

# MapBox & TileMill

An open-source-ish alternative to MapKit

Flip Sasser  
@flipsasser

inthebackforty.com  
@InTheBackForty

# I'm Flip

I'm just learning about MapBox but it's kinda cool but kinda not so let me explain

# The Fit



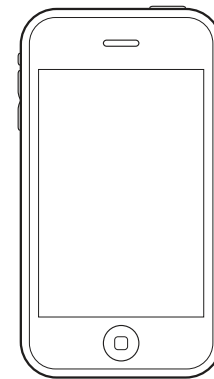
TileMill

(makes tiles)



MapBox

(makes maps)



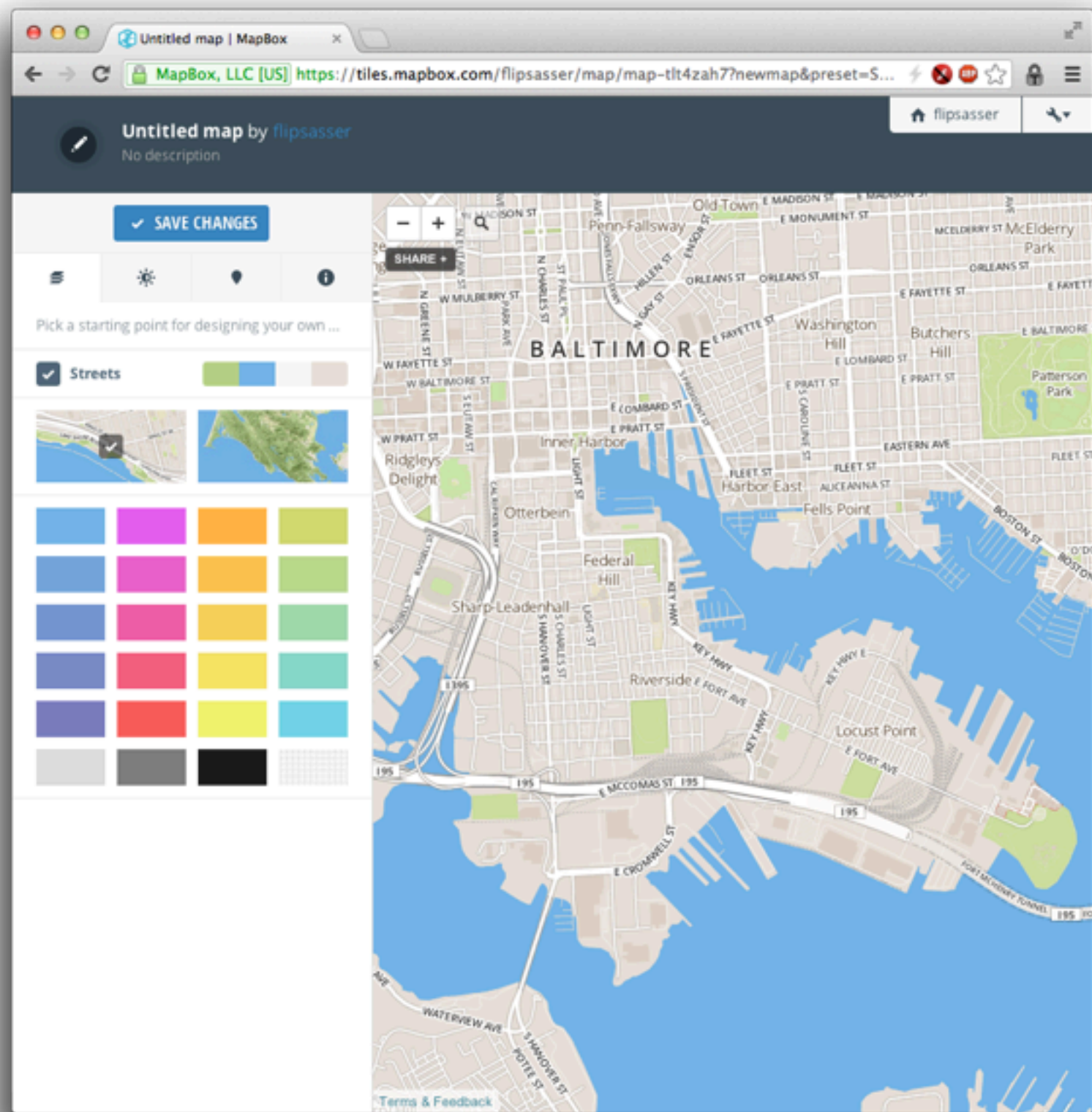
iOS

(renders maps)



