Due Date: June 17th by 11:59pm

Points: 10

Description: In this tutorial we will use web workers to apply filters to images. You should have an *image-filtering* directory with some starter files. Specifically, the *index.html*, which is pre-loaded with several images from Pixabay.com.

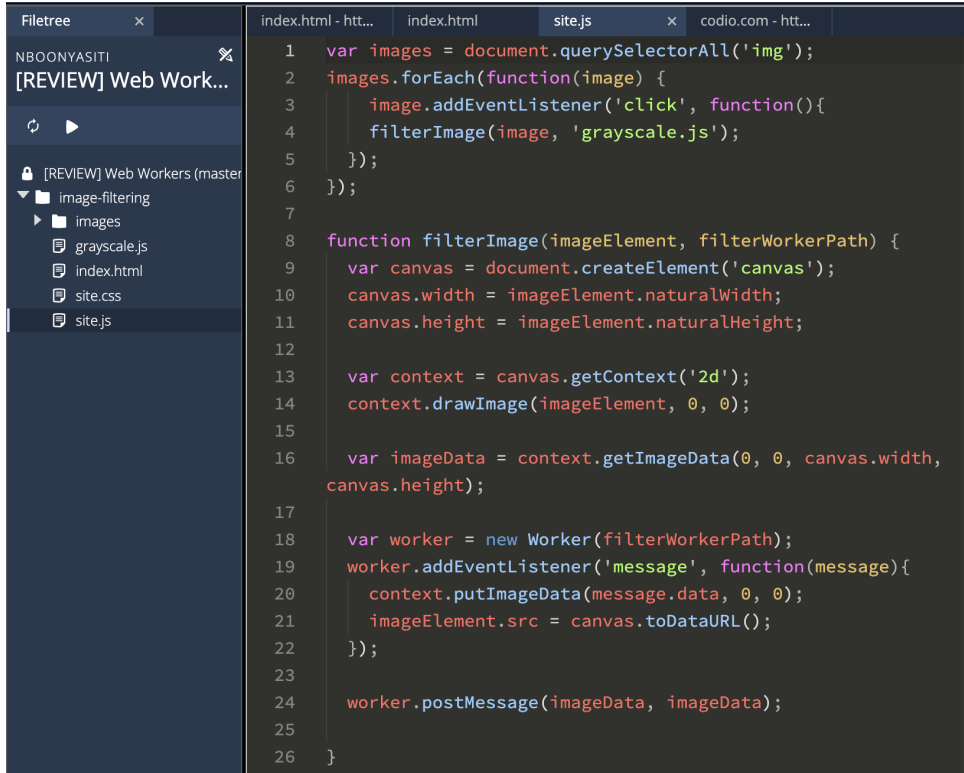
Overall Summary:

The screenshot shows a web browser interface with a dark sidebar on the left and a main content area. The sidebar contains a file tree for a project named '[REVIEW] Web Work...'. The main content area displays the assessment results for 'Web Development / Week 2'. The assessment is titled '[REVIEW] Web...' and is attributed to 'Nina Boonyasiti'. The results show a 'Grade' of 100, 'Points (%)' of 100, 'Graded' status of 1, and 'Answered Assessments' of 0. Below this, there is a table with columns for 'Section', 'Assessment Name', 'Points', 'Type', and 'Correct'. The table contains one entry for 'Summary' with the 'Assessment Name' 'Instructor Review', 'Points' '10/10', 'Type' 'Instructor Review', and 'Correct' status 'Correct'. The date of the assessment is 'Jun 22nd 2022 1:22pm (UTC -05:00) America/Chicago'. The 'Grade details' section is also visible.

Section	Assessment Name	Points	Type	Correct
Summary	Instructor Review	10/10	Instructor Review	Correct

Student Submission:

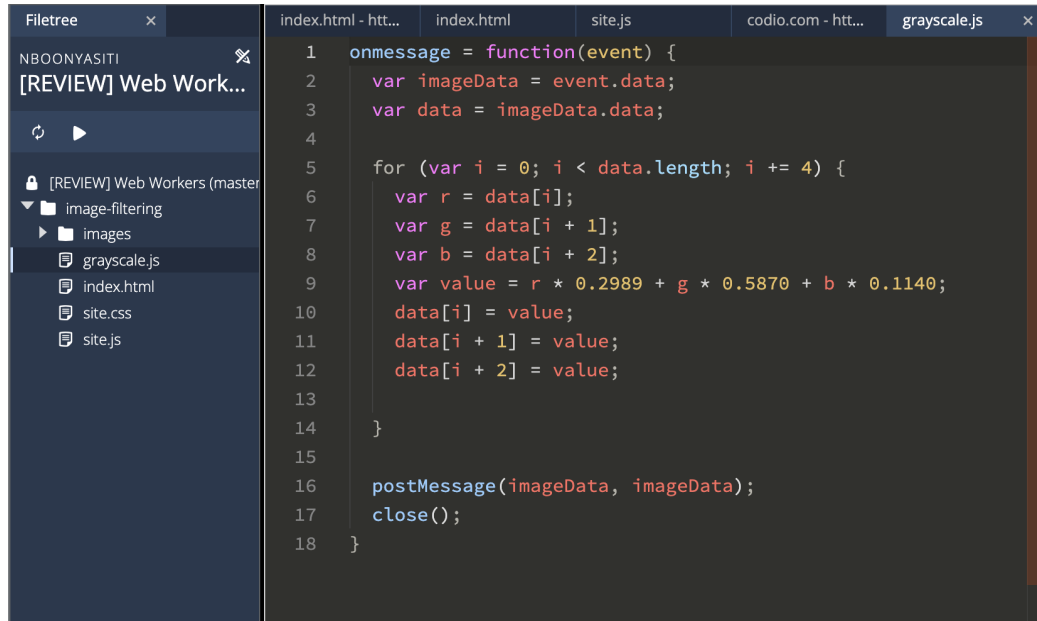
We first create our filterImage() function to create a “canvas” element and capture the data of the image and save it to our canvas. With this information, we will have a basis of what images we want to select to undergo filtering. Then we have a looping condition to listen for a click event and upon that click event, the image filter is triggered via grayscale.js.



The screenshot shows a web development IDE with a dark theme. On the left, a file tree is visible under the name 'NBOONYASITI'. It contains a folder '[REVIEW] Web Work...' and a sub-folder 'image-filtering'. Inside 'image-filtering', there is a sub-folder 'images' containing 'grayscale.js', 'index.html', 'site.css', and 'site.js'. The main editor area on the right displays the code for 'site.js'. The code is as follows:

```
1  var images = document.querySelectorAll('img');
2  images.forEach(function(image) {
3      image.addEventListener('click', function(){
4          filterImage(image, 'grayscale.js');
5      });
6  });
7
8  function filterImage(imageElement, filterWorkerPath) {
9      var canvas = document.createElement('canvas');
10     canvas.width = imageElement.naturalWidth;
11     canvas.height = imageElement.naturalHeight;
12
13     var context = canvas.getContext('2d');
14     context.drawImage(imageElement, 0, 0);
15
16     var imageData = context.getImageData(0, 0, canvas.width,
17     canvas.height);
18
19     var worker = new Worker(filterWorkerPath);
20     worker.addEventListener('message', function(message){
21         context.putImageData(message.data, 0, 0);
22         imageElement.src = canvas.toDataURL();
23     });
24     worker.postMessage(imageData, imageData);
25
26 }
```

In grayscale.js, we are taking in the image data passed through site.js and iterating each pixel through a for loop. In this loop we want to save each pixel as rgb and change their hexadecimal values to reflect that of a grayscale filter. We save these new values and together they will create a new gray image. After the loop has completed execution, the new image data will be shared back to site.js for index.html to display. A demo of this tutorial is provided in the Week 2 materials.



The image shows a web-based code editor interface. On the left is a file tree with a dark blue background. It shows a project named "[REVIEW] Web Workers (master)". Inside, there is a folder "image-filtering" which contains a subfolder "images". The "images" folder is expanded, showing three files: "grayscale.js" (selected), "index.html", "site.css", and "site.js". The main area on the right has a dark gray background and displays the code for "grayscale.js". The code is a JavaScript function that processes image data into grayscale. It starts with an "onmessage" event listener. It receives "event.data" and assigns it to "imageData" and "data". Then, it loops through "data" in increments of 4. For each iteration, it extracts the red, green, and blue components (r, g, b) and calculates a grayscale value using the formula: $value = r * 0.2989 + g * 0.5870 + b * 0.1140$. It then assigns this value to the corresponding indices in the "data" array. Finally, it posts the processed "imageData" back to the parent script and closes the worker.

```
1 onmessage = function(event) {
2   var imageData = event.data;
3   var data = imageData.data;
4
5   for (var i = 0; i < data.length; i += 4) {
6     var r = data[i];
7     var g = data[i + 1];
8     var b = data[i + 2];
9     var value = r * 0.2989 + g * 0.5870 + b * 0.1140;
10    data[i] = value;
11    data[i + 1] = value;
12    data[i + 2] = value;
13  }
14
15
16  postMessage(imageData, imageData);
17  close();
18 }
```