# Geography 572
## Lab #3: Aesthetic Styling Challenge

## Lab Objectives:

- Deconstruct an existing aesthetic style to its constituent form, colors, type, and textures;
- Apply these stylistic components to a multiscale, "slippy" tileset using CartoCSS, extending your cartographic design workflow to the web for the first time;
- Serve your own set of tiles using Mapbox Studio Classic;
- Use pictorial icons to represent points of interest using Mapbox Editor.

## Evaluation:

This lab is worth **25 points** toward the Lab Assignments evaluation item, which is worth 25% of your overall course grade. A grading rubric is provided at the end of the lab to inform your work.

## Schedule of Deliverables:

- **October 19th:** Lab #3 Assigned //client contract begins
- **October 26th:** Inspiration board & POIs due //meet the artist
- **November 2nd:** CartoCSS Tips w/ Katie Kowalsky //design consultation
- **November 9th:** Lab #3 Due //exhibit launch

## Challenge Description

You have been contacted by the director of a local art museum to design a web map that promotes an upcoming gallery exhibit. Since the museum is renowned for its forward thinking use of digital media, the map should have an aesthetic style consistent with the gallery exhibit. Thus, the form, color, type, and texture of the basemap tileset should confirm to artist's stylistic choices. Finally, the director wants the exhibit to receive international attention, so wants you to include ~25 worldwide locations that fit with the gallery exhibit theme in some way; the director is allowing you to determine which locations are mapped and how you represent them with icons.

### *Notes from the Art Director*

Your design must be based on a deconstruction of an ***inspiration board*** (~three carefully-selected and thematically-related examples from art, advertising, graphic design, etc.) into its constituent components based on form, color, type, and texture. Your inspiration board can include any set of coherent visual designs, and is not constrained by medium or any single conceptualization of "art". Your basemap tileset will be built to incorporate all zoom levels, from a view of the world (referred to as zoom level 0) to street level view (as far as 22). The basemap should be pannable and zoomable (i.e., a ***slippy map***).

Your tilesets must include at least ***25 points of interest*** on the map (i.e., location markers). POIs can include any kind of locations (e.g., more than just museums containing art) as long as they fit the exhibit/style theme in some way. Each kind of POI should be represented with an unambiguous iconic point symbol. Selection of the POIs must activate an information window.

## 1. Creating an Inspiration Board

A common task in professional cartography is the reproduction of an existing style, perhaps when join a mapping team that already has an established aesthetic (e.g., the National Park Service, National Geographic) or when receiving a client request to leverage existing branding and promotional materials for the map design. Until recently, this style recreation occurred primarily in print production due to limitations in customizing web mapping technology.

Luckily, cartographers now can reclaim aesthetic style on the web using ***CartoCSS***, a specification for cartographic styling in web design primarily used for designing slippy basemaps. As introduced in athe *Primer on CartoCSS*, the CartoCSS scripting language is a styling option supported by emerging technology leaders in web cartography, such as CartoDB and Mapbox. In Lab #3, you'll extend your existing knowledge of the graphic design workflow to the web using CartoCSS along with Mapbox Studio Classic and Mapbox Editor!

So, what do you need to consider as you deconstruct an existing style? In lecture, you are encouraged to consider four components of style:

- ***Form:*** The variable aspects of the linework you are styling, including its level of generalization (simplification, smoothing, etc.) as well as line weights, cap and joint styles, tapering, etc.;
- ***Color:*** The overall color palette, including both basic palette colors and accent colors;
- ***Type:*** The selected typefaces and their size and style variant, as well as the placement of type on the page;
- ***Texture:*** The additional pattern fills and overlays that add visual complexity and aesthetic embellishments.

Your first step in completing Lab #3 is to create an inspiration board of ~three pieces of artwork by which you will be styling your multiscale tileset. Your inspiration board should be largely (although not completely) consistent in color, type, and texture. However, **before** you finalize your inspiration board, confirm that you actually can reproduce these stylistically components in CartoCSS by reviewing the *Primer on CartoCSS* and the online CartoCSS reference: https://github.com/mapbox/carto/blob/master/docs/latest.md. **Figure 1** provides an inspiration board for the ***Pop Art*** style introduced in lecture, specifically drawing from the work of Roy Lichtenstein, a renown 1960s artist known for his comic-like paintings. The examples provided in this lab make use of the Lichtenstein example, but you should modify the instructions for your style.

Your inspiration board must include **three** components:

- ~Three inspiration board visuals of consistent style;
- A description of the aesthetic style broken down by: (1) form, (2) color, (3) type, and (4) texture;
- A preliminary listing of CartoCSS style rules identified from the reference that you could apply to the following source layers to achieve these stylistic components: (1) water features, (2) political units, (3) cities, (4) roads, and (5) land cover.

We have provided a worksheet to organize your stylistic components versus CartoCSS style rules. The artboard + worksheet are due on **October 26th**.

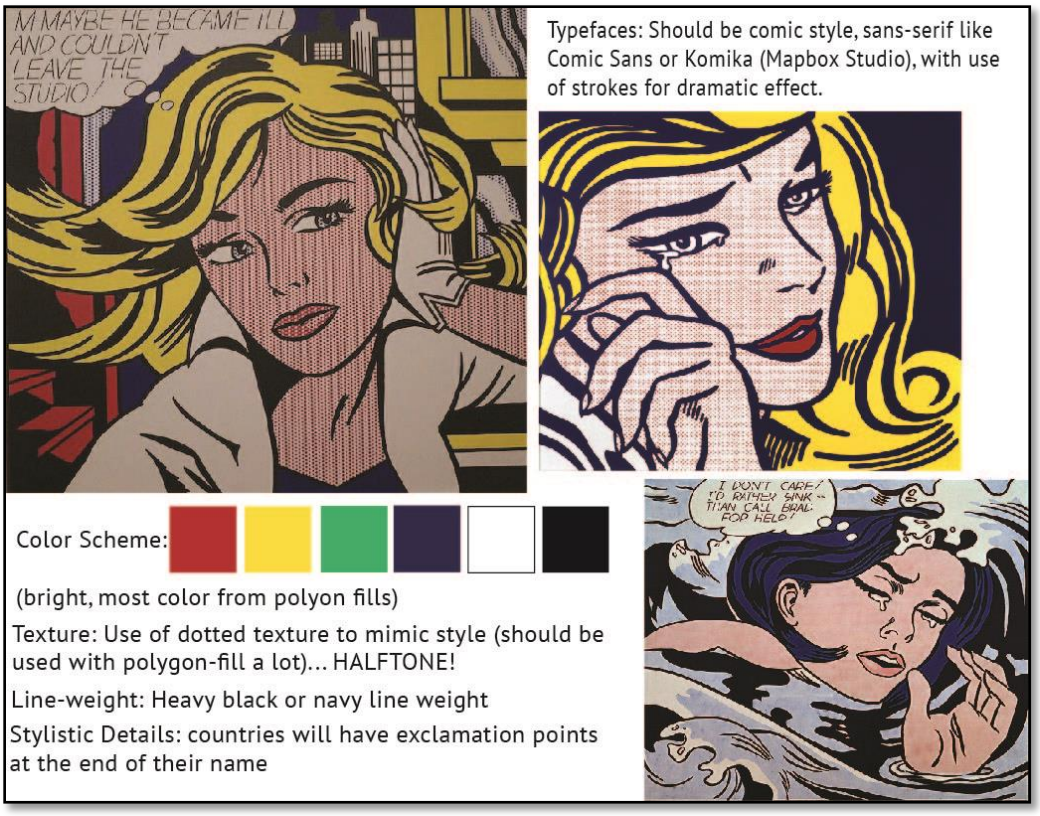


**Figure 1: An example inspiration board, based off Roy Lichtenstein paintings.**

## 2. Assembling Your Points of Interest

After creating your inspiration board, next consider your points of interest (POIs). The POIs you collect and map must be based on real information. Focus on collecting features at a point dimensionality. While you can include linear or areal features in your map (e.g., a scenic drive, a park), you will need to represent them as points for Lab #3. Although this may seem like a limitation, the conceptualization and subsequent representation of features as points is common on web maps due to the provision of integrated designs at multiple scales, the smallest of which require collapsing to points.

