learn.js
BOOK #3

#3

awesome.addTo(map);

mapping *with* leaflet.js

SETH VINCENT

Learn.js #3

Mapping with Leaflet.js

Seth Vincent

This book is for sale at http://leanpub.com/learnjs-03

This version was published on 2014-02-02

Contents

Introduction

Thank you.

I really appreciate the support you've given by purchasing this book. I welcome you to help guide the direction of this book and the Learn.js series of javascript books. If there are particular libraries, development tools, or programming patterns that you'd like to see covered, please email me at hi@learnjs.io[1].

The Learn.js series is highly inspired by the lean publishing model (read more about it[2]) proposed by Peter Armstrong, founder of leanpub.com. It has proven successful as a way to receive feedback from you, the readers, so that together we can make the best book about javascript tools and libraries possible.

You'll get updates about upcoming books in the series, and I'd love to hear your thoughts on what would be most useful.

Tools we'll use in this book

Many of the tools used in this book are covered in detail in one of my other books, Development Environments for Beginners.

If you're feeling like you could use more information about what a development environment is and how best to set one up, you can purchase the Development Environments book at superbigtree.com/books/dev-envs[3].

If needed you can check out the Development Environments book on GitHub for free here: github.com/sethvincent/dev-envs-book[4].

Sublime text editor

Sublime is a popular text editor with versions for Mac, Windows, and Linux.

You can download an evaluation copy for free, and pay for a license when you're ready.

In this book we'll be using version 2 of Sublime, in future updates to the book we'll switch to version 3.

[1] mailto:hi@learnjs.io
[2] https://leanpub.com/manifesto
[3] http://superbigtree.com/books/dev-envs
[4] http://github.com/sethvincent/dev-envs-book

Install

Go to sublimetext.com[5] and click the download button.

Terminal

You'll be using the terminal (or command-line) for many of the tutorials in this book. The Terminal chapter in Development Environments for Beginners[6] will help you get started.

Node.js

We'll be using node.js mostly to get at npm, node's bundled package manager.

We will also write modules in the style of node.js.

Our code written for the browser will utilize the node.js style of modules thanks to browserify, a tool for bundling node modules for the browser.

This means that we won't cover the RequireJS/AMD toolset for javascript development, but will focus on node/CommonJS modules whenever possible.

You'll learn more about this later in the book as we go into depth with using Leaflet.js alongside browserify.

npm

npm is the package manager that comes bundled with node.js. Sometimes people call it the node package manager, but I prefer a term used by programmer Max Ogden: "node packaged modules". The idea is that we can store whatever modules we want on npm. Leaflet.js is an example.

In each chapter we'll explore npm a little more, and by the end of the book you'll have learned it pretty thoroughly.

git

For many of the examples we'll use git as our version control software. If you're new to git, learn more about it in this Git chapter of the Development Environments book[7], the Try Git interactive tutorial[8], and the Pro Git book that's available for free on the git website[9].

[5]http://www.sublimetext.com/
[6]https://github.com/sethvincent/dev-envs-book/blob/master/manuscript/terminal.md
[7]http://git-scm.com/book
[8]http://try.github.io/
[9]http://git-scm.com/book

GitHub Pages

All of the examples we work through in the book can be hosted using GitHub Pages. I recommend that put each map you make while reading this book up on GitHub Pages. It'll be good practice if you're new to git and GitHub.

Learn more about GitHub pages in the related chapter of the Development Environments book[10], and on GitHub's Help section[11].

Chapters

Here's the progress I've made so far on chapters. As you likely know, I'm releasing this book in pieces, a few chapters at a time, and you're essentially subscribed to those chapter releases. The upcoming chapter list is somewhat revisable. I'm particularly interested in hearing your thoughts on what chapters would be valuable to you. Email me at seth@superbigtree.com to let me know what types of Leaflet.js content you would find useful.

Completed:

- The simplest possible map you can make
  - html file
  - Remote script and stylesheet files
  - A map with one marker
  - Put the map on GitHub Pages
- Getting started with Leaflet.js using node.js, npm, and browserify
- Using alternate tilesets

Started:

- Useful Leaflet.js plugins
  - Leaflet.markercluster
  - Leaflet.label
  - Leaflet.awesome-markers
  - Leaflet.TextPath
  - leaflet-hash
  - L.GeoSearch
  - Leaflet.draw
  - leaflet-search

[10]https://github.com/sethvincent/dev-envs-book/blob/master/manuscript/github.md
[11]https://help.github.com/categories/20/articles

Upcoming:

- Data visualization styles you can use in your maps
- Customizing your maps
  - Markers
  - Controls
- Drawing shapes on maps
- Advanced layer types
- GeoJSON & Leaflet.js
  - TopoJSON
- Using remote data in your maps
  - Google Spreadsheet and Tabletop.js
  - Twitter
  - Instagram
  - LocalWiki
- Using D3.js with Leaflet.js
- Additional resources directory

Please feel free to suggest topics or chapters by emailing me at seth@superbigtree.com,

Who are you? Who am I? What is this?