# Chapter 1

**MapBox serves your tiles  
(if you ever get them)**



# Creates a tile API endpoint for your map

```
<iframe width='500' height='300' frameBorder='0'  
src='http://a.tiles.mapbox.com/v3/flipsasser.map-  
tlt4zah7.html#14/39.274300000000004/-76.602'></iframe>
```

# This is a \*pay\* service

But if you can get TileMill to export, you get a  
free, locally cached tileset!



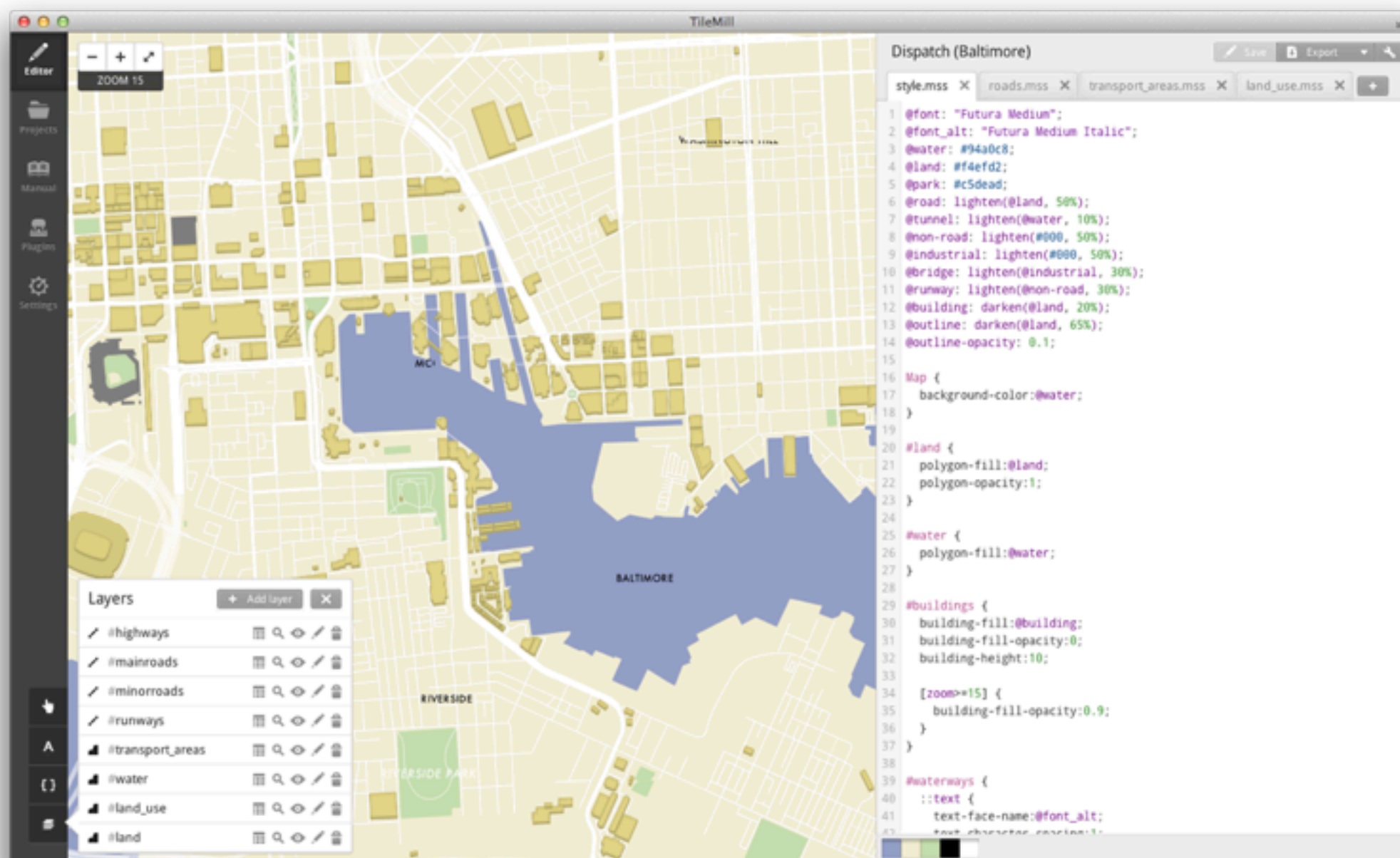
# Chapter 2

TileMill in all its misery glory misery



# Node.js-backed HTML UI

It's “cross platform”