It is up to you to decide the locations you'll feature as POIs on your map but keep in mind the artistic theme of the scenario. Regardless of the features you ultimately decide to map, you will need to collect **four** pieces of information about each feature:

1. **Name:** The specific name of the instance of the feature. You will choose a point icon from the Mapbox Maki library based on the name; multiple POIs can have the same icon if they represent the same category of thing.
2. **Latitude:** In decimal degrees, using negative numbers for latitudes in the Southern Hemisphere. It is recommended that you make use of the iTouchMap tool (http://itouchmap.com/latlong.html) to determine the latitude and longitude of your POIs, if you do not otherwise have a georeferenced file. Use a precision of six decimal places to accurately place each POI.

3. **Longitude:** In decimal degrees, using negative numbers for longitudes in the Western Hemisphere. Mapbox Editor searches for any column with the terms "lat"/"latitude" and "long"/"longitude", using the first instance found in the spreadsheet for the X or Y coordinate, respectively.
4. **Description:** A brief (100 words or less) description of the feature, perhaps with a link to additional information. You should compose the descriptions yourself such that the set of descriptions are at a common length and contain a common set of summary information. Both the name and description columns will be used to populate an information window upon clicking the icon.

| | location | lat | long | painting |
|---|---|---|---|---|
| 2 | Albright-Knox Art Gallery | 42.932976 | -78.875618 | Head: Red and Yellow, Picture and Pitcher |
| 3 | Art Institute of Chicago | 41.8795 | -87.624089 | Alka Seltzer |
| 4 | Arthur M. Sackler Gallery | 38.88765 | -77.026295 | Landscape in Scroll |
| 5 | Chrysler Museum | 36.85611 | -76.293546 | Live Ammo (Ha! Ha! Ha!) |
| 6 | Cranbrook Art Museum | 42.57273 | -83.24947 | Modular Painting with Four Panels, #7 |
| 7 | Denver Art Museum | 39.73685 | -104.98946 | The Violin |
| 8 | Des Moines Art Center | 41.58393 | -93.681963 | Perforated Seascape #1, The Great Pyramids |
| 9 | Detroit Institute of Art | 42.35911 | -83.065223 | Interior With Mirrored Closet, Modern Sculpture with Three Voids, Modern Sculpture (Maquette) |
| 10 | Fogg Art Museum, Harvard University | 42.37402 | -71.113725 | Reclining Nude |
| 11 | Frederick R. Weisman Art Museum | 44.97332 | -93.236969 | World's Fair Mural |
| 12 | Hirshhorn Museum and Sculpture Garden | 38.88785 | -77.022945 | Modern Painting with Clef, Modern Painting with Sun Rays, Modern Sculpture with Black Shaft |
| 13 | Lowe Art Museum, University of Miami | 25.7191 | -80.275899 | Modular Painting with Four Panels, #5 |
| 14 | Milwaukee Art Museum | 43.039865 | -87.897594 | Crying Girl |
| 15 | Musée d'Art Contemporain de Montréal | 45.50769 | -73.567092 | Brushstroke Mural for the University of Dusseldorf |
| 16 | Museum of Contemporary Art (LA) | 34.05339 | -118.25071 | Many |
| 17 | Museum of Contemporary Art (San Diego) | 32.84442 | -117.27772 | Mirror Six Panels #3 |
| 18 | Museum of Modern Art | 40.76123 | -73.977745 | Many |
| 19 | Modern Art Museum of Fort Worth | 32.74878 | -97.36321 | Atom Burst, Mr. Bellamy |
| 20 | Museum of Fine Arts | 42.33889 | -71.093812 | Seascape |
| 21 | Nasher Sculpture Center | 32.78794 | -96.800355 | Double Glass, Head with Blue Shadow, Peace Through Chemistry |
| 22 | National Gallery of Art | 38.89177 | -77.019954 | Many |
| 23 | Nelson-Atkins Museum | 39.04536 | -94.580789 | Still Life in Yellow and Black |
| 24 | Norton Simon Museum | 34.1458 | -118.15931 | Long Modern Sculpture, Modern Painting for Expo '67 |
| 25 | Parrish Art Museum | 40.90523 | -72.364566 | Collage for Apple, Drawing for Leda and the Swan, Drawing for Peace through Chemistry |
| 26 | Philadelphia Museum of Art | 39.96621 | -75.181717 | Still Life with Goldfish |
| 27 | Rhode Island School of Design Museum | 41.82686 | -71.40731 | Pyramids II |
| 28 | Rose Art Museum, Brandeis University | 42.3655 | -71.262058 | Forget It! Forget Me! |
| 29 | Saint Louis Art Museum | 38.63955 | -90.294083 | Curtains, Goldfish Bowl II, Him, Sailboats |
| 30 | San Francisco Museum of Modern Art | 37.78562 | -122.40116 | Many |
| 31 | Seattle Art Museum | 47.60718 | -122.33841 | Study for Vicki! |
| 32 | Solomon R. Guggenheim Museum | 40.78279 | -73.959143 | Many |
| 33 | Spencer Museum of Art University of Kansas | 38.95963 | -95.244903 | No- Nox (drawing) |
| 34 | University Art Museum: California State Long Beach | 33.7834 | -118.11488 | Study for Purist Still Life |
| 35 | Virginia Museum of Fine Arts | 37.55572 | -77.474145 | Gullscape, Still Life with Folded Sheets |
| 36 | Whitney Museum of American Art | 40.73942 | -74.008884 | Goldfish Bowl, Little Big Painting, Modern Sculpture with Velvet Rope, Still Life with Crystal Bowl |
| 37 | Yale University Art Gallery | 41.30822 | -72.930798 | Many |
| 38 | Albertina (Batliner Collection) | 48.20463 | 16.367796 | Face (Green Nose), Reflections on Hair, Glass and Lemon in a Mirror |
| 39 | Basil & Elise Goulandris Foundation | 37.83867 | 24.938869 | Nude with White Flower, Sunrise |
| 40 | The Berardo Collection | 38.69536 | -9.209594 | Mirror #4, Interior with Restful Paintings |

**Table 1 provides example POI information for Lichtenstein example, identifying museums with Lichtenstein pieces worldwide.**

Your information must be stored in a *comma separated file*, or ***.csv***, for loading into Mapbox Editor. As its name implies, a *.csv* file is a flat text file that delimits rows using the comma (",") character. To create your *.csv* file, you can assemble your spreadsheet in Microsoft Excel (as shown in **Table 1**) and export using *Save As→Save As Type: CSV.* You are required to have your *.csv* file assembled and ready for mapping on **October 26th**.

# 3. Styling Your Tileset with Mapbox Studio Classic & CartoCSS

## a. Getting Started in Mapbox Studio Classic

Once you have your inspiration board and POI data, you can begin using the Mapbox products. Mapbox is a relatively new mapping company (~5 years old) focusing explicitly on web mapping. Mapbox supports a variety of web mapping tools. For Lab #3, you will use the Mapbox Studio Classic desktop application to style your tileset based on your inspiration board and Mapbox Editor to turn your tileset into a slippy web map and add your POIs.

The *Primer on Mapbox Studio Classic* provides the information needed to start designing with the tool. As described in the *Primer*, first create an educational account (https://www.Mapbox.com/education/) and then open Mapbox Studio Classic, log into your account, and create a *New Project* for your Lab #3 tileset. You can download a free version of Mapbox Studio Classic if you would like to work on your own machine at: https://www.mapbox.com/mapbox-studio-classic/.

Creating a new project will open the Mapbox Studio Classic design interface (**Figure 2**). You will find a default set of CartoCSS rules already defined in the *Style Editor*, all of which are applied consistently across zoom levels. Once again, refer to the *Primer on Mapbox Studio Classic* for details about each interface function. You also are encouraged to complete the interface tour included in the *Docs* tab. Importantly, opening a new project also creates a file directory on your hard drive or flash drive with a ***.tm2*** extension (e.g., `sample-project.tm2/`). This folder then contains several additional files related to your sources and styles:

- an *.mss* style sheet containing CartoCSS style rules; a style sheet called *style.mss* is created by default, but it is recommended that you create multiple stylesheets to organize your styles by source map layer;
- an *.xml* file that contains information for rendering your tileset in Mapnik;
- a *.yml* file that contains metadata for publishing your tileset to Mapbox;
- a *.thumb.png* that provides a preview of your design; and
- any assets (fonts, textures, images, etc.) that you have embedded into your tile design.

