Scrollytelling Tutorial

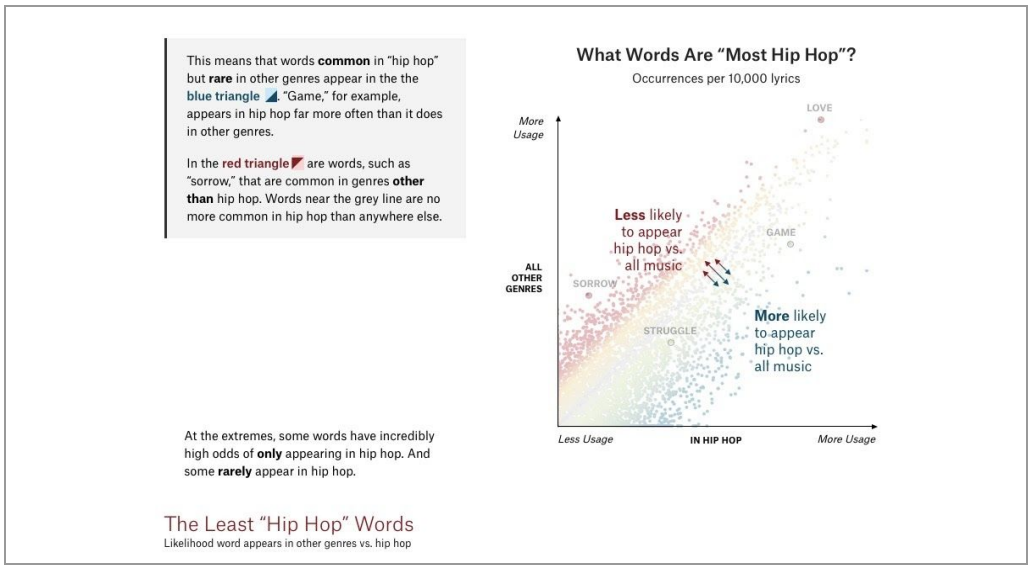
Big Data Class

Readings prior to class:

- “Emerging and Recurring Data-Driven Storytelling Techniques: Analysis of a Curated Collection of Recent Stories” from Microsoft Research [link]
- “How to Implement Scrollytelling with Six Different Libraries” from The Pudding [link], “Responsive Scrollytelling Best Practices” from The Pudding [link], and Easier Scrollytelling with Position Sticky” from The Pudding [link]

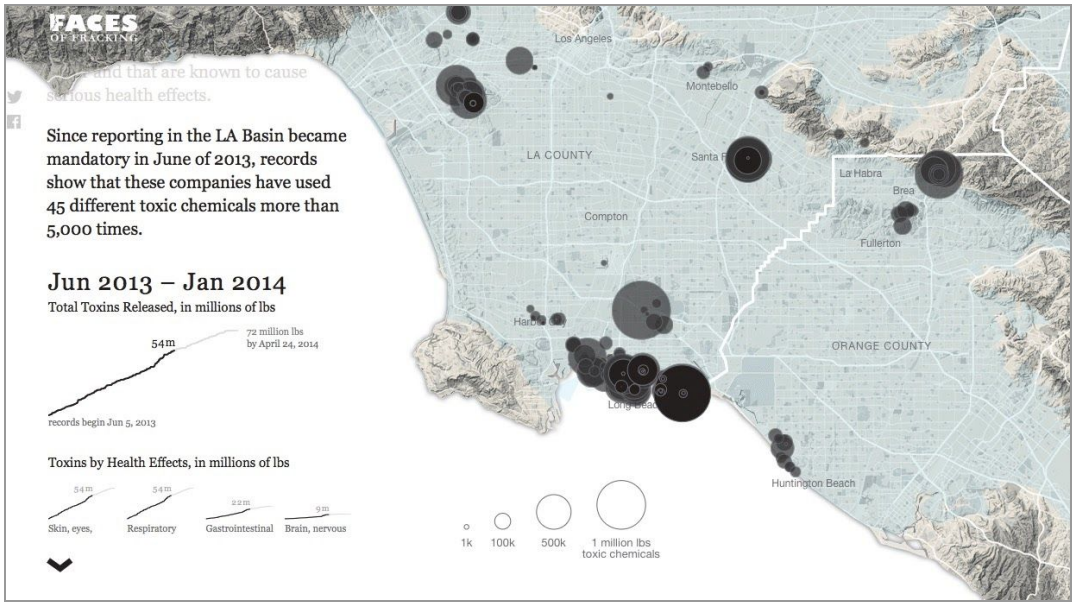
Scrollytelling is a technique often used in data visualization; with scrollytelling, elements are revealed as users scroll through a story, ultimately building a narrative structure. Scrollytelling stories often combine text, data visualizations, and other media to create a visual narrative. Scrollytelling is often used in visual narratives by The New York Times and other news outlets, and new publications such as The Pudding are also pioneering scrollytelling. Here are a few examples of scrollytelling:

“The Language of Hip Hop” from The Pudding
The Pudding is an online publication that produces visual data-driven narratives and frequently use scrollytelling (they produce incredible work and are a great resource for scrollytelling inspiration). “The Language of Hip Hop” analyzes similarity between the lyrics of hip hop artists and visually illustrates the results. The analysis uses machine learning and an algorithm called t-SNE.



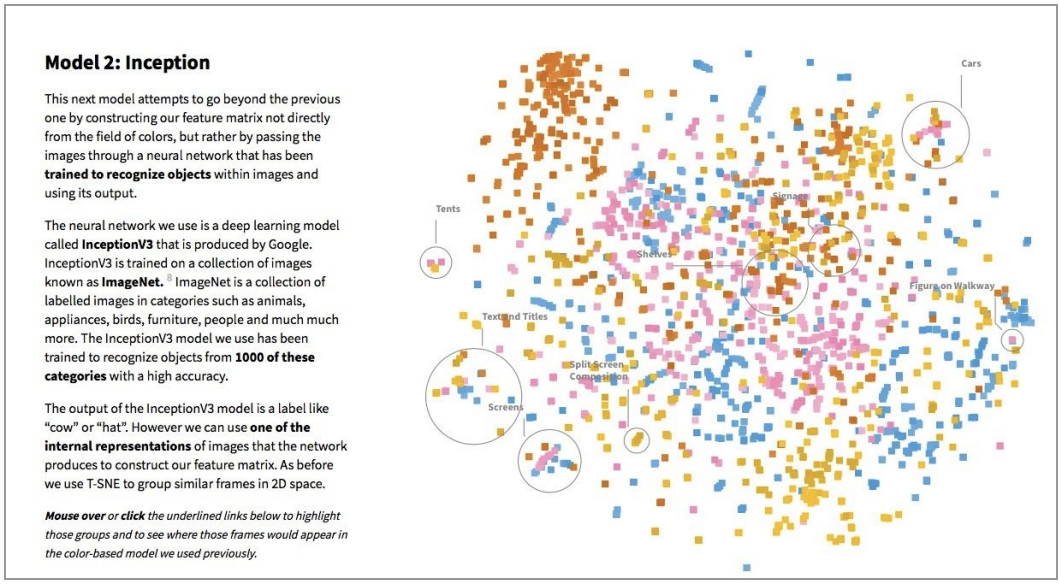
[Faces of Fracking]

“Faces of Fracking” is a scrollytelling site that features map-based narrative. The site uses D3 and illustrates fracking in California.