The book

This book is meant as an introductory text that will get people up to speed for building interactive maps with Leaflet.js.

This book is an introductory text. I aim for this book to be a conversational and low-barrier approach to learning javascript and Leaflet. Everything we work on in this book can be done with just a browser, a terminal, and a text editor.

The reader

I expect that the ideal reader for this book is someone who likes exploring, imagining, and inventing for themselves. You might even have some experience with javascript already. And that's OK, because practicing, and even repetition is an important part of learning.

The author

I'm Seth Vincent. I write code, stories, and music.

I'm an independent programmer, designer and writer that is passionate about news, publishing and civic technology – particularly as it applies to local issues.

I'm a co-organizer of seattle.io[12], Code for Seattle[13], and SeattleWiki[14].

In case you couldn't tell, I currently live in Seattle, Washington.

I write books like the one you're reading, and build things like crtrdg.js, a toolkit for 2d games[15] at Super Big Tree[16].

Setting up a development environment

There's a lot of wind-up to getting started programming. You should understand things like git, github, the terminal, and more.

Instead of baking that information into each book in the series, I created a book called Development Environments for Beginners that helps you set up a javascript development environment (as well as ruby and python, but you can skip those sections if needed).

From that book you'll learn how to install node.js, work with version control and testing tools, best practices for automating tasks and other programming tips and tricks.

If you're feeling like you could use more information about what a development environment is and how best to set one up, you can purchase the Development Environments book at superbigtree.com/books/dev-envs[17].

If needed you can check out the Development Environments book on GitHub for free here: github.com/sethvincent/dev-envs-book[18].

Though, if you're feeling generous and able to purchase the book, that'll get you pdf, epub, and mobi versions, as well as support my work.

[12]http://seattle.io
[13]http://codeforseattle.org/
[14]http://seattlewiki.net/
[15]http://crtrdg.github.io/
[16]http://superbigtree.com/
[17]http://superbigtree.com/books/dev-envs
[18]http://github.com/sethvincent/dev-envs-book

The simplest possible map you can make

Get a CloudMade account and API key

For a number of examples in this book we'll use map tiles from CloudMade. Create an account at cloudmade.com[19] and get an API key to use in these examples.

Create an index.html file

Here's the full index.html file we're using in this example. As you can see there's a minimal amount of work involved in creating a basic map. If you're new to some of these concepts keep reading, we'll go over this code in detail.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Leaflet Example</title>

    <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.6.4/leaflet.c\
ss" />
    <!--[if lte IE 8]>
      <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.6.4/leaflet\
.ie.css" />
    <![endif]-->

    <style>
    html, body { height: 100%; margin: 0px; }
    #map { height: 100%; }
    </style>
  </head>
  <body>
    <div id="map"></div>

    <script src="http://cdn.leafletjs.com/leaflet-0.6.4/leaflet.js"></script>
```

[19] http://cloudmade.com/

```

    <script>
    var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStr\
eetMap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/"\
>CC-BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';

    var map = L.map('map').setView([48.98337, -123.05855], 13);

    L.tileLayer('http://{s}.tile.cloudmade.com/!!!APIKEY!!!/997/256/{z}/{x}/{y}.p\
ng', {
      attribution: attribution,
      maxZoom: 18
    }).addTo(map);

    var marker = L.marker([48.98337, -123.05855]).addTo(map);
    var markerHtml = 'This place is weird';
    marker.bindPopup(markerHtml);
    </script>
  </body>
</html>
```

The full index.html file explained in detail

Get started

Give your html file the HTML5 doctype, open the `<html>` and `<head>` sections, and give your page a title.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Leaflet Example</title>
```

Add Leaflet style

Include the Leaflet.js stylesheets from the leafletjs.com servers. The `<!--[if lte IE 8]>` tag is used to include the css styles that are specifically for versions of Internet Explorer less than or equal to IE 8.

```
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.css" \
/>
```

Add your own style

Here we're adding the least amount of css it takes to get the map to fill the screen.

```
<style>
  html, body { height: 100%; margin: 0px; }
  #map { height: 100%; }
  </style>
</head>
```

Create the map container.

This opens the `<body>` section. The `#map` div is where we'll put the map using javascript.

```
<body>
  <div id="map"></div>
```

Leaflet javascript library

Include the Leaflet.js library that's hosted on the leafletjs.com servers using this script tag.

```
<script src="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.js"></script>
```

Your map script

We use an opening `<script>` tag, then set a variable named attribution to a string that contains all the copyright info we need to convey about the map. If you use different tiles or data you'll want to change the attribution to reflect the resources you use in your map.

```
<script>
var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetM\
ap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-\
BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';
```

Initialize the map

We're creating a map object and assigning it to the map div. `L.map()` takes the id of an html element as an argument, so we're passing in `map` to target the div we made.