# Draws tile layers from various data sources

**What data sources?**

- **Open Street Maps**
- **Open ... Street Maps**
- **Open, well, Street Maps**

# Ways to get OSM data

Because there's a lot of it

# The firehose

[planet.openstreetmap.org/](https://planet.openstreetmap.org/)  
25GB of data

# Landmasses (landmassi?)

[download.geofabrik.de/openstreetmap/](https://download.geofabrik.de/openstreetmap/)

Large maps or maps of specific territories

# Coastlines

[openstreetmapdata.com/data/land-polygons](https://openstreetmapdata.com/data/land-polygons)

These make a \*huge\* difference



Coastlines w/OSM base data



\*that's Baltimore, yo!



Coastlines w/detailed data\*

# Streets, railways, and buildings

[metro.teczno.com/](https://metro.teczno.com/)

Look for your specific metro area

Path layers

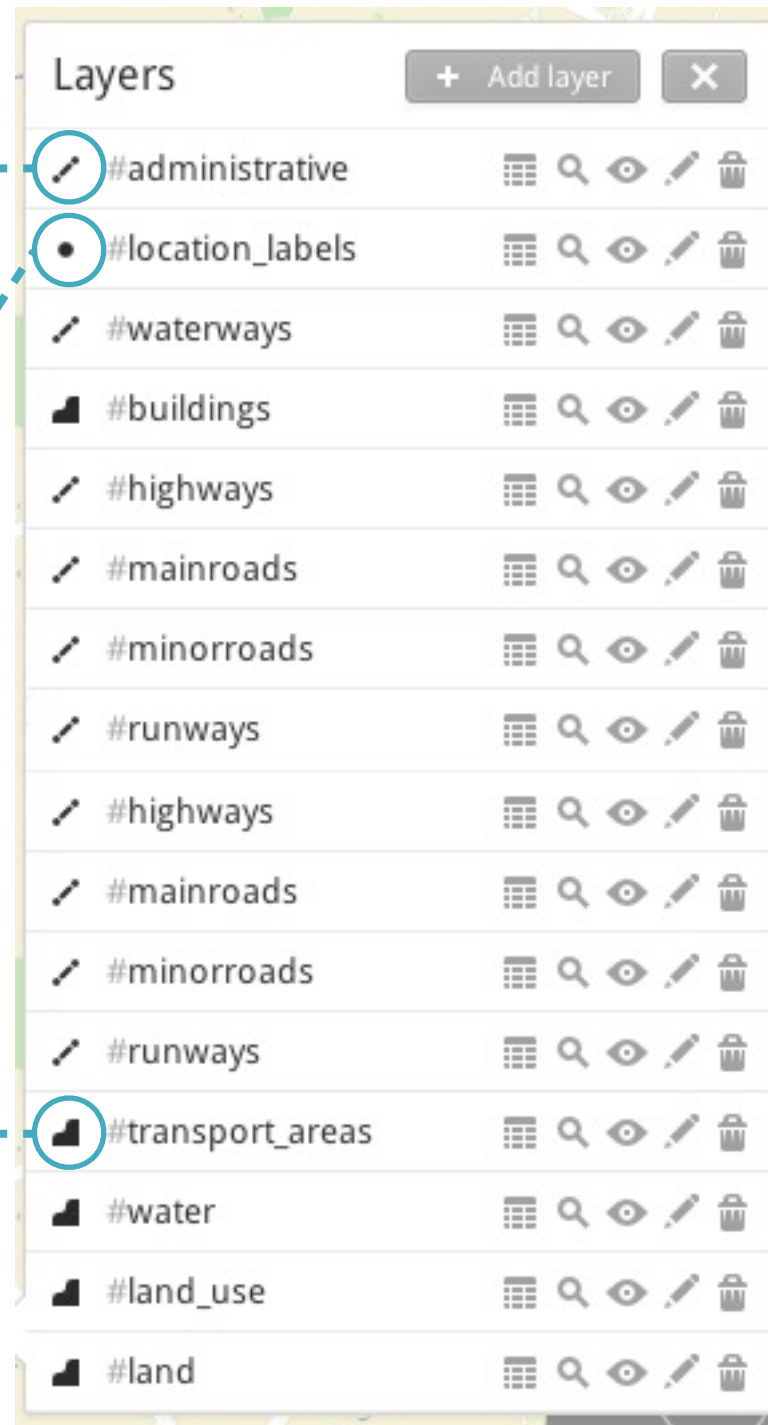
Style as lines

Point layers

Style as markers

Polygon layers