Be sure to save your entire *.tm2* folder if you are moving between computers.

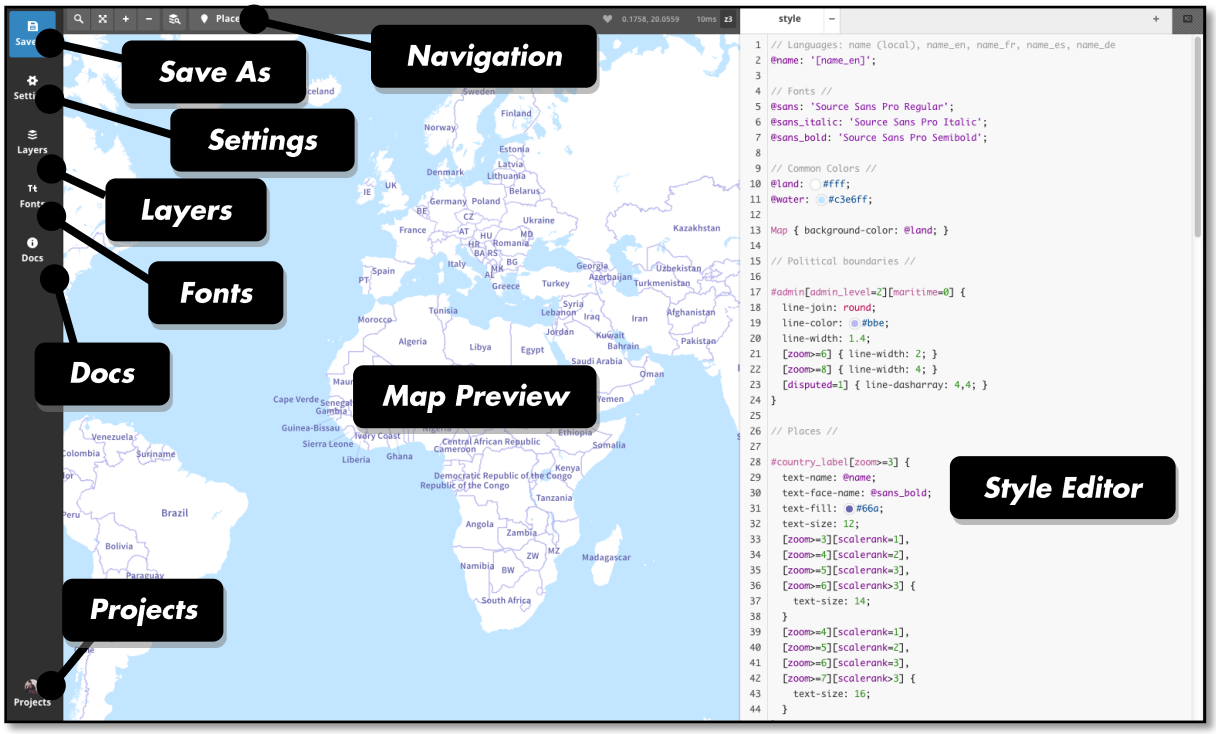


**Figure 2: The Mapbox Studio Classic Interface**

## b. Defining CartoCSS Style Rules

With you new project configured, you now can start adding the CartoCSS style rules you identified in your inspiration board worksheet. We **recommend** that you start your design at zoom level 0 and add conditional rules cascading to higher zoom levels, as smaller scales tend to have less complexity and therefore are more straightforward to design.

The following description provides example CartoCSS for styling a tileset based on the Lichtenstein inspiration board shown in **Figure 1**. The Lichtenstein example uses two of the three ***source*** datasets available in Mapbox Studio Classic: *Mapbox-Streets* and *Mapbox-Terrain*. As described in the *Primer on Mapbox Studio Classic*, you also can load and style custom source map layers in a variety of data formats: Shapefiles, GeoJSON, SQLite/PostGIS Databases, GeoTIFFS, VRTs, KML, GPX, and CSV.

The Lichtenstein example should be used for reference only as you learn how to declare CartoCSS rules in Mapbox Studio Classic. You will need to declare your own CartoCSS rules by form, color, type, and texture to match your inspiration board. Throughout the styling process, refer back to the *Primer on CartoCSS* and the CartoCSS documentation: https://github.com/mapbox/carto/blob/master/docs/latest.md.

The first step to styling your tileset is declaring ***global variables*** that describe the stylistic characteristics of your tileset that you will reuse across the source map layers and map scales. **Example 1** declares global variables for the color palette and type family used by the Lichtenstein example (**Lines 1-11**).

Next, call the `Map{}` function to create a new a map object for the tileset (**Example 1: 13-15**). Review the CartoCSS documentation to learn about which aspects of the tileset are manipulated through the `Map{}` function. Minimally, most CartoCSS-based tilesets use the `Map{}` function to style the `background-color` of the map (**Example 1: 14**).

```
1     //colors used a lot
2     @red: #BC243B;
3     @blue: #1D4E89;
4     @yellow: #FBD82B;
5     @black: #000;
6     @white: #fff;
7     @green: #019875;
8
9     @sans: 'Komika Hand Regular';
10    @sans_italic: 'Komika Hand Italic';
11    @sans_bold: 'Komika Hand Bold';
12
13    Map{
14          background-color: #fff;
15    }
16
```

**Example 1: Defining Global variables in Mapbox Studio and Styling the Map**

After declaring global variables and instantiating the map, you now can begin styling the source map layers. We recommend that you begin styling water features because they rarely are styled differently across zoom levels. To style the water, first call the `#water` unique identifier using the hashtag notation to identify the source map layer receiving the style rules (**Example 2**); the `#water` unique identifier is drawn from the *Mapbox-Streets* source, which you can view in the *Layers* tab.

As described in the *Primer on CartoCSS*, CartoCSS style rules are organized according to ten ***symbolizers***. **Example 2** declares fill and coastline style rules for water features using selectors included in the *Polygon* and *Line* symbolizers. For the water fill, the `opacity` is set to slightly transparent and the `polygon-fill` is set to global variable `@blue` color. The `polygon-gamma` also is set to `.2` to improve the anti-aliasing for the water features, preventing gaps in the water coloring across the tileset.

Notice that the water coastline is styled using a slightly different notation that the water fill. The line styles are declared using an ***attachment*** with double colon (`::`) notation. An attachment in CartoCSS is conceptually similar to a layer style in Photoshop. Thus, use of attachments allows you to style the same linework multiple times, building up sophisticated and highly stylized designs. In **Example 2**, the coast `line-color` is set to the same as the fill (although with no transparency) and three aspects of the coastline form are defined: `line-width`, `line-join`, and `line-cap`. **Figure 3** provides a visual preview of the style rules declared for the `#water` source layer.

```
1     #water {
2           opacity: .75;
3           polygon-fill: @blue;
4           polygon-gamma: .2;
5           ::line{
6                 line-color: @blue;
7                 line-width: 1.5;
8                 line-join: round;
9                 line-cap: round;
10          }
11    }
```

**Example 2: Styling #water in Mapbox Studio Classic**

**Figure 3: Styling the `#water` Source Layer**

Next, style country boundaries using the `#admin` layer included in the *Mapbox-Streets* source. If you inspect `#admin` in the *Layers* tab, you will see that the `admin_level` attribute can be used to apply styles to different administrative boundaries (e.g., countries, states, counties). **Example 3** declares conditional style rules that are applied to administrative boundaries equal to or below an `admin_level` of `2` (i.e., all countries and microstates). The `maritime` condition also is applied to avoid drawing boundaries over the previously styled coastline. You will need to study the attributes available in each source map layer in order to create a coherent and comprehensive visual style.