`.setView()` takes the latitude and longitude that we want the map to center on as an array, and an integer for the zoom level.

```
var map = L.map('map').setView([48.98337, -123.05855], 13);
```

Add tiles to the map

`L.tileLayer()` is used to add tiles to the map. We set our attribution variable to the attribution option, and tell the layer to max out zooming at level 18.

Make sure to replace !!!APIKEY!!! with the API key you got from Cloudmade.

The `.addTo(map);` method is used to finally put the tile layer on the map.

```
L.tileLayer('http://{s}.tile.cloudmade.com/!!!APIKEY!!!/997/256/{z}/{x}/{y}.png',\
 {
  attribution: attribution,
  maxZoom: 18
}).addTo(map);
```

Add a marker

The `L.marker()` method just takes an array with the latitude and longitude of the place you'd like the map to be positioned.

Next we create a variable named `markerHtml` witih some text we want to show in a pop-up.

With `marker.bindPopup()` we add a pop-up window to the marker, and pass in the markerHtml variable to get that text to show up in the pop-up. You can put any kind of html in a pop-up, not just text.

```
var marker = L.marker([48.98337, -123.05855]).addTo(map);
var markerHtml = 'This place is weird';
marker.bindPopup(markerHtml);
</script>
```

That's about it

Here we close the `<body>` and `<html>` tags. And that's it for this example.

```
  </body>
</html>
```

Nice! Now we can get that map running on a server somewhere!

Publish the map on GitHub Pages

Let’s host the map on GitHub pages, an awesome free service from GitHub that we can use for simple html sites.

First, create an account on github.com[20] if you haven’t already.

Next, create a new repository.

Give your repository a name.

Now clone the project repository onto your computer:

```
git clone git@github.com:__YOUR-USERNAME__/__YOUR-PROJECT-NAME__.git
```

You can copy the git url to clone from the right-hand sidebar of your project repository.

After cloning the repository, cd into it and make some changes:

```
cd __YOUR-PROJECT-NAME__
nano index.html
```

Add the contents of our map project to index.html.

After you’ve added the content, add them to the repo and commit the changes:

```
git add .
git commit -m 'include a brief, clear message about the changes'
```

Put your changes on a branch named gh-pages:

```
bit checkout -b gh-pages
```

Now, push your changes back to GitHub:

```
git push origin gh-pages
```

Troubleshooting

If GitHub Pages shows a 404 error at first, don’t worry. It can take as much as 15 minutes before the GitHub server builds and starts serving your site. Be patient when you first launch the site. Later when you make updates the live site will usually reflect the changes more quickly.

[20]http://github.com

Getting started with Leaflet.js using node.js, npm, and browserify

Leaflet.js[21] is an awesome, easy to learn mapping library. Here we'll go through some introductory examples so you can get started using it in your projects.

Hey, if leaflet is a mapping library for browsers, what's it doing on npm?

Client-side code can be distributed through npm[22], and combined with a tool like browserify[23], which bundles node modules for the browser, it works pretty great.

Make sure you have node installed. I recommend using a tool called `nvm` for installing node.js if you're using mac or linux.

nvm:

github.com/creationix/nvm[24].

You can also install node.js by downloading an installer from the nodejs.org downloads page:

nodejs.org/downloads[25].

Installation and setting up

First, create and navigate to a new project directory and create these files:

```
mkdir leaflet-example
cd leaflet example
touch index.html app.js style.css
```

Now run `npm init` to create a package.json file.

You should get something like this after you answer the questions:

[21] http://leafletjs.com/
[22] http://maxogden.com/node-packaged-modules.html
[23] https://github.com/substack/node-browserify
[24] https://github.com/creationix/nvm
[25] http://nodejs.org/downloads

```
{
  "name": "leaflet-basics",
  "version": "0.0.0",
  "description": "introduction to using leaflet with browserify",
  "main": "app.js",
  "scripts": {
    "start": "beefy app.js:bundle.js --live",
    "bundle": "browserify app.js -o bundle.js"
  },
  "author": "",
  "license": "MIT",
  "dependencies": {
  },
  "devDependencies": {
  }
}
```

Now install leaflet using `npm`

We'll use the `--save` argument so that leaflet.js is saved as a dependency in package.json

```
npm install --save leaflet
```

html and css

Create an html file that looks like this:

```
<!doctype html>
<html>
<head>

<title>leaflet example</title>
<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="node_modules/leaflet/dist/leaflet.css">
<link rel="stylesheet" href="style.css">
</head>
<body>

```

```
<div id="map"></div>

<script src="bundle.js"></script>
</body>
</html>
```

Note that for the stylesheets we're providing the path directly to the leaflet package in the node_-modules folder:

```
node_modules/leaflet/dist/leaflet.css
```

Our script tag points to `bundle.js`, which will be generated by a tool called beefy[26], which creates a development server with live reloading of browserify bundles.

