CS/INFO 3300; INFO 5100
Homework 10
Due 11:59pm Monday, November 21
(we will also accept late submissions before 11:59pm on 11/28 with no slipday use)

Goals: Practice using d3 to work with raster drawing functions

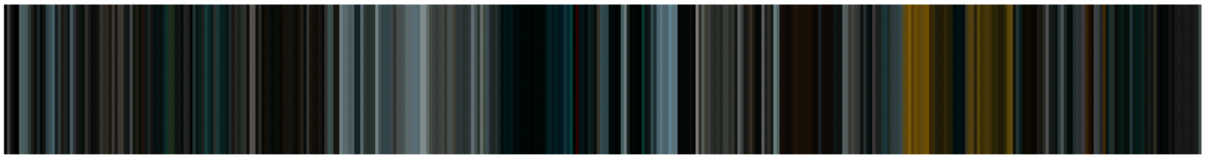
Your work should be in the form of an HTML file called index.html or index.htm with one <p> element per problem. For this homework we will be using d3.js. **In the <head> section of your file, please import d3 using this tag: <script src="https://d3js.org/d3.v7.min.js"></script>**

Create a zip file containing your **HTML file and associated data files** (i.e. movie_barcodes.json) and upload it to CMS before the deadline. Submissions that do not include data files may be penalized. Your submission will be graded using a Python web server run in a parent directory containing your zip file contents along with many other students' submissions.

1. In this assignment, you will create a series of *movie barcodes* using <canvas> elements. A barcode uses vertical lines to show the *average color of individual frames of a film*, helping to reveal how color is used by cinematographers to give flavor to a film. This has been a popular subject of online visualization blogs, to the extent that some have even made publicly available tools for generating them. For this assignment you'll be taking a simpler approach compared to these tools. Your end barcodes will look exactly like this:

The Grand Budapest Hotel

The Green Knight



The Lion King (1994)

Total Recall (1990)

We have generated a movie barcode dataset for an eclectic set of 29 films, `movie_barcodes.json`. The dataset contains a list of Objects, one for each movie barcode. The relevant properties in each of the barcode Objects are:

```
{
  title:  a _string_ identifying the title of the movie
  bars:   a _list_ of individual color bars, used to fill the barcode's canvas
     [ {
           color:  a hexadecimal color _string_ for that specific color bar
           x: the x integer position of this specific color bar
       } …
     ]
}
```

You will find 800 individual bars inside of each movie's `bars` list. Your eventual `<canvas>` elements will be 800 pixels in width. This means that you will have a single 1px wide vertical bar for each item in the `bars` array. You'll have 29 individual `<canvas>` elements, one for each movie Object.

There are any number of ways to solve this assignment. For class purposes, we will pick the simplest path that *does not* involve data joins and instead just uses simple loops. You are welcome to use joins if you are confident that you can solve the problem. You will have a choice between two different methods for modifying your canvas. While we suggest that you use the array method, the line method will deliver identical results.

(next page)

(TECHNICAL DATASET DETAILS (feel free to ignore): The dataset was generated by sampling 400 frame images from each movie file. They were sampled evenly over the duration of the film (i.e. total_frames / 400). For each of those individual frame images, there are any number of strategies for finding the dominant or average color. Though I tried RGB, LAB, and even K-means clustering, averaging XYZ colors gave the nicest looking results. Within each image, I converted each RGB pixel into the XYZ color coordinate space. I then computed the average XYZ value for all the pixels. If you want to try this, you might also consider additional statistical methods such as median. The average was then converted back into a hexadecimal RGB color for use in the dataset. To obtain 800 individual bars, each of the 400 frames was duplicated, making thicker barcode lines.)

Procedure:

A. Begin by creating an asynchronous function and importing the dataset, `movie_barcodes.json`, using `d3.json` and `await`. Make sure that you call your async function.

B. In your HTML, create a `<div>` element that will contain all of your barcodes
(e.g. `<div id="barcodes"></div>`)

C. Now iterate through each of the movie Objects in `movie_barcodes` using a `forEach` loop.

-- **D**. For each of the movies, create a new `<div>` element to hold that movie's title and canvas.
Add an `<h5>` element inside of the new `<div>` containing the movie's title (hint: use .text())
Add a new `<canvas>` element inside of the new `<div>`. Set its `width` to 800 and `height` to 100.