Style as shapes



Put 'em together

# CartoCSS

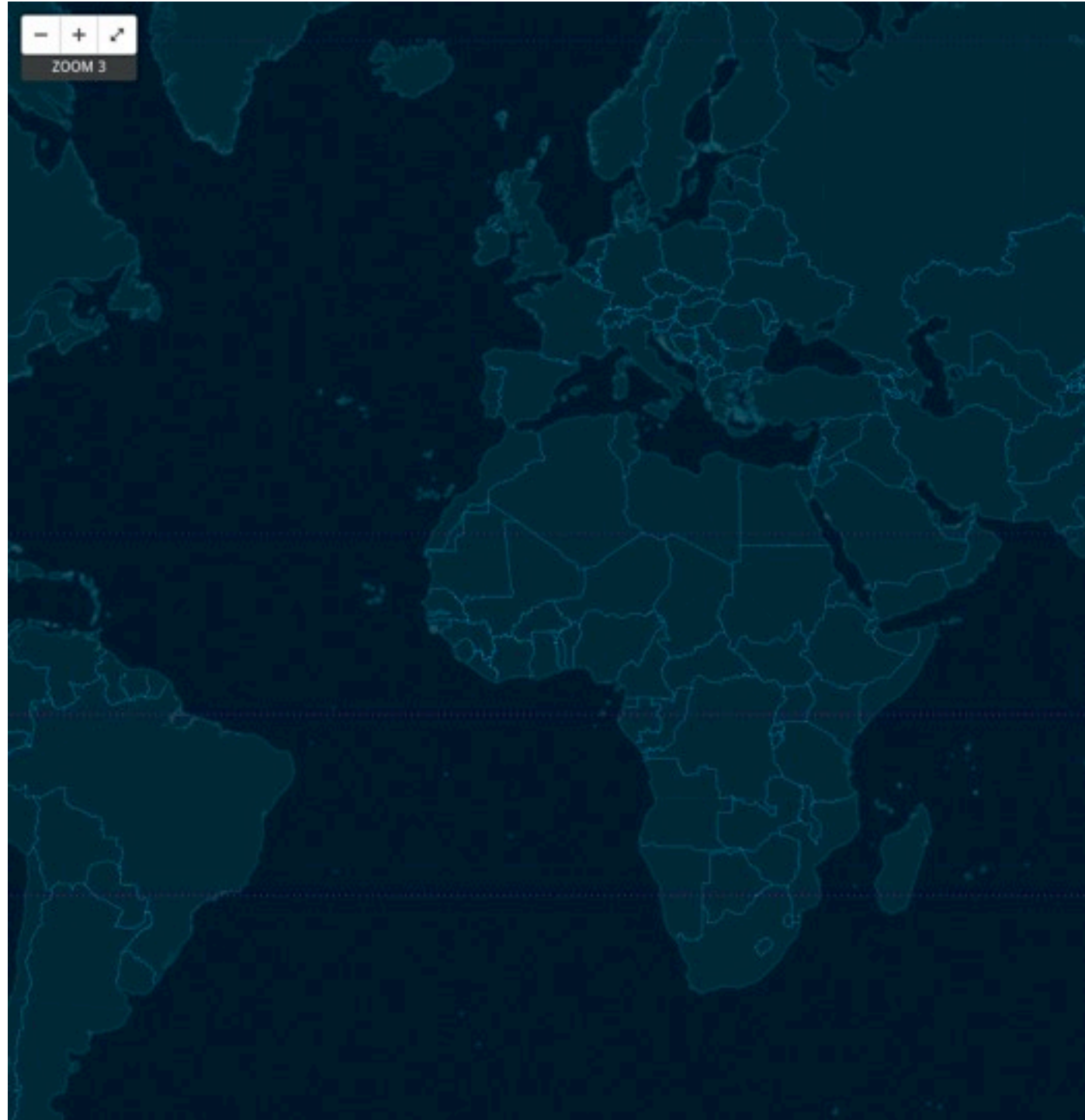
for to style your maps with

It's LESS CSS, but **insane**

```
1 @font: "Futura Medium";
2 @font_alt: "Futura Medium Italic";
3 @water: #94a0c8;
4 @land: #f4efd2;
5 @park: #c5dead;
6 @road: lighten(@land, 50%);
7 @tunnel: lighten(@water, 10%);
8 @non-road: lighten(#000, 50%);
9 @industrial: lighten(#000, 50%);
10 @bridge: lighten(@industrial, 30%);
11 @runway: lighten(@non-road, 30%);
12 @building: darken(@land, 20%);
13 @outline: darken(@land, 65%);
14 @outline-opacity: 0.1;
15
16 Map {
17   background-color:@water;
18 }
19
20 #land {
21   polygon-fill:@land;
22   polygon-opacity:1;
23 }
24
25 #water {
26   polygon-fill:@water;
27 }
28
29 #buildings {
30   building-fill:@building;
31   building-fill-opacity:0;
32   building-height:10;
33
34   [zoom>=15] {
35     building-fill-opacity:0.9;
36   }
37 }
38
```