[“Machine Visions: Exploring Visual Motif’s in Wes Anderson Films”] by Yannick Assogba

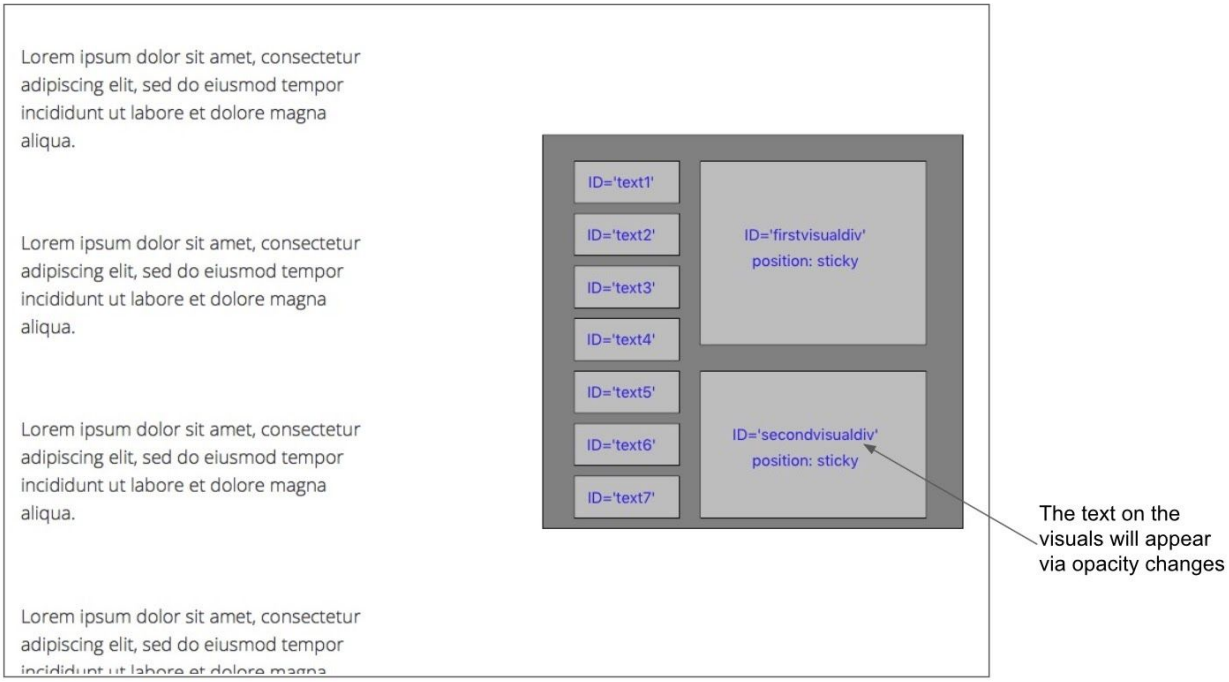
This project is an exploration of visual motifs in Wes Anderson Films and explains the methodology behind the data analysis. The project uses machine learning (neural nets and t-sne) to analyze the similarity between frames in Wes Anderson films. The scrollytelling site illustrates the methods and results of this analysis.



Creating a basic scrollytelling site with Scrollmagic.js and Bootstrap

Now that we've seen some scrollytelling examples, we'll implement a basic scrollytelling site using Bootstrap and Scrollmagic.js (a Javascript library for Scrollytelling).

The site will be a visual illustration of scrollytelling itself! It'll look something like the image below; the visuals on the right will change and the text over the visuals will appear and disappear via opacity changes. To implement this, we'll use Scrollmagic to trigger changes to CSS classes; specifically, we'll change the opacity of elements.



Follow along with these steps:

Step 1: Download the basic site
Download the basic site structure: scrollytelling_demo_index.html. This site structure includes some basic visuals made with D3. For now, don't worry about how the D3 visuals are created. We'll primarily be working with Scrollmagic.js and CSS to change opacity as the user scrolls through the site.

Open the file in your text editor. You'll should see the following:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Scrollytelling Demo</title>
    <script src="https://d3js.org/d3.v4.min.js"></script>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,500" rel="stylesheet">
    <style>

        p {
            font-family: 'Open Sans', sans-serif;
            color:#353536;
            font-weight:300;
        }

        svg {
            width:100%;
            height:600px;
            padding-left:5%;
            padding-right:50px;
        }

        h1 {
            font-size:60px;
            font-weight:200;
            line-height:1.3;
        }

