Print Preview Feature for Mapbox Studio

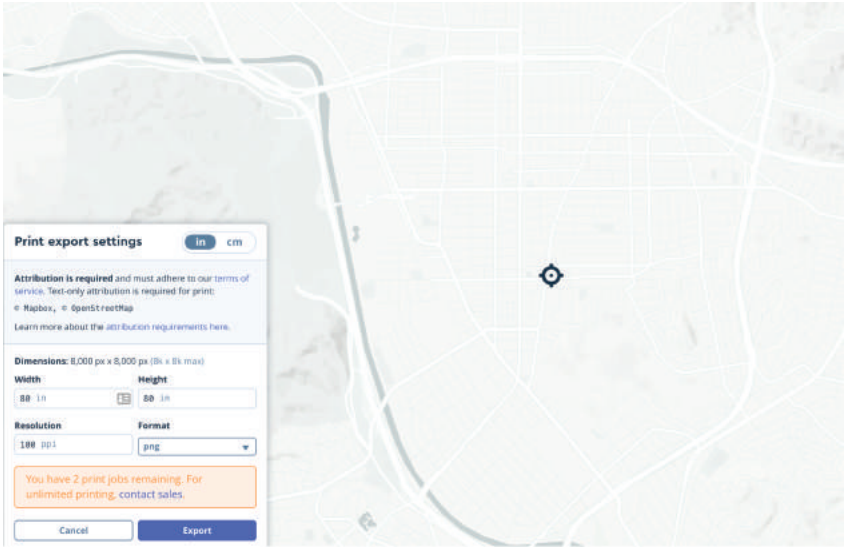
Madison Draper | 2018

Observation

Mapbox Studio has feature to print a high resolution map. A user can pan and zoom to a location on the map to create a center point for the image. When a user changes the dimensions of the map print, the map preview does not change. This leads users to believe they will receive this image preview in their given dimensions.

Problem

When a user changes the dimensions to something that doesn't align with the amount of pixels allocated to the browser window, the map is not distorted to fit the dimensions. The map zoom level is changed based off the dimensions, which changes which map features are present and the level of detail. Pay-as-you-go, commercial and enterprise customers attempted to print a high resolution map from Mapbox Studio and ran into this issue.



Voice of the Customer

How large of a map can I print? Is it whatever you can see on your screen or is there a tool where I can drag and size the dimension?

Here attached the screen as it looked like before launching the print : [then after exporting] half of the names were cut out, is it my browser or my cache? It is a shame because I only have 5 prints and this made me try things... Without result :(...

Scope of Project

This feature was intended to be implemented in two places:

1. Print a Map, an internal testing application
2. Mapbox Studio, the public facing application

There has been a long-standing open issue about this print feature. Summarizing pain points and discussion into actionable items was critical to learning and fully understanding the issue before determining the best solution.

With a clearly defined problem & potential solution set in hand, the next step involved finding a mentor to guide through the codebase and learn React and Mapbox's internal development tools.

The research methodology for this involved qualitative research:

- Read Support tickets and look for themes about printing
- Test the feature myself and see if expectations match
- Talk to PMs, Sales and Customer Success to gauge the value of this feature to see if it's should be an engineering priority
- Talk to engineers to figure out the current implementation, ideas they've witested, and current constraints

The design below breaks down how the UI and unpacks the layers of the preview to avoid any inaccurate previews or other issues, such as container overflow or image clipping.

Uncertainties

- User confusion around how to why the viewport preview doesn't update
- User confusion around how to make the print preview match the viewport preview
- User confusion around the size of the print preview

Goals

- Provide a tool that will help users print maps of expected size
- Decrease Support tickets about print issues
- Increase usability of our tools
- Increase use of map prints

User Profile

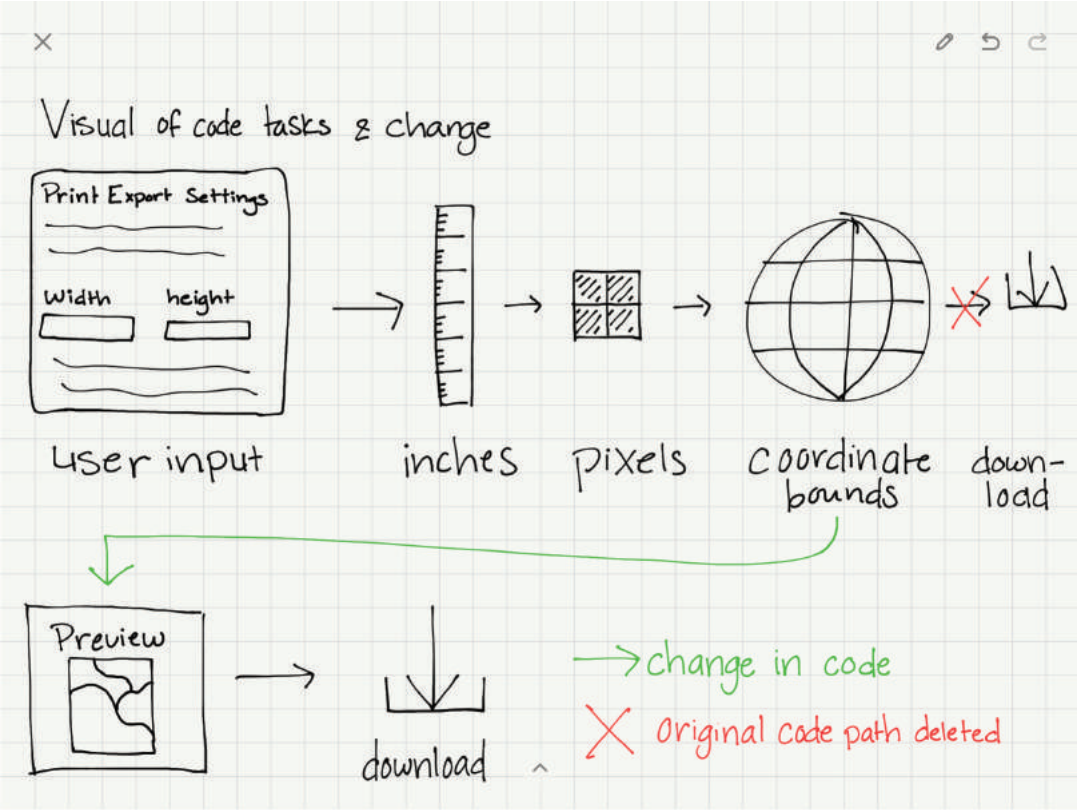
A user is someone who meets the following criteria:

- User of Mapbox Studio
- Attempts to print a map

We are hoping this feature addition is intuitive so that if a user who is new to this feature activates this feature, they will see the print preview and print match are matching and it meets their expectations. Because this is only a minor change and not a major change, no external testing was performed on users.

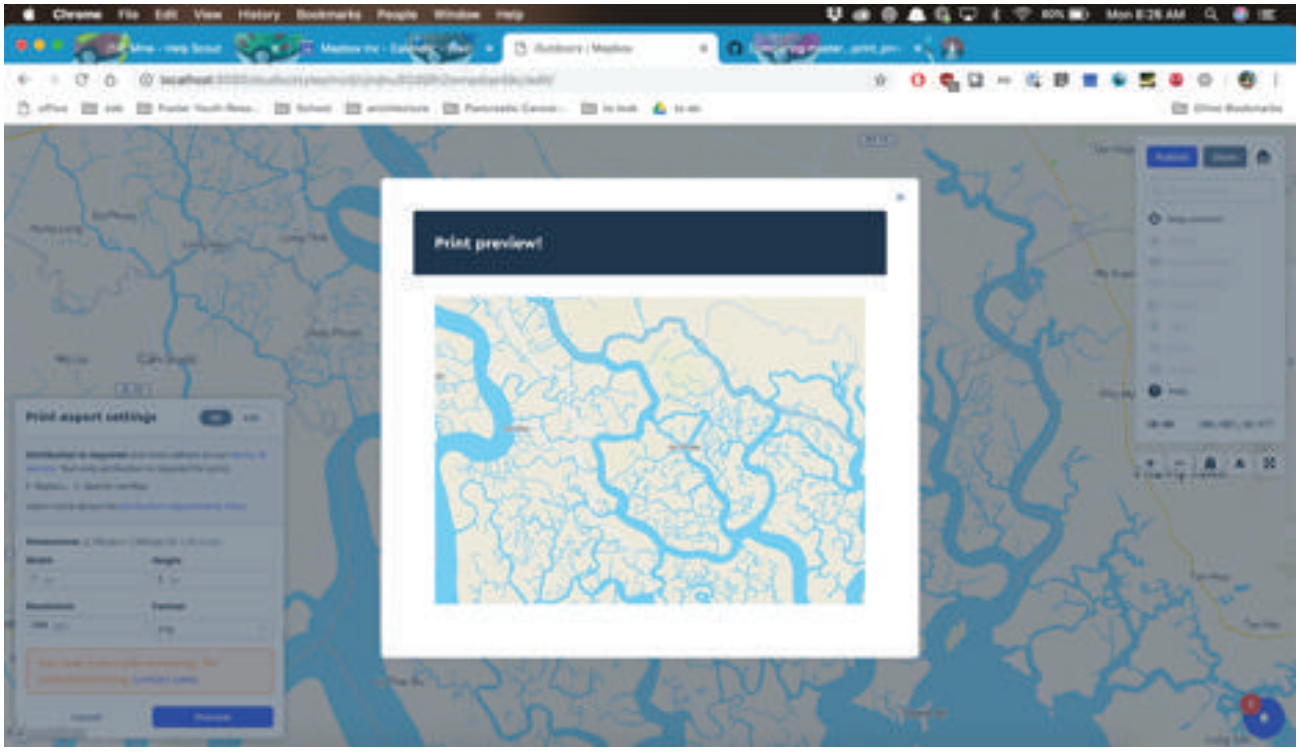
User Journey

Since the core workflow of the print still stands, the changes made are depicted in the image below. Here we are forgoing an immediate download of something the user may not want into a preview for a better expectations match.



Results

- A print preview appears when printing from the Print a Map standalone application
- The pull request is mergeable, but has been withheld for other options
- There is an open pull request with this feature in the app's repository




```
preview = () => {
  const { stylesheet, mapState } = this.props;
  const { tempPrintFormValues } = this.state;
  const previewResolutionValue = tempPrintFormValues.set('resolution', 90);

  generateMapAsImage(stylesheet.toJS(), mapState, previewResolutionValue)
    .then(blob => {
      var url = window.URL.createObjectURL(blob);
      this.setState({ previewUrl: url });
    })
    .catch(err => {
      throw new Error(err);
    });
};

togglePreviewModal = () => {
  this.setState(state => ({ previewModal: !state.previewModal }));
};
```

```
{this.state.previewModal && (
  <PrintPreviewModal
    print={this.print}
    onExit={this.togglePreviewModal}
    previewUrlForModal={this.state.previewUrl}
  />
)}
```