} Variables & functions like LESS

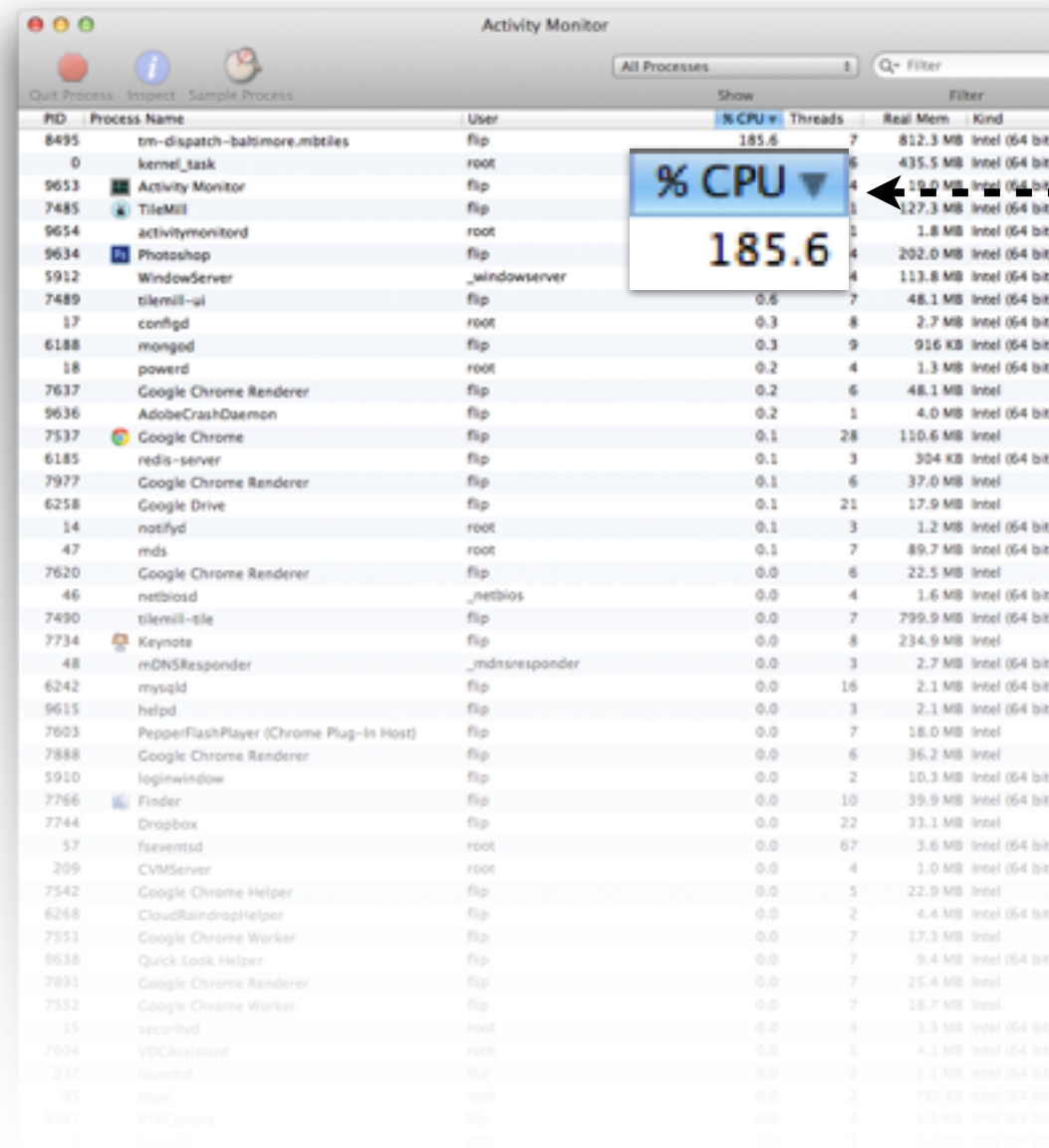
...but that ain't LESS



Still, you can make pretty maps...



Unless they're too complex.



PID	Process Name	User	% CPU	Threads	Real Mem	Kind
8495	tm-dispatch-baltimore.mbtiles	flip	185.6	7	812.3 MB	Intel (64 bit)
0	kernel_task	root		6	435.5 MB	Intel (64 bit)
9653	Activity Monitor	flip		4	19.0 MB	Intel (64 bit)
7485	TileMill	flip		1	127.3 MB	Intel (64 bit)
9654	activitymonitord	root		1	1.8 MB	Intel (64 bit)
9634	Photoshop	flip		4	202.0 MB	Intel (64 bit)
5912	WindowServer	_windowserver		4	113.8 MB	Intel (64 bit)
7489	tilemill-ui	flip	0.6	7	48.1 MB	Intel (64 bit)
17	configd	root	0.3	8	2.7 MB	Intel (64 bit)
6188	mongod	flip	0.3	9	916 KB	Intel (64 bit)
18	powerd	root	0.2	4	1.3 MB	Intel (64 bit)
7637	Google Chrome Renderer	flip	0.2	6	48.1 MB	Intel
9636	AdobeCrashDaemon	flip	0.2	1	4.0 MB	Intel (64 bit)
7537	Google Chrome	flip	0.1	28	110.6 MB	Intel
6185	redis-server	flip	0.1	3	304 KB	Intel (64 bit)
7977	Google Chrome Renderer	flip	0.1	6	37.0 MB	Intel
6258	Google Drive	flip	0.1	21	17.9 MB	Intel
14	notified	root	0.1	3	1.2 MB	Intel (64 bit)
47	mds	root	0.1	7	89.7 MB	Intel (64 bit)
7620	Google Chrome Renderer	flip	0.0	6	22.5 MB	Intel