The conditional styling in **Example 3** then is used to define the color (using the global `@red` variable) and form of the country lines. The administration boundaries can appear jagged at times, so the `line-cap` and `line-joins` are set to `round` and a `line-smooth` element is applied. Finally, a pair of conditional styles is applied by `zoom` level to make the country boundaries thicker as the user zooms into the map, growing from `1` to `1.15` at `zoom 3` and then from `1.15` to `1.5` at `zoom 4`. **Figure 4** shows the styled administration unites along with the styled water.

```
#admin[maritime=0][admin_level<=2]{
      line-color: @red;
      line-cap: round;
      line-join: round;
      line-smooth: .75;
      [zoom>=3]{
            line-width: 1.15;
      }
      [zoom>=4]{
            line-width: 1.5;
      }
}
```

**Example 3: Styling country lines in Mapbox Studio Classic**

**Figure 4: Styling the Country Boundaries**

Now that you have explored how to style lines and polygons, it is time to add and style type. The *Mapbox-Streets* source layer separates geometry and labels into different layers. This separation makes it much easier to manage your style rules and avoids having to use attachments for labeling. Two symbolizers are available for adding and styling type: *Text* and *Shield*. The *Text* symbolizer is treated here for labeling polygons; details about the *Shield* symbolizer are provided towards the end of this section when labeling points.

**Example 4** uses selectors from the *Text* symbolizer to add country labels. The *Text* symbolizer requires two style rules: (1) the `text-name`, an attribute in the source map layer, and (2) the `text-face-name`, or typeface used to render the text in the tileset. Mapbox provides labels in multiple languages and stores them as different attributes in the label source layer (**Figure 5**); you therefore **must** use conditional attribute notation to select the desired `text-name` (`[name-end]` in **Example 4**).

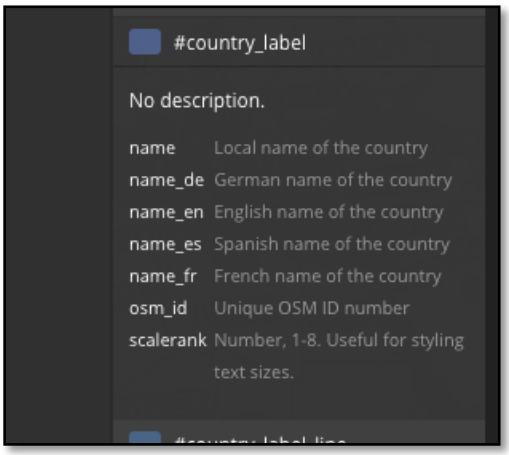


**Figure 5: Available `text-name` options in the `#country_label` source layer.**

Remember the global variables for type declared in **Example 1**? **Example 4** uses the `@sans_bold` global variable to set the `text-face-name` for country labels. While not necessary, you probably want to customize your `text-fill` and `text-size`. **Example 4** uses `@white` for the `text-fill` and `@black` for the `text-halo-fill`, also giving the halo a thick `text-halo-radius` of `2` to make it look like type from a comic book. This process is conceptually similar to using the *Appearance panel* in Illustrator to add a halo to text. Advanced text style rules then are defined to put text on an angle and to customize the spacing between lines and characters (**Example 4: 8-11**), again applied to make the text mirror that in a comic book. Experiment with the various text functions to place and style your labels how you want them.

A common concern with adding country labels is changing the text size of various countries in order to not make the map look too busy. Just like Natural Earth, the source layers provided by Mapbox have a `scalerank` attribute to enable conditional styling based on the intellectual hierarchy of the mapped features. **Example 4** (**Lines 12-18**) uses the `scalerank` attribute to adjust the `text-size` of country labels (violating a cartographic convention, but for the sake of producing the desired visual style!), and then uses additional conditional styling to increase each text size level as the user zooms into the map (**Lines 19-30**).

```
#country_label[scalerank<=8]{
      text-name: [name_en];
      text-face-name: @sans_bold;
      text-size: 12;
      text-fill: @white;
      text-halo-fill: @black;
      text-halo-radius: 2;
      text-orientation: 5;
      text-character-spacing: -1;
      text-wrap-width: 80;
      text-line-spacing: -10;
      [scalerank<=1]{
            text-size: 24;
      } [scalerank=2]{
            text-size: 18;
      } [scalerank=3]{
            text-size: 14;
      }
      [zoom>=4]{
            [scalerank<=1]{
                  text-size: 30;
            } [scalerank=2]{
                  text-size: 24;
            } [scalerank=3]{
                  text-size: 18;
            }
                  text-size: 16;
            }
      }

#country_label_line {
      line-width: 3;
}
```

**Example 4: Styling country names in Mapbox Studio Classic**

Notice that **Example 4** also declares a style rule for the `#country_label_line` source layer. Declaring a `line-width` for the `#country_label_line` will add a leader line to any feature that is labeled in the water during automated label placement.

Now that you are comfortable using the *Text* symbolizer, try labeling the oceans using the `#marine_label` source layer. Not every source layer provided by Mapbox has an attribute like `scalerank`, but you always can use another attribute for conditional style or adjust the styling based on the zoom level. **Figure 6** shows the map with country and ocean labels added.


**Figure 6: Country Labels and Ocean Labels**

While **Figure 6** represents a good amount of progress, the tileset still does not have the cartoon textures made famous by Lichtenstein. Next, use *Mapbox-Terrain* to add a texture component to your visual style. *Mapbox-Terrain* includes four useful source layers for adding textures: (1) `#hillshade`, (2) `#contours`, (3) `#landcover`, and (4) `#landuse`. **Example 5** uses the `landcover` and `#landuse` source layers included in the *Mapbox-Terrain* dataset to add visual complexity to the tileset.

Specifically, **Example 5** uses conditional styling on the `class` attribute to add a colored pattern to five different land cover or land use types: (1) `scrub`, (2) `snow`, (3) `grass`, (4) `park`, and (5) `wood`. Each land cover `class` first is given a `polygon-fill` color and—in the case of `scrub`, `grass`, `park`, and `wood`, which all use `@green`—and `polygon-opacity` value. A texture then is added to each `class` (excepting `grass`) using the `polygon-pattern-file` selector. The `polygon-pattern-file` is unique in that it loads an *.svg* file called `halftone5.svg` located in the *.tm2/img/* folder using the `url()` function. The texture included in `halftone5.svg` is repeated within the polygon boundaries; thus, use Illustrator to properly align your pattern for repeated tessellation. **Importantly**, the same `url()` function can be used to embed custom fonts and images into your tileset as well!

Finally, the `scrub` and `snow` layers declare rules for the `polygon-pattern-comp-op` selector. The `comp-op` is short for compositing operators from Photoshop tools, or the blending modes introduced in Lab #2. Mapbox enables you to blend two source layers together using many of the Photoshop compositing operators. Each operator performs a specific type of effect; explore the effects to determine which blending mode best suits your tileset style. Refer to the Mapbox documentation for more information about compositing operators: https://www.mapbox.com/guides/compositing-reference/. **Figure 7** shows your tileset after adding textures.

```
#landcover[class='scrub']{
      polygon-fill: @green;
      polygon-opacity: .35;
      polygon-pattern-file: url("img/halftone5.svg");
      polygon-pattern-comp-op: soft-light;
}

#landcover[class='snow']{
      polygon-fill: @blue;
      polygon-opacity: .45;
      polygon-pattern-file: url("img/halftone5.svg");
      polygon-pattern-comp-op: overlay;
}

#landcover[class='grass']{
      polygon-fill: @green;
      polygon-opacity: .5;
}

#landuse[class='park'],[class='wood']{
      polygon-fill: @green;
      polygon-pattern-file: url("img/halftone5.svg");
}
```