    </style>
</head>
<body>
  <main role="main" class="container" >
     <div class="row">
        <div class="col-6">
            <br><br>
            <h1>Scrollytelling Demo</h1>
            <hr>
        </div>
     </div>
     <div class="row">
     </div>
  </main><!-- /.container -->
</body>

<script type="text/javascript">
```

The basic site structure includes basic styles (for paragraphs, h1, and svg), very basic body structure, and Javascript for the D3 visuals. Remember, the basic site structure of most websites involves:

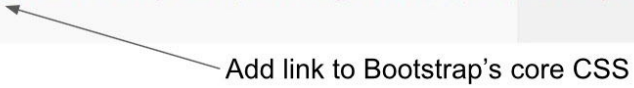
<head></head> ---> links to external resources and site metadata go here
<body></body> ---> most of your site content goes here
<script>/<script> ---> your Javascript goes here

Step 2: Add Bootstrap's core CSS file

This site will use the Bootstrap framework's grid. To use this grid, you need to link to Bootstrap's core CSS file. Insert the following code between the <head></head> tags to link to Bootstrap's CSS:

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css">

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Scrollytelling Demo</title>
    <script src="https://d3js.org/d3.v4.min.js"></script>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,500" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css">
    <style>
```



Step 3: Add Scrollmagic Javascript library

Since we'll be using Scrollmagic.js, a Javascript library implementing scrollytelling, you'll need to add a link to the Scrollmagic Javascript library. We'll draw from this library as we build out the site. To use Scrollmagic.js, add the following code between the <head></head> tags on the site:

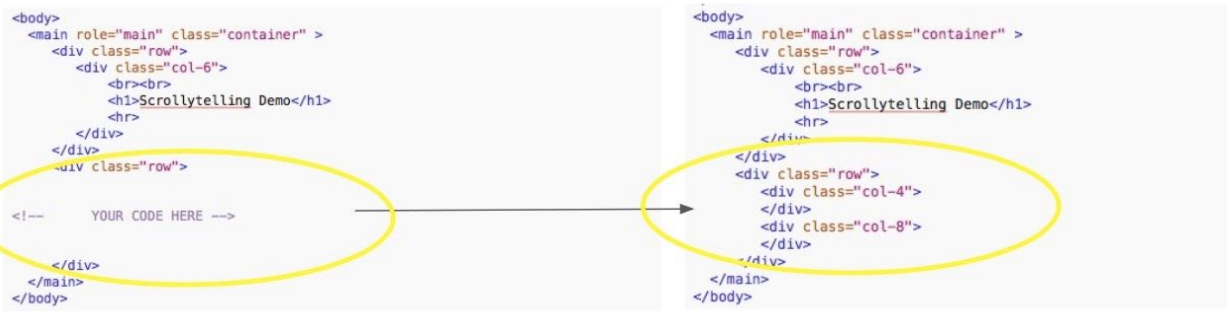
<script src="https://cdnjs.cloudflare.com/ajax/libs/ScrollMagic/2.0.5/ScrollMagic.min.js"></script>