46	netbiosd	_netbios	0.0	4	1.6 MB	Intel (64 bit)
7490	tilemill-tile	flip	0.0	7	799.9 MB	Intel (64 bit)
7734	Keynote	flip	0.0	8	234.9 MB	Intel
48	mDNSResponder	_mdnsresponder	0.0	3	2.7 MB	Intel (64 bit)
6242	mysqld	flip	0.0	16	2.1 MB	Intel (64 bit)
9615	helpd	flip	0.0	3	2.1 MB	Intel (64 bit)
7603	PepperFlashPlayer (Chrome Plug-In Host)	flip	0.0	7	18.0 MB	Intel
7888	Google Chrome Renderer	flip	0.0	6	36.2 MB	Intel
5910	loginwindow	flip	0.0	2	10.3 MB	Intel (64 bit)
7766	Finder	flip	0.0	10	39.9 MB	Intel (64 bit)
7744	Dropbox	flip	0.0	22	33.1 MB	Intel
57	fservntsd	root	0.0	67	3.6 MB	Intel (64 bit)
209	CVMServer	root	0.0	4	1.0 MB	Intel (64 bit)
7542	Google Chrome Helper	flip	0.0	5	22.9 MB	Intel
6268	CloudRaindropHelper	flip	0.0	2	4.4 MB	Intel (64 bit)
7551	Google Chrome Worker	flip	0.0	7	17.3 MB	Intel
9638	Quick Look Helper	flip	0.0	7	9.4 MB	Intel (64 bit)
7891	Google Chrome Renderer	flip	0.0	7	25.4 MB	Intel
7552	Google Chrome Worker	flip	0.0	7	18.7 MB	Intel
15	securityd	root	0.0	4	3.3 MB	Intel (64 bit)
7604	VDCAssistant	root	0.0	3	4.1 MB	Intel (64 bit)
237	launchd	flip	0.0	2	3.1 MB	Intel (64 bit)
95	mpd	root	0.0	2	792 KB	Intel (64 bit)
6337	PTPCamera	flip	0.0	4	5.5 MB	Intel (64 bit)
5	lsinitd	root	0.0	5	1.2 MB	Intel (64 bit)