**Code Bank 5: Styling landcover in Mapbox Studio Classic**

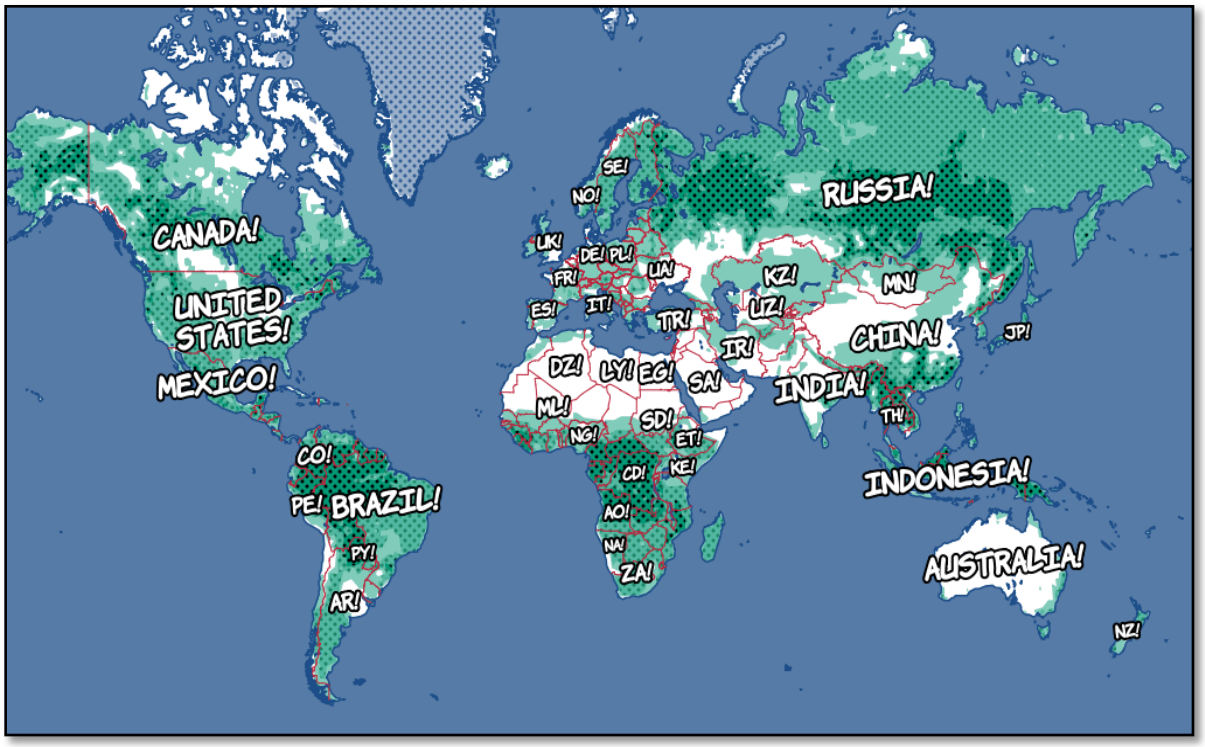


**Figure 7: Adding Textures to `#landcover` and `#landuse` by `class`.**

The tileset design is really starting to come together, at least at the lower zoom levels. Next, refine your design by adding in layers and styles at higher zoom levels, such as cities, roads, buildings, etc. **Example 6** demonstrates how to add symbols and labels for cities using the `#place_label` source layer and the *Shield* symbolizer. Conditional styling is used to add symbols and labels to cities with a `scalerank` less than `2` (showing only capitals) and at a `zoom` level below `13` (higher zooms are beyond the scale of a city).

Selectors included in the *Shield* symbolizer build upon the *Text* symbolizer in that they include style rules for both the label and a map feature being labeled. Specifically, the *Shield* symbolizer combines an image file used as a point marker with the text label placed next to it. In **Example 6**, the point symbol is added using the `shield-file` selector and the `url()` function. Thus, you have complete control over your symbols using the *Shield* symbolizer, as you can manipulate these graphics in Illustrator and export as an *.svg*! The `shield-name` and `shield-face-name` and are similar to `text-name` and `text-face-name` from the *Text* symbolizer, defining the type style for the labels.

```
#place_label[type='city'][scalerank<2][zoom<=13]{
      shield-file: url("img/dot.svg");
      shield-name: "[name_en]";
      shield-face-name: @sans;
      shield-unlock-image: true;
      shield-text-dx: 3;
      shield-text-dy: 3;
      shield-transform: scale(.5,.5);
      shield-placement: point;
            shield-size: 12;
      shield-fill: @white;
      shield-halo-fill: @black;
      shield-halo-radius: 1.25;
      [ldir = 'N'],[ldir = 'E']{
            shield-text-dx: 3;
            shield-text-dy: 3;
      }
      [ldir = 'W']{
            shield-text-dx: -3;
            shield-text-dy: 3;
      }
      [ldir = 'S']{
            shield-text-dx: 3;
            shield-text-dy: -3;
      }
      [zoom>=6]{
            shield-size: 16;
      }
}
```

**Example 6: Adding Symbols and Labels to Cities using the *Shield* Symbolizer**

In Illustrator or Photoshop, you can manually position each label based on cartographic conventions of type placement. In CartoCSS, you can use conditional styling to unlock (`shield-unlock-image`) the default positioning for all labels or a group of labels and then reposition the labels using the `shield-text-dx` and `shield-text-dy` selectors (**Example 6: Lines 5-7**). This placement is relative to the `ldir`, or label direction based on the preferred position of point labels. It is important to note that the *Text* symbolizer has a similar set of selectors, but is used less commonly due to the lack of a relationship between symbol and label.

The remainder of **Example 6** styles the labels placed using the *Shield* symbolizer (**Lines 8-13**) and then uses conditional styling to adjust the repositioning based on the `ldir` used for the label during automated placement (**Lines 14-25**). **Figure 8** shows the tileset with city labels!

