Try It! Green Screen Web Page Exercises

Duke University
Programming Foundations with JavaScript, HTML and CSS
Module 4, Lesson 2: Green Screen Web Page

Try It! Upload and Display an Image

Create the prototype web page

1. Create a new pen, and add the following HTML elements: 1 header, 1 canvas (with an ID), 1 text type input (with an ID), 1 button type input (with *value* and *onclick* attributes—the onclick event handler should call an **upload()** function you will write later)

2. Practice CSS by adding formatting commands such as: (a) a font family or color for the header and body; (b) a margin for the web page; (c) dimensions and a border for the canvas; (d) a font size for the input elements

3. Add the JavaScript function *upload()* in the JS panel. This function should:

- **Create a variable that gets the value of the text from the text input element**, and
- **Display this text in an alert**.

Need help? Review the **Upload and Display an Image** video and see the following screenshot example.

HTML

```
<canvas id="can">
</canvas>

<p>
  Filename:
  <input type="text" id="finput" >
  <input type="button" value="Upload"
onclick="upload()" >
</p>
```

CSS

```
canvas {
  height: 200px;
  border: 1px solid lightgray;
}

input {
  font-size: 14pt;
}
```

JS

```
function upload() {
  //Get input from text input
  var fileinput =
document.getElementById("finput");
  var filename = fileinput.value;
  //Alert displaying text
  alert("You chose " + filename);
}
```

Adapt the page to upload an image

1. Delete the text and button inputs.

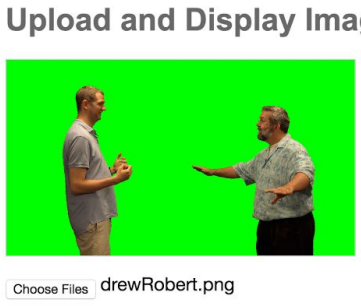
2. Create a file input element. Make sure the file input:

- Accepts only single files
- Accepts only image files
- Has an ID

3. Adapt the *upload()* function to use the file input to display an image. Your function should also:

- **Get the file input**.
- **Create a SimpleImage from the chosen file**. The SimpleImage library can be found at http://www.dukelearntoprogram.com/course1/common/js/image/SimpleImage.js. Remember you will need to use the **<script src=' '></script>** tags in the HTML pane to tell your web page where to find the Simple Image library.
- **Get the canvas element, and draw the image on the canvas**. Note that you can define only one of the width or height of the canvas to avoid changing the image aspect ratio.

Need help? Review the **Upload and Display an Image** video and see the following screenshot example.



Try It! Convert an Image to Grayscale

Develop grayscale algorithm and code at DukeLearnToProgram

1. Make sure you have used the grayscale algorithm to work several examples by hand.

2. Write out the steps of your solution by hand (e.g., loading the image, iterating over each pixel, etc.).

3. Implement the algorithm in the DukeLearnToProgram (DLTP) environment (http://www.dukelearntoprogram.com/course1/example/index.php, also linked in the **Resources** tab). Make sure you have a working copy that converts an image to grayscale. It is a good idea to practice any new implementation of a problem in this environment because it has features to help you debug.

Need help? Review the **Convert Image to Grayscale** video and study the algorithm we walk through.

Move code to your web page

1. Fork your previous Pen to start with a web page that allows a user to upload an image, which it will then display.

2. In the HTML panel, add a button that calls the function **makeGray()** when clicked.

3. Add the function *makeGray()* to the JS panel. Now you will take the code you wrote in the DLTP environment and adapt it to the web page. Because you are containing the grayscale process in the function **makeGray()**, there are some changes to make.

- The grayscale image should be a global variable to allow the **upload()** and **makeGray()** functions to both access it.
- The function **upload()** should set the global image variable to the file input.
- The function **makeGray()** should draw the image to the canvas instead of printing it.
- Remember that we are able to use the SimpleImage and SimplePixel libraries because we are importing these libraries in the HTML panel using **script** tags.

Need help? Review the code we walk through in the **Convert Image to Grayscale** video; and seek help in the forums!

Refactor your code

Let's restructure and improve on our web page's code so that we have two canvases: one to show the original image and another to show the grayscale version of the image.