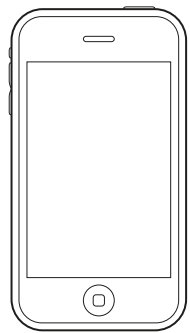
That's not an amount of CPU!  
That a thing takes!

Unless they're too complex.



My map of Baltimore  
wouldn't export.

It's \*just\* of Baltimore.



# Chapter 3: iOS

**Cause you're all like, "WTF THIS IS BMORE  
COCOA NOT BMORE MAPPING"**

# 3.1: Installing MapBox

I prefer git submodules. YMMV, but this is how I got it working.

```
$ git submodule add git://github.com/mapbox/mapbox-ios-sdk.git
```

# Add MapBox's submodules

```
$ git submodule update --init --recursive
```

This is the **\*most important\*** part of getting  
MapBox running!

# Add MapBox to your target

Demo/mapbox-ios-sdk/MapView/MapView.xcodeproj

**drag to your Frameworks folder**

# Add libraries to your target

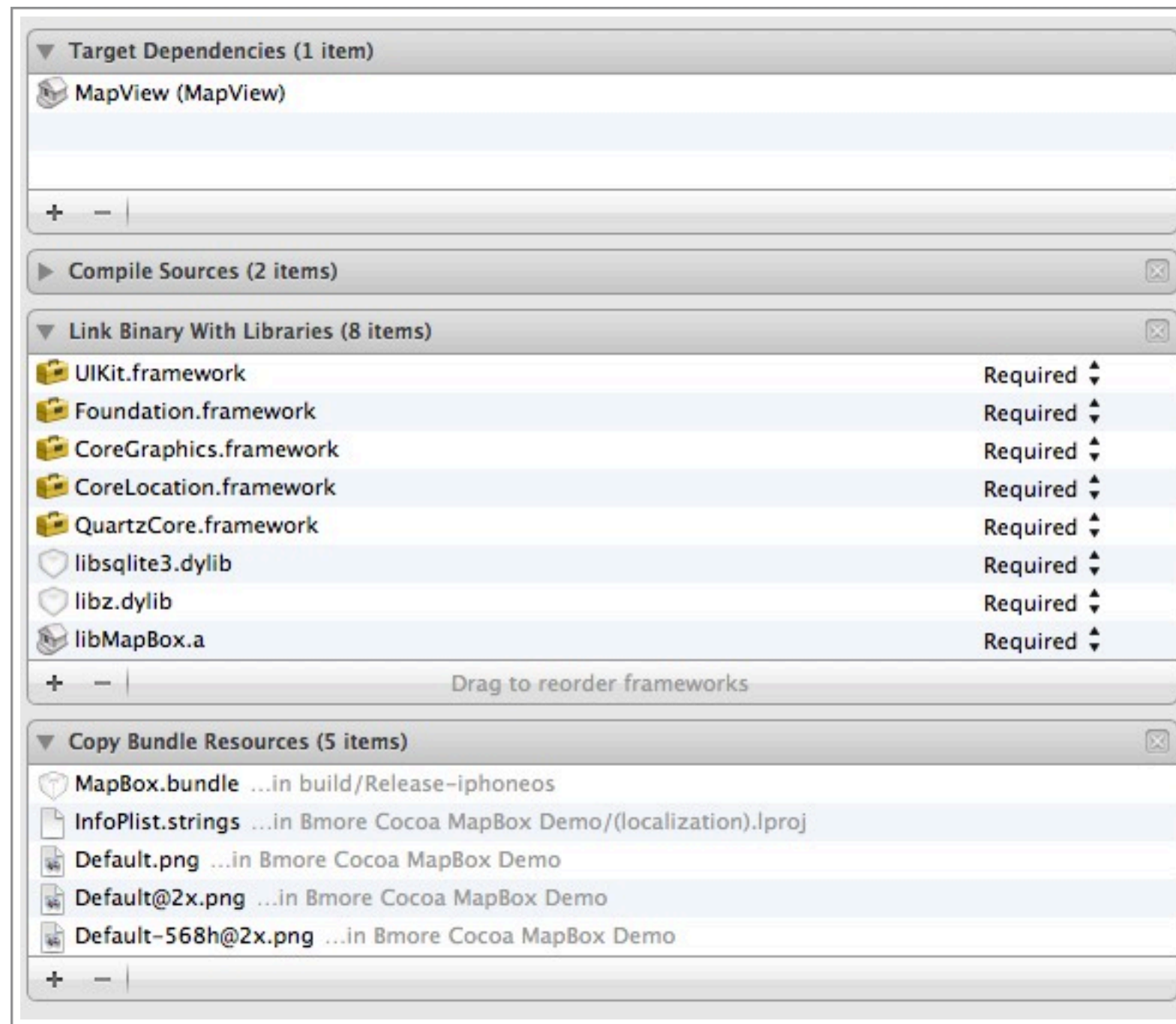
- CoreLocation
- QuartzCore
- libsqlite3
- libz
- libMapBox