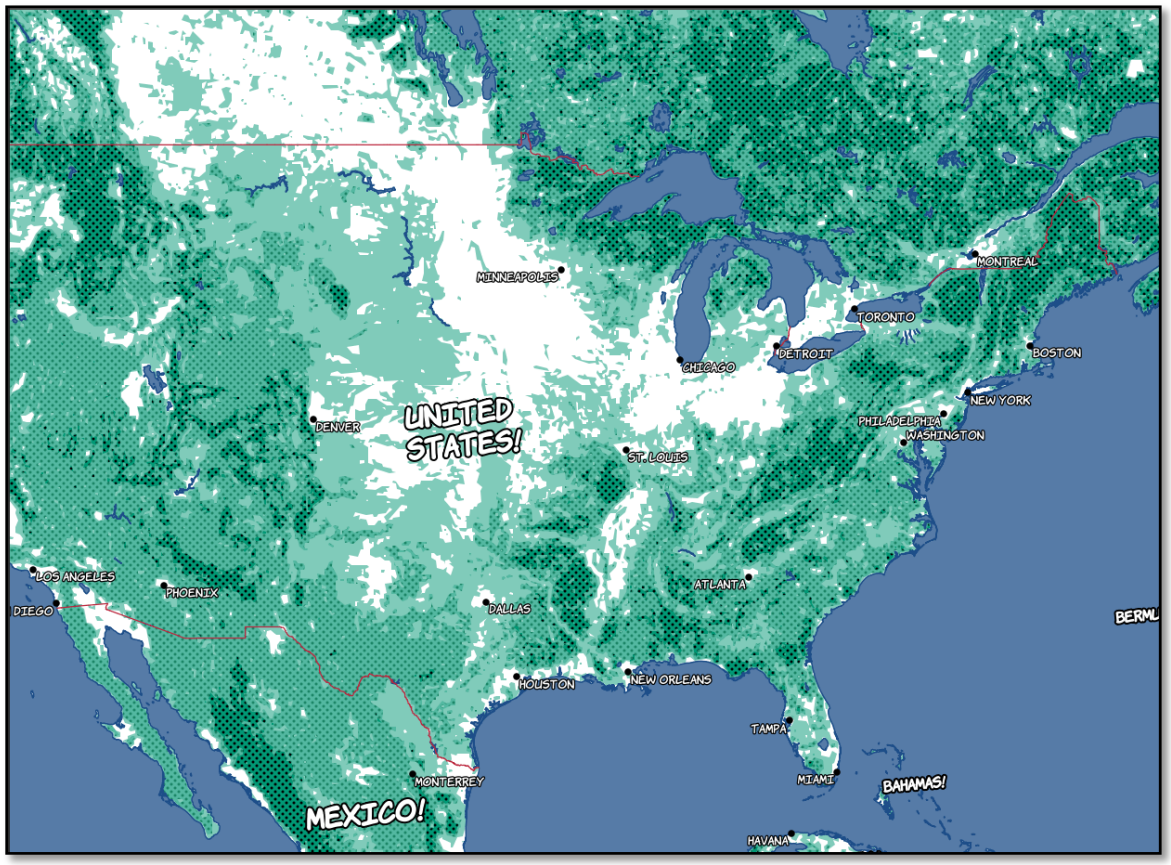


**Figure 8: Shields and shield-text added to major US cities**

The final step of this tutorial is adding road styles to improve the tileset design at large zoom levels. When designing at high zoom levels—or large cartographic scales—it is recommended to use the *Places view* to see how your design works around the world. **Example 7** uses attachments to create an outer, thick black line and an inner, thin white line for the roads. Such a use of attachments mimics the halo effect used for country and water labels, but through the *Line* symbolizer instead of the *Text* symbolizer. The `#road` source layer included the *Mapbox-Streets* dataset has a ton of attributes based on the OpenStreetMap schema. Take advantage of these attributes in your design to apply different styles to different categories of roads.

```
#road[zoom>=6] {
      line-join: round;
      line-cap: round;
      ::case {
            line-width: 3;
            line-color: @black;
      }
      ::fill {
            line-width: 1.5;
            line-color: @white;
      }
      [zoom>=14]{
            ::case {
                  line-width: 4;
                  line-color: @black;
            }
            ::fill {
                  line-width: 2;
                  line-color: @white;
            }
      }
}
```

**Code Bank 7: Styling Roads using Attachments**

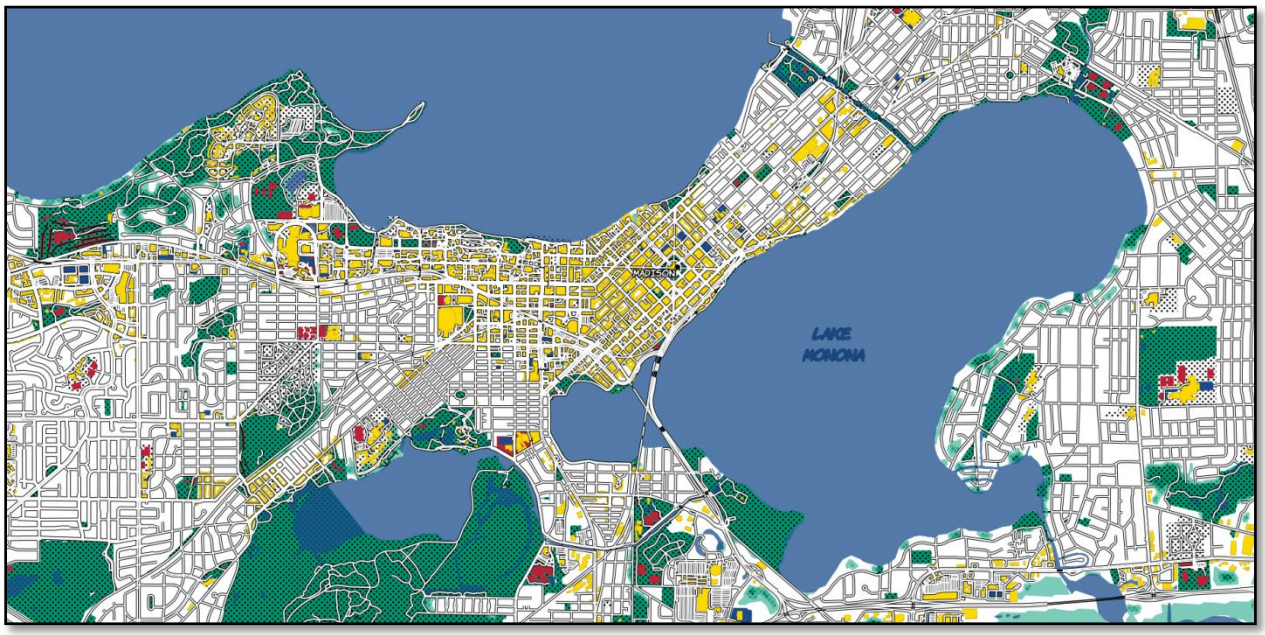


**Figure 9: Roads Added at a High Zoom Level of Madison, WI**

Congrats! Look at what you've accomplished in just a few lines of CartoCSS (**Figure 9**)!

# 4. Creating and Publishing a Slippy Map

## a. Uploading Your Tileset to Mapbox Editor

Congratulations, you have designed a custom tileset with a unique aesthetic style! Ooh, aah, nice!

Once you are finished with your design, save your tileset in Mapbox Studio Classic and upload it to the Mapbox servers. As introduced in the *Primer on Mapbox Studio Classic*, Mapbox has different pricing structures for tile rendering (free with Mapbox Studio Classic) versus tile serving (incremental cost by number of impressions or views). While there are open tile serving options—such as TileStache—you will use your Mapbox account to serve tiles from their servers. The ***Starter*** account level is free, and enables up to 50,000 map views per month, which should be sufficient for Lab #3 (unless your tiles go viral, which is a great problem to have!).

To upload your tileset, save your project a final time and open the *Settings* tab. After filing out the metadata form (*Name*, *Description*, etc.), select the blue *Upload to Mapbox* button. Once the tileset upload is complete, navigate to mapbox.com using a browser and log into your Mapbox account. On the Mapbox website, you will see three tabs next to your account name: *Projects*, *Styles*, and *Data* (**Figure 10**). Click on the *Styles* tab to view your uploaded tilesets; if it is not yet listed, click *Uploads* to see how much longer the upload will take. Once your tileset is available in the *Styles* tab, select it and create a *New Project* using this tileset style (**Figure 11**). The new project will then open in Mapbox Editor.