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Scrollytelling Demo</title>
    <script src="https://d3js.org/d3.v4.min.js"></script>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,500" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/ScrollMagic/2.0.5/ScrollMagic.min.js"></script>
    <style>
```

Step 4: Add two columns to the site

The structure we'll use on this scrollytelling site is a two column structure where the left column contains text and the right column contains visuals. This is a common structure for scrollytelling sites, as we saw in the examples at the beginning of this tutorial.

Bootstrap makes it simple to create a two-column structure. To add the columns, use Bootstrap's built in classes for columns. Locate the second row in the site ( ). Within this div, add two additional divs back-to-back. The first div should have class="col-4" and the second div should have class="col-8."



Bootstrap uses a 12-column grid system, so col-4 and col-8 determine the relative width of these two columns. The divs in the left column (containing text) use Bootstrap's class="col-4". The divs in the right column (containing visuals) use Bootstrap's class="col-8".

This creates the basic structure for the site. It has two columns, and we'll put all the descriptive text in the left column, and all the visuals in the right column. The left column will primarily contain paragraph elements, and the right column will primarily contain divs that house the visuals.

Step 5: Add two more empty elements

We've added two empty elements so far, which site side-by-side in a two column layout. Now add two additional elements, with the same classes as above. We now have two main sections for our content.

Your div's will now be as follows (though nothing will show up visually yet since the divs are empty):

Step 6: Add paragraph elements to the left columns

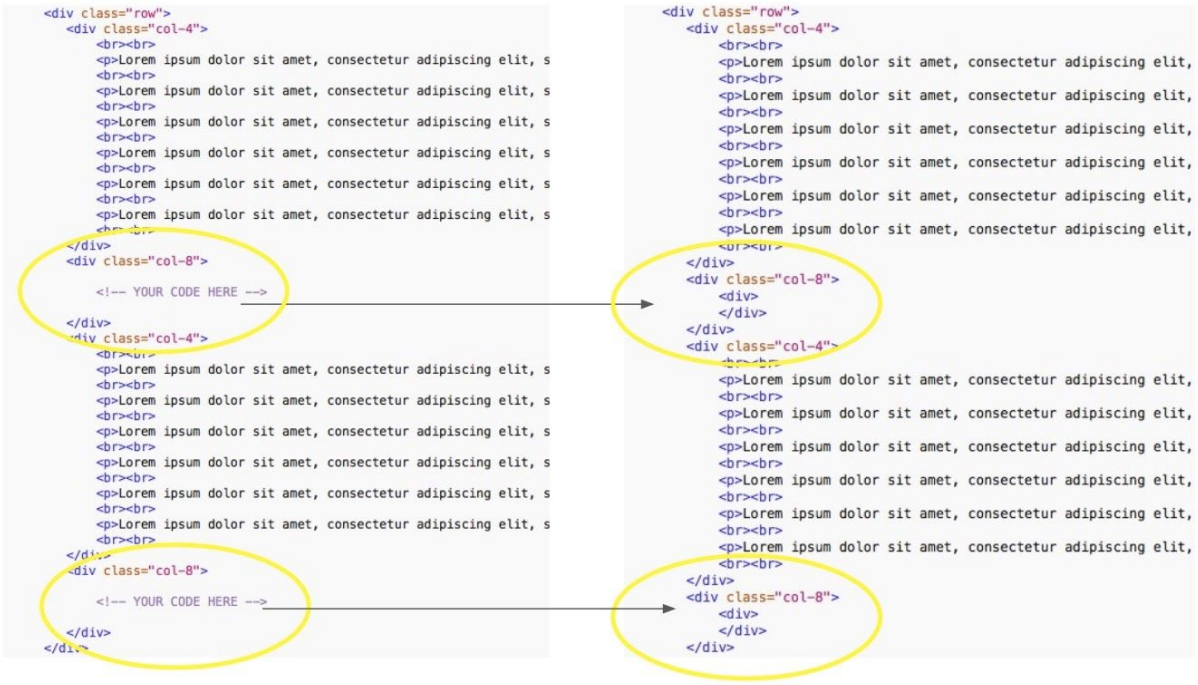