# Add to your header search path

```
$ (SRCROOT) /mapbox-ios-sdk/MapView/
```

**Add to Header Search Paths for your  
target**

**Check “recursive”**




# Target dependencies & resources



# ...back to the demo

## MapBox ID



```
1 - (void)viewDidLoad {
2   RMMapBoxSource *onlineSource = [[RMMapBoxSource alloc] initWithMapID:@"flipsasser.map-tlt4zah7"];
3   self.mapView = [[RMMapView alloc] initWithFrame:self.view.frame andTileSource:onlineSource];
4   self.mapView.autoresizingMask = UIViewAutoresizingFlexibleHeight | UIViewAutoresizingFlexibleWidth;
5   self.mapView.hideAttribution = true;
6   self.mapView.showLogoBug = false;
7   self.mapView.tileSource = onlineSource;
8   [self.view addSubview:self.mapView];
9   [super viewDidLoad];
10 }
```




Voilà!

# RMMBTilesSource

For storing tiles locally

# Local Source



```
1 - (void)viewDidLoad {
2   NSURL *tileSetURL = [[NSBundle mainBundle] URLForResource:@"Baltimore" withExtension:@"mbtiles"];
3   RMMBTilesSource *localSource = [[RMMBTilesSource alloc] initWithTileSetURL:tileSetURL];
4   self.mapView = [[RMapView alloc] initWithFrame:self.view.frame andTilesSource:onlineSource];
5   self.mapView.tileSource = localSource;
6   [self.view addSubview:self.mapView];
7   [super viewDidLoad];
8 }
```



Voilàier!\*

\*this is a tiny subset of my original map

# RMMMapViewDelegate

For adding markers, shapes, layers!

For responding to boundary changes!

For handling taps and gestures!

**RTFM!**

# Other awesome stuff

- **REAL shape drawing**
- **Custom tile systems (for the adventurous!)**
- **Caching of remote tiles**
- **Animated zooming (looks AWESOME)**

# Drawbacks



# TileMill

The worst or the worst?

# Raster vs. Vector

Tiles are old technology

# Conclusions

# MapBox is right if you need...

- Custom map styles
- Complicated drawing
- Beautiful animation
- Public APIs for drawing, tiling, and mercator projections
- Accurate data (thanks anyway, Apple)

# MapBox is wrong if you need...

- Simple or quick maps
- Vector maps

# Thnaks!

[github.com/BackForty/map\\_box\\_demo](https://github.com/BackForty/map_box_demo)

Check out demo the  
source and this  
presentation:

[github.com/BackForty/map\\_box\\_demo](https://github.com/BackForty/map_box_demo)