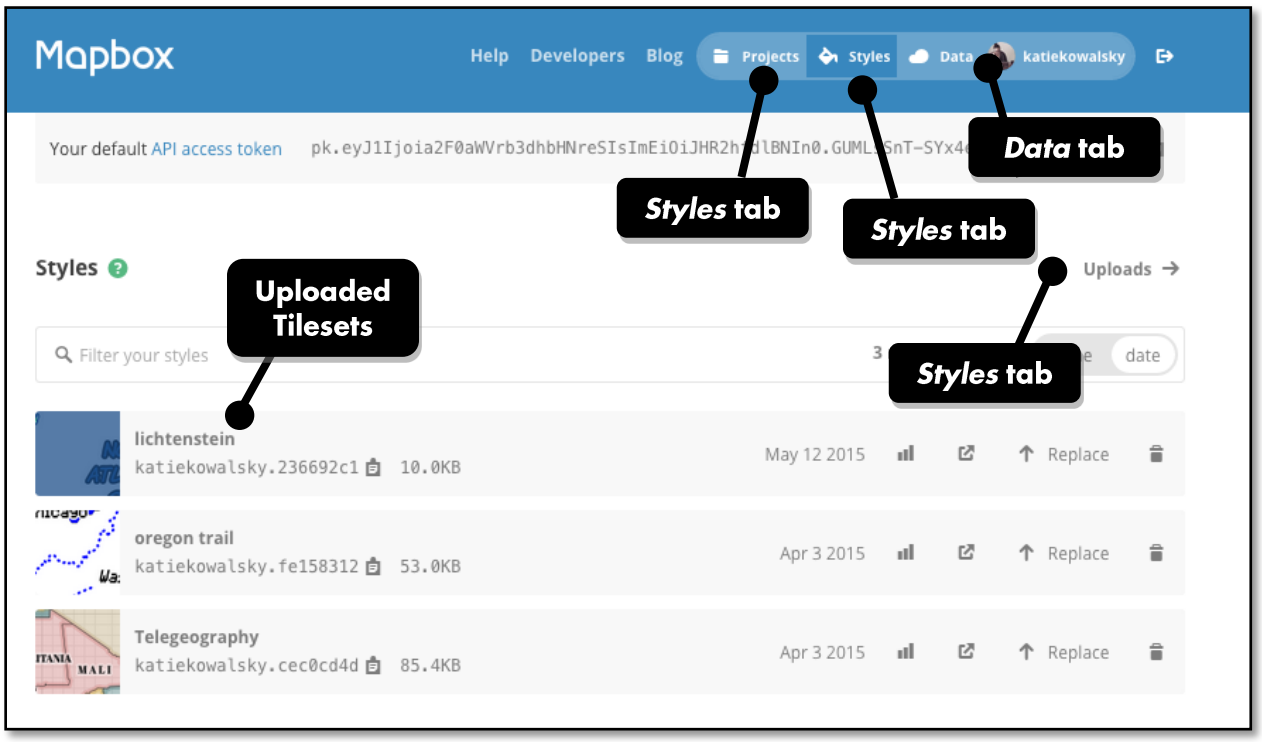


**Figure 10: Styles Page on Mapbox.com**

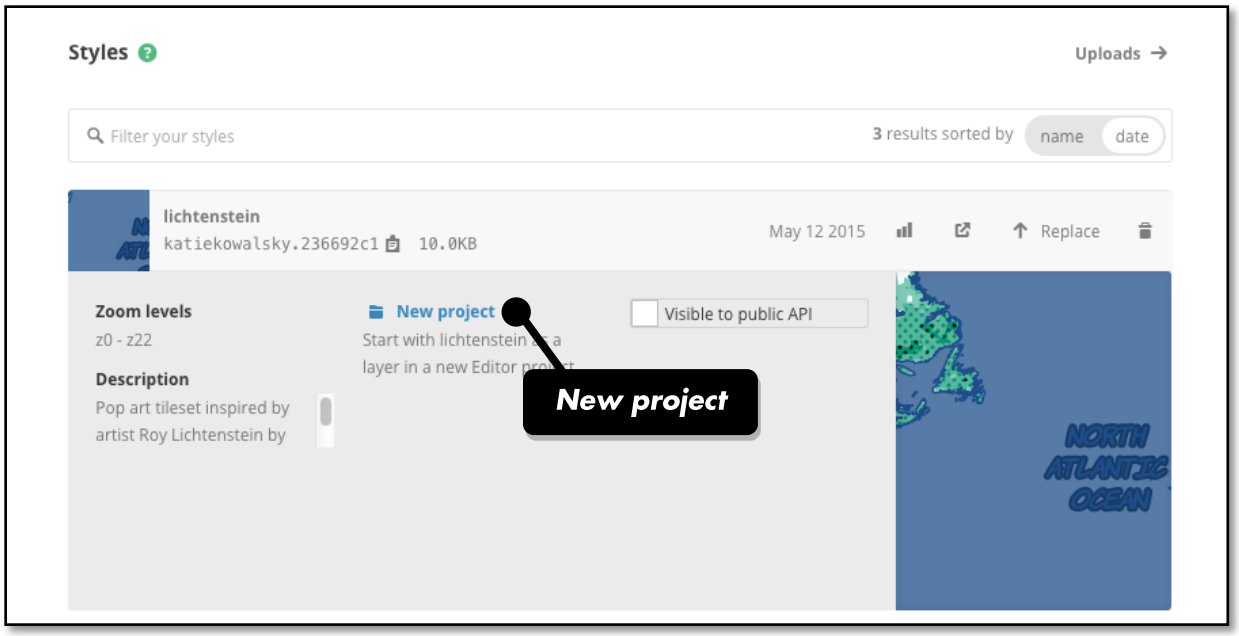


**Figure 11: Creating a New Project on Mapbox.com (Clicked on Uploaded Style)**

## b. Adding Your POIs with Mapbox Editor

Mapbox Editor is a browser-based application that supports the creation of slippy web map mashups using a point-and-click user interface. The Mapbox Editor interface writes Mapbox.js JavaScript code for you as you parameterize your slippy map. Mapbox.js is built atop the Leaflet.js mapping library that you will be introduced to in Geography 575!

Like the Mapbox website, Mapbox Editor includes *Style*, *Data*, and *Project* tabs. Because you created a new project from your tileset on the Mapbox website, your tileset is selected under the *Style* tab by default. If you want to edit your tileset at this stage, you will need to do so in Mapbox Studio Classic and re-upload your tileset to the Mapbox server.

Open the *Project* tab and give your slippy map a relevant *Map ID* or title. Through the *Project* tab, you can share you slippy map in two ways: (1) as a direct link that opens the slippy map in a browser tab, and (2) as an embedded `iframe` element within a larger website. **For Lab #3, you will submit a direct link to the Dropbox**. You will learn how to embed your slippy map into a responsive website in Lab #4.

You will add markers to represent your POIs through the *Data* tab (Figure 12). Mapbox Editor allows you to annotate the map with custom point, line, and polygon features using the three menu items under the *Data* tab. You also can import point, line, and polygon layers that are in the CSV, GeoJSON, GPX, and KML formats. Each feature or layer is added individually, and has a wizard interface to walk through its import and styling.

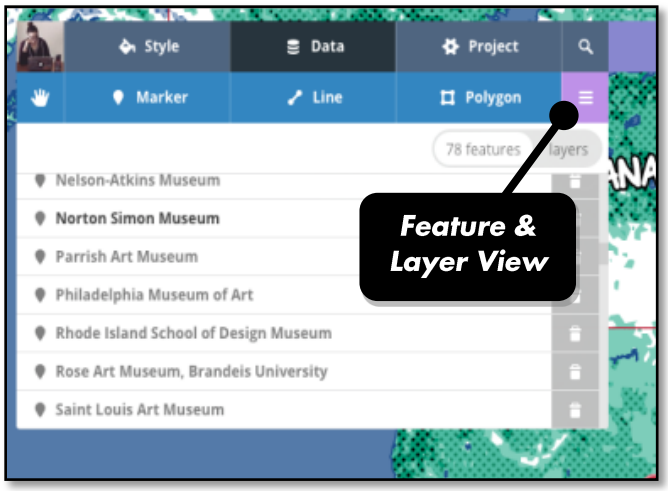


**Figure 12: The *Style*, *Data*, and *Project* Tabs in Mapbox Studio.** Here, the *Data* tab is activated, allowing for custom annotation and inspection of individual features and layers added atop the map.

To add your POIs, click the *manually import* link in the *Feature View* of the *Data* tab and navigate to your *.csv* file containing your POI information. Next, walk through the import wizard that helps you configure your markers. Importing includes four steps:

1. **Choose your marker title:** The import wizard allows you to choose any column in your *.csv* file that is used for the heading of your marker, other than latitude & longitude. In **Figure 13a**, the *location* attribute is selected.
2. **Choose your description.** Similarly, the importer wizard allows you to choose any column for use in the popup window description. In **Figure 13b**, the *painting* attribute is selected.
3. **Style the Marker:** Next, select the marker color and size. While you are constrained somewhat by Mapbox Editor in your design, try to select a color and size that matches your basemap style (or at least one that does not undermine your basemap style). As stated in lecture, if you plan on supporting mobile viewing, a larger marker size is recommended.
4. **Add an Icon to the Symbol:** Finally, select an icon to represent your POIs. The icons are drawn from the Mapbox Maki library, which includes 114 symbols with simple silhouettes. Return to the lecture notes on semiotics to choose a pictorial or associative icon that signifies the interpretant and referent as unambiguously as possible.

Select *Finish Importing* once you have walked through the import wizard. As discussed in lecture, you may want to signify higher level information about your POIs, meaning you will want to differentially style your POIs using different colors or different icons. There are two strategies for differential symbol styling: (1) You can divide your *.csv* file into multiple files based on the higher level categorization and import each file separate; or (2) you can modify the default marker style for each marker one-at-a-time by selecting the marker on the map and changing the style through the *Data* tab.

Congratulations, you have created a slippy map with a custom basemap style and overlaid set of POIs (**Figure 14**)! Copy the direct link provided in *Project→Share* and paste this link into the Learn@UW Dropbox.