-- **E**. Obtain a drawing context for the new `<canvas>`.
You now have two choices for how to draw the barcode onto the movie canvas:

-- -- *Option A. Draw vertical lines for each of the bars*

Using a nested `forEach` loop, begin iterating through the bars array

For each of the bars, set the `strokeStyle` of the drawing context to the bar's `color`

For each of the bars, use the context to draw a *vertical line* from top to bottom.

To determine the x-position of the bar, refer to the bar's `x` property and add 0.5 to it.
(We add 0.5 so that the 1px wide line properly fills that column of pixels.)
(If the bars appear to be very light in color, you forgot to add 0.5)

-- -- *Option B. Manually manipulate the pixel array*

Use the `getImageData` function on the drawing context to get an array of canvas pixels

Using a nested `forEach` loop, iterate through the bars array within the movie Object.

For each bar, get a hexadecimal color string from its `color` property. Use the following code to compute `r`, `g`, and `b` values from the hex color string:

```
let num = parseInt(color.replace('#',''), 16);
let r = (num >> 16) & 255;
let g = (num >> 8) & 255;
let b = num & 255;
```

Get the x-position for this bar using the bar's `x` property

Using a nested `for` loop, iterate from 0 to 100. This will become your `y` value.

Within that loop, use array operations to set the pixel array's `r`, `g`, and `b` values for each (`x`, `y`) position. This will fill in each bar pixel-by-pixel

(hint: this approach has three nested loops: one looping through movies, another through bars, and a final one looping through pixels from 0 to 100)

Finally, use `putImageData` to paste your pixel array onto each canvas

Part F of this problem is completely optional.
Students who successfully complete Part F will receive one free quiz grade drop. If you've missed a quiz, this is an easy chance to win back one of those 0 grades.

In Part C to E, you were asked to use a forEach loop to create each of the `<div>` elements, `<h5>` elements, and `<canvas>` elements. **For Part F, we would like you convert the code for whichever option you chose (both will work) to use a data join.** The example from the lollipop chart lecture will be helpful if you aren't sure how to achieve this goal.

Here is a short overview of the steps necessary:

1. Select your container `<div>` using d3.

2. Use a data join to make `<div>` elements for each item in your data array. Make sure to bind the result of the data join to a variable so you can use it again.

3. In a separate chain, run `.append()`to add an `<h5>` tag inside of each `<div>`. Since it is a child node of a data join, you will have access to `d =>` functions.

4. In a separate chain, run `.append()`to add a `<canvas>` tag inside of each `<div>`. Configure the canvas so it has the right size and shape. Make sure to bind the result of this to a variable so that you can iterate through the canvases later.

5. Instead of using a `.forEach()` to iterate through the dataset (which only works on JS arrays), you need to iterate through your `<canvas>` tags. Use the d3 function, `.each()` to do so. Check out the documentation for `.each()` if you are unfamiliar. Within your `.each()` call, run the same basic JS functions you did in Part E. You will have to make use of the reserved keyword `this` in order to access each of the canvas objects to draw on them.

If you succeed in adding a data join, you can submit that version instead of the original code and still receive full credit. You do not need to keep two versions of the same code.

REFERENCE:

`index.html` should look something like this *before* your code runs:

```
<html>
      <head> </head>
      <body>
            <p> #1 </p>
            <div id="barcodes">
                  ( This is where content will be added )
            </div>
            <script>
                  ( Your code here )
            </script>
      </body>
</html>
```

`index.html` should look something like this *after* your code runs:

```
<html>
      <head> </head>
      <body>
            <p> #1 </p>
            <div id="barcodes">
                  <div>
                        <h5>Aladdin</h5>
                        <canvas width="800" height="100">
                  </div>
                  <div>
                        <h5>Avatar</h5>
                        <canvas width="800" height="100">
                  </div>
                  <div>
                        <h5>Avengers — Infinity War</h5>
                        <canvas width="800" height="100">
                  </div>
                  ...
            </div>
            <script>
                  ( Your code here )
            </script>
      </body>
</html>
```