1. Add another canvas to the right of the existing one.

2. In the JS panel, declare two global variables, one for the original image, and one for the grayscale image.

3. Modify the function *upload()*. When we upload our image file, we want to use it to create two new SimpleImages, one that gets drawn to our canvas on the left and another that we will eventually make gray scale. So:

- Have **upload()** assign a SimpleImage of the uploaded image file to both of the global variables.
- Also have **upload()** draw the original image global variable's Simple Image to the left hand side canvas.

4. Modify the function *makeGray()* so that it modifies the grayscale image global variable to grayscale and then displays the modified image in the right hand side canvas.

Need help? If you get stuck, don't give up! You can do this! Review the past couple videos and seek help in the forums.

Try It! Green Screen Online

Create the prototype web page

1. Create the prototype web page (i.e., Pen) with all necessary HTML elements, before you connect them to the green screen JavaScript code. The page should have the following elements:

- A heading
- Two canvases
- Two file input elements, labeled Foreground and Background
- Two buttons, with values "Create Composite" and "Clear Canvases", respectively

2. Create event handlers for each of the input elements, pointing each handler to a function, e.g., **loadForegroundImage()**, **loadBackgroundImage()**, **doGreenScreen()**, **clearCanvas()**.

4. For the purposes of the prototype, have each function alert the user, such as "foreground image loaded" and check that they each work as expected.

Return to DLTP

1. Consider the green screen code you wrote in the previous module. You created image variables **fgImage**, **bgImage**, and **output**. Then you looked at each pixel in **fgImage** to determine how to set the corresponding pixel in **output**. Finally you printed **output**.

2. Think about which part of this code needs to be executed in the upload functions and where the filename you wrote explicitly will come from.

3. Think about which part of this code needs to be executed after clicking "Create Composite".

4. Think about which variables should be global, once the rest of the code is contained in functions.

5. Finally, think about how you will "print" on the web page.

6. Write an outline of how your HTML, JavaScript, and CSS will be written to accomplish the above steps.

Implement Green Screen

1. Create global variables for the foreground image and background image. You can also create global variables for the canvas elements, since several functions will need to access them.

- Initialize these global variables to null, so that you will be able to check if they are complete.

2. Modify your load image functions so that:

- The foreground image variable is initialized to the selected image and displayed in the left panel.
- The background image is initialized to the selected image and displayed in the right panel.
- Remember: What do you need to do in order to use SimpleImage in this pen?

3. Modify the clearCanvas() function to clear both canvases. Recall the **clearRect** method you used in a previous Try It! exercise.

4. Begin writing your *doGreenScreen()* function with a set of conditionals.

- Check to see if the foreground image is null (i.e., **== null**) OR if the foreground image has not finished loading (i.e., **! fgImage.complete()**); if either of these conditions are true, alert the user that the image has not been loaded.
- Create a similar conditional statement for the background image.
- Otherwise, have the function clear both canvases and execute the following green screen algorithm.

5. Write the green screen algorithm in your *doGreenScreen()* function.

- As part of the final conditional in Point 4, initialize a variable that is a “blank” new SimpleImage with the dimensions of the foreground image.
- Add your green screen algorithm code, making sure to reference your uploaded foreground and background images and newly created final image.
- Specify a green threshold value that you will use to determine if a pixel is “green.” Note that there are many ways to define “green.” You can use a green threshold value, such as 240 used in this lesson, or you can compare the green to the sum of red and blue, as in the last module, or you can devise a way of your choosing.
- The rest of the for loop should be familiar from the code you have written previously.

6. Draw the final composite image to the right canvas.

You can also consider adding checks to your code to display an error message if the images are different sizes. You could also check their sizes and trim one to be the dimensions of the other.

Now test your web page’s code. Go to the DLTP JavaScript enviornment (http://www.dukelearntoprogram.com/course1/example/index.php, also linked in the **Resources** tab) and download the “dinos.png” and “drewrobert.png” image files in the “Available Images” section (both images are 1920x1080). Upload these images to your web page and see if your green screen code works!

Need help? Review the **Moving to CodePen** videos and the earlier videos that helped you design your code. Also seek help in the forums!