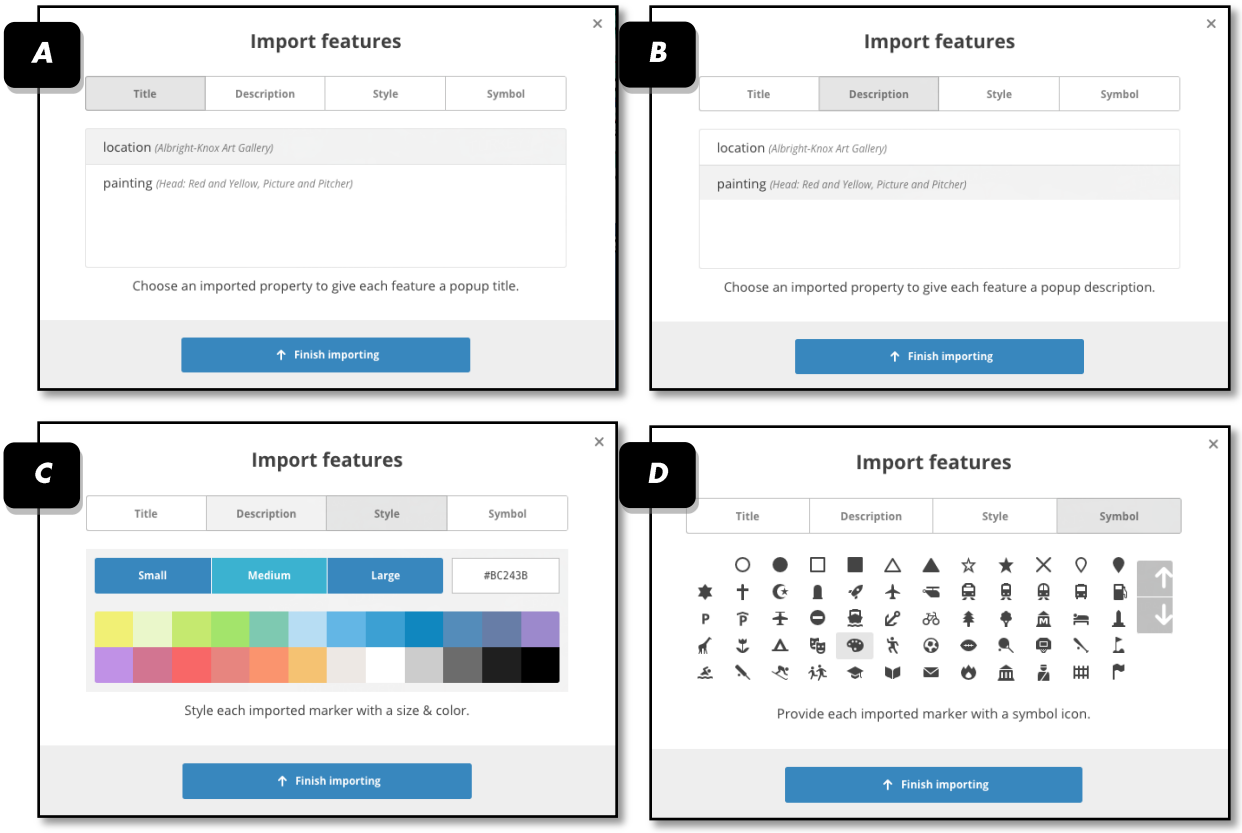


**Figure 13: Importing Your *.csv* File.**

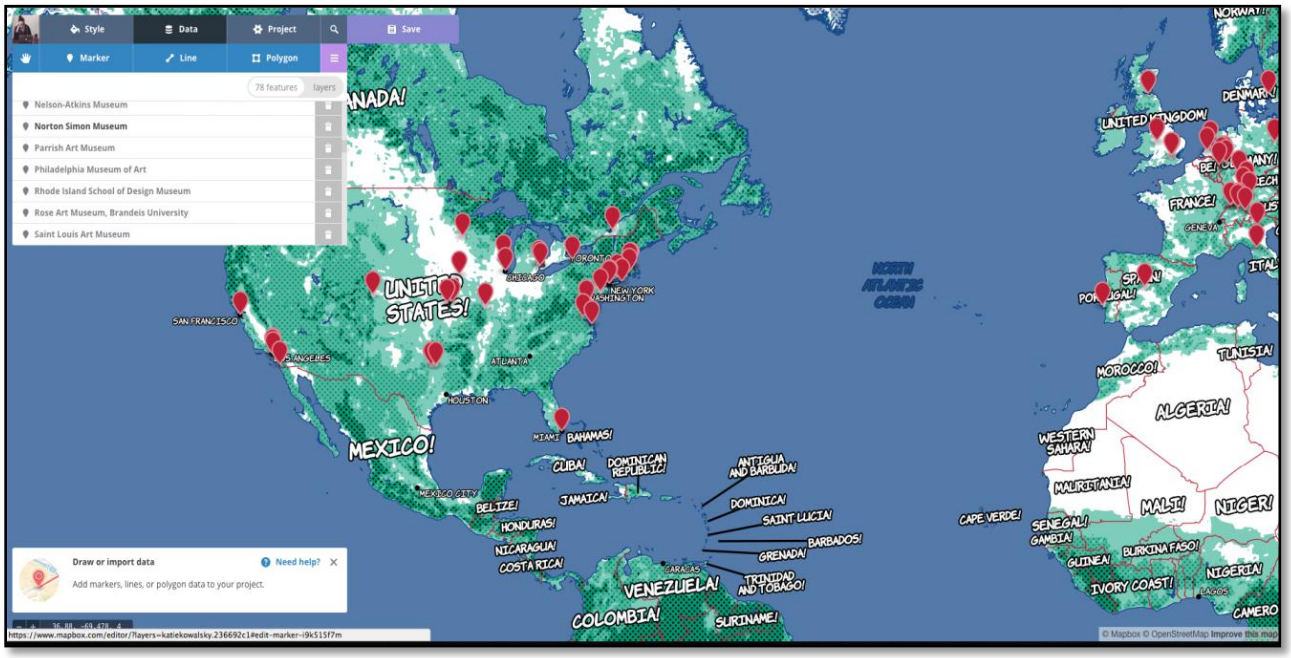


**Figure 14: Your Slippy Map with POI markers.**

## Evaluation Rubric: Slippy Map Iconicity Challenge (40pts)

## Inspiration Board + Worksheet Check-In (2)

**2pts:** The inspiration board has an interesting, coherent style and the worksheet properly reflects the decomposition of this style into form, color, type, and texture CartoCSS rules.

**1pt**: The inspiration board has an interesting, coherent style, but one or several aspects of the style are missing from the worksheet.

**0pt:** The submitted icon library did not meet the expectations of the assigned challenge, or was not submitted on time.

## POIs Check-In (2)

**2pts:** You have assembled your *.csv* file and they are appropriate for your inspiration board.

**1pt:** You have assembled your *.csv* file, but you may several POIs do not fit with your inspiration board.

**0pts:** The submitted POIs did not meet the expectations of the assigned challenge, or was not submitted on time.

## Slippy Map (21): Submit Link to Learn@UW

**19-21pts:** The slippy map is attractive, informative, and engaging. You have successfully replicated the form, color, type, and texture of your inspiration board. The tileset has a consistent style across zoom levels and makes use of numerous source map layers to build up this style. Your tileset is loading properly and the POI markers reinforce the theme of your map. What a stunning example of multiscale map design!

**16-18pts:** The slippy map overall is successful, but there are a few aspects of the design that can be improved. One or several of the design components (form, color, type, or texture) does not match your inspiration board, or you could have considered one or several of these components further in your design. There are one or several jarring zoom level transitions in which your map design is inconsistent. You could have made use of one or more additional source map layers discussed in the lab to add complexity to your map style. Your tileset is loading properly and the POI markers reinforce the theme of your map.

**13-15pts:** The slippy map only marginally improves upon an existing style template provided by Mapbox, or does not appear to be uniquely styled at all. The inspiration board style did not translate into your design of form, color, type, and texture in the tileset. Multiscale design was not considered across zoom levels, or only one or two conditional transitions are provided. You did not style many of the source map layers discussed in the lab and included in your worksheet. Your tileset is loading properly, but the POI markers distract from the overall style.

**Below 12pts:** The submitted slippy map mashup did not meet the expectations of the assigned challenge in multiple and critical ways. Please speak with Rob and Chelsea about strategies to improve the design