The left column contains the text content for the site. Add tags in each of the divs with class “col-4” (remember, this is the left column). The exact number of paragraphs can vary, but we suggest adding 6 paragraphs to each div. Add two breaks using the
 tag between each paragraph. Your code should look as follows:

Step 7: Add text to the paragraph elements

Add text to each of the paragraph tags. You can use lorem ipsum or your can write your own text. Each paragraph should be fairly short, between 20 and 40 words.

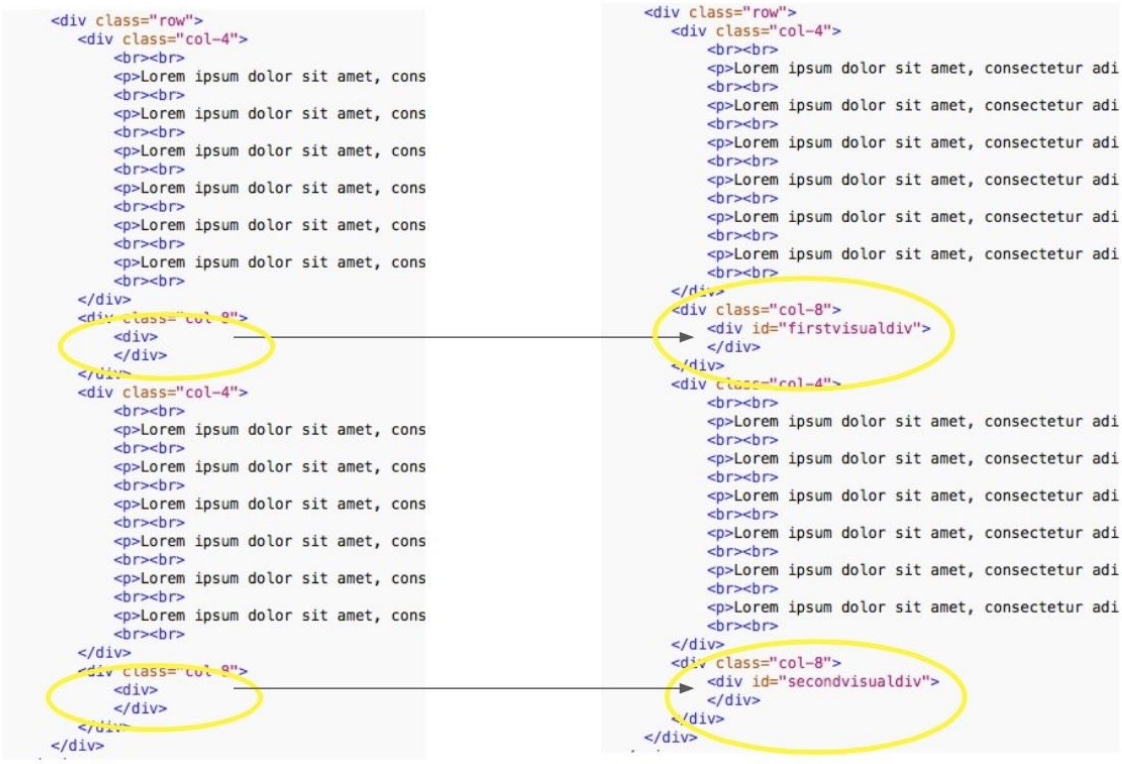
Step 8: Add divs to the right column

The right column will contain the visuals. The right column currently has two divs with class col-8. In each of these div’s, add an addition div (this is where we’ll house the visuals) as illustrated below:



Step 8: Add IDs to elements in right column

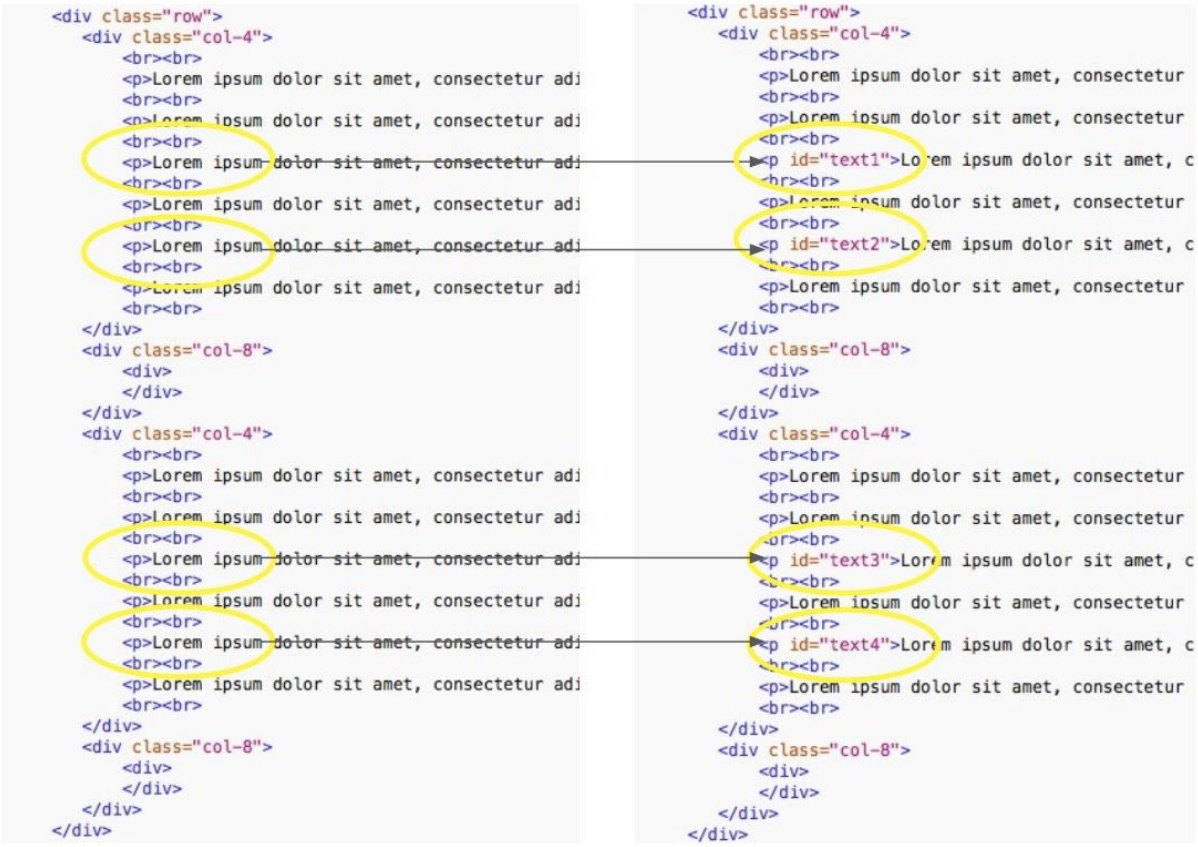
In each column, add IDs to each div element or paragraph element. You'll use these IDs to control the "trigger" positions with ScrollMagic.js. For the div's in the right column, use the IDs "firstvisualdiv" and "secondvisualdiv" as follows:



Step 9: Add IDs to elements in left column

In the left column, we'll add IDs to some of the paragraph elements. The IDs we've added to the paragraph elements will be especially helpful later when we use Scrollmagic to trigger changes to the visuals. When the scroll position encounters a specific paragraph element (based on the ID), the visuals will change. The IDs on the paragraph elements are critical for determining the scroll trigger position.

Add IDs to the 3rd and 5th paragraph element of each div. Use ID “text1”, “text2”, “text3” etc. as follows:



Step 10: Add position: sticky to the elements for visuals

Using CSS, set the two divs in the right column (the div's that contain the visuals) to position sticky. Now that you've added IDs to the two divs for visuals, you can use these IDs to add styles. Add the following code within the <style></style> tags on the site:

```
#firstvisualdiv, #secondvisualdiv {
        position: -webkit-sticky;
        position: sticky;
        top: -1px;
}
```

Using position: sticky will keep the divs in the right column stationary (while the text in the left column keeps scrolling) until you reach the next section of content on the site. It's very easy to implement and works reasonably well across browsers.

Step 10: Set styles for the text that will appear on the visuals

The visuals in the right hand columns will change as the user scrolls; specifically text will appear that annotates different elements in the visuals. These text elements have classes that we've already specified in the D3 code. Don't worry about how we set these in the D3 code, since we haven't learned D3 yet. Just keep in mind that the text over the visuals have the following classes: text1opacity, text2opacity, and text3opacity. They have "opacity" in the class name because ultimately we'll be changing the opacity of these text elements to make them appear and disappear on the visuals. The following CSS code defines the styles for this text (color blue and size 30px):

```
.text1opacity, .text2opacity, .text3opacity {
        font-size:30px;
        fill:blue;
        }
```

Step 11: Set default opacity for the text that will appear on the visuals

We'll use Scrollmagic to trigger opacity changes to the text on the visuals. Thus, we need to set initial opacity of 0 (so the text doesn't appear until we trigger the state change). We'll also add a transition opacity 0.8s (this will give a short transition to the opacity state change). Your styles should be as follows:

```
.text1opacity, .text2opacity, .text3opacity {
        font-size:30px;
        fill:blue;
        opacity:0;
        transition: opacity 0.8s;
        -webkit-transition: opacity 0.8s;
        }
```

Step 12: Set "active" state opacity for the text on the visuals

We'll use Scrollmagic to trigger an opacity change on the text on the visuals; an "active" state will be triggered where the opacity is 1. To define this active state, add the following CSS:

```
.active {
        opacity:1;
}
```

Step 13: Add a Scrollmagic "controller"

To use Scrollmagic, you set up a controller and then add different scenes. The first step is to add the "controller" which you can do with the following code. Make sure to add this within the <script></script> tags since it's Javascript, and also be sure to add this code below all the existing Javascript in the <script></script> tags.

```
var controller = new ScrollMagic.Controller();
```

Step 14: Add a Scrollmagic "scene"

Next, we'll add a "scene" to change the CSS opacity of the text on the visuals. To do this, we add the following code:

```
new ScrollMagic.Scene({triggerElement: "#text1"})
        .setClassToggle(".text1opacity", "active")
        .addTo(controller);
```

Let's break down what these mean:

triggerElement → which element will trigger the state change (this is the ID of one of the paragraph elements). In this case, the paragraph element with ID="text1" will trigger

the change. Thus, when the user scrolls and reaches the paragrah with ID="text1", the CSS change will be triggered.

.setClassToggle(".text1opacity", "active") → This changes the CSS class of all elements with class="text1opacity" to "active." As we set earlier in our CSS, "active" has an opacity of 1. So this class toggle is changing the opacity of these elements from 0 (their default) to 1 (the active state).

.addTo(controller); → This adds the scene to the controller.

You've now added a scene. Your code should look like this:

```
var controller = new ScrollMagic.Controller();

            new ScrollMagic.Scene({triggerElement: "#text1"})
                        .setClassToggle(".text1opacity", "active")
                        .addTo(controller);
```

Step 15: Add additional scenes

We'll have a total of two scenes, so there are two more to add as follows:

```
new ScrollMagic.Scene({triggerElement: "#text2"})
        .setClassToggle(".text2opacity", "active")
        .addTo(controller);

new ScrollMagic.Scene({triggerElement: "#text3"})
        .setClassToggle(".text3opacity", "active")
        .addTo(controller);
```

These function the same as the first scene we added, but they are triggered at different paragraph IDs (#text2 and #text3). They also change the opacity of different classs (.text2opacity and .text3opacity). After you've added this code your final Scrollmagic code should look like the following:

```
var controller = new ScrollMagic.Controller();

            new ScrollMagic.Scene({triggerElement: "#text1"})
                            .setClassToggle(".text1opacity", "active")
                            .addTo(controller);

            new ScrollMagic.Scene({triggerElement: "#text2"})
                            .setClassToggle(".text2opacity", "active")
                            .addTo(controller);

            new ScrollMagic.Scene({triggerElement: "#text3"})
                            .setClassToggle(".text3opacity", "active")
                            .addTo(controller);
```

Your site should now be fully functioning! You should be able to scroll through your site and view how the visuals change (i.e. text annotation appears on the visuals) as you progress through the site. Though this is a very simple example, the concept could be expanded to a longer and more elaborate site.

Other libraries for scrollytelling

We just used Scrollmagic.js to implement scrollytelling, but there are many other Javascript libraries you can use to implement scrollytelling. The Pudding article “How to implement scrollytelling with six different libraries” has helpful info on some of these libraries:

- Scrollstory
- Scrollarama
- Waypoints

Additional scrollytelling tools

- Mapbox scrollytelling template

Resources

Scrolling in Data Visualization (Jim Vallandingham)
How to Scroll (Mike Bostock)
From Storytelling to Scrollytelling: A Short Introduction and Beyond (by Lorenzo Amabili)