Make sure the `#map` div has a height, otherwise the map won't show up.

Add this to your the styles.css file:

```
#map {
  height: 300px;
}
```

require leaflet

Now, in the app.js file, require the leaflet module like this:

```
var L = require('leaflet');
```

You could probably use any variable name here, but `L` is a common variable name for leaflet. You'll see it in the leaflet docs[27] and tutorials[28].

fix image path

Because we're using browserify, we need to tell leaflet the specific location of the images folder that it needs to render the page:

```
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';
```

Now we can create a map!

You can use the `L.map()` method to create the map:

[26]https://github.com/chrisdickinson/beefy
[27]http://leafletjs.com/reference.html
[28]http://leafletjs.com/examples.html

```
var map = L.map('map');
map.setView([47.63, -122.32], 11);
```

The `map.setView()` method centers the map at a latitude/longitude, then sets the zoom level. In the above example the lat/long centers the map on Seattle, WA, and the zoom level is at 11.

Finally, we need to tell the map to use a specific tileset:

```
var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetM\
ap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-\
BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';

var tiles = 'http://{s}.tile.cloudmade.com/!!! YOUR API KEY !!!/997/256/{z}/{x}/{\
y}.png';

L.tileLayer(tiles, {
  maxZoom: 18,
  attribution: attribution
}).addTo(map);
```

It's important to properly attribute tile image and data sources, so we're setting the `attribution` variable to a string crediting OpenStreetMap and CloudMade.

Next the `tiles` variable is set to a url for a popular cloudmade tileset. Note that we're using the api key that Leaflet uses in tutorials. For production code you should create an account on cloudmade.com[29], and create an api key for your app.

Then you can replace the `!!! YOUR API KEY !!!` portion of the url seen below with your api key:

```
var tiles = 'http://{s}.tile.cloudmade.com/!!! YOUR API KEY !!!/997/256/{z}/{x}/{\
y}.png';
```

Currently, your app.js file should look like this:

[29]http://account.cloudmade.com/

```
// require leaflet.js
var L = require('leaflet');

// specify the path to the leaflet images folder
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';

// initialize the map
var map = L.map('map', {
  scrollWheelZoom: false
});

// set the position and zoom level of the map
map.setView([47.63, -122.32], 11);

// set an attribution string
var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetM\
ap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-\
BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';

// set the tiles the map will use
var tiles = 'http://{s}.tile.cloudmade.com/!!! YOUR API KEY !!!/997/256/{z}/{x}/{\
y}.png';

// create a tileLayer with the tiles, attribution
var layer = L.tileLayer(tiles, {
  maxZoom: 18,
  attribution: attribution
});

// add the tile layer to the map
layer.addTo(map);
```

Serving the development site using browserify & beefy

Now that we’ve got our basic components together, let’s use browserify and beefy so you can serve the site on your computer.

First, install beefy and browserify. Here we’ll use the `--save-dev` argument so that these modules get saved in our package.json file as development dependencies.

```
npm install --save-dev beefy browserify
```

Now that those modules are installed, we'll use the `beefy` command to run a development version of the site:

```
beefy app.js:bundle.js --live
```

This command takes our node-style code in app.js, and bundles it using browserify to serve the bundle.js file. The `--live` argument enables live reloading of the page, so every time you save a file, the browser will reload.

We can make our lives simpler and not have to remember/type that command all the time by using npm scripts.

In your package.json file, and change the scripts property to look like this:

```
"scripts": {
  "start": "beefy app.js:bundle.js --live"
},
```

Now, we can run a much simpler command to run the site:

```
npm start
```

Run that and you'll be able to go to http://localhost:9966 to see your site.

Note that running beefy doesn't create a bundle.js file in your file system, it just serves one through the development server.

To create a bundle.js file that you can use to deploy on GitHub Pages or wherever you host your site, you'll use the `browserify` command.

This will generate a bundle.js file:

```
browserify app.js -o bundle.js
```

We can use npm scripts again to simplify this:

```
"scripts": {
  "start": "beefy app.js:bundle.js --live",
  "bundle": "browserify app.js -o bundle.js"
},
```

Now, the `start` command is one of the few that npm directly supports. We can create commands with arbitrary names like `bundle`, but to use it, we prepend it with the `run` command, so it'll look like this:

```
npm run bundle
```

Run the above command and that will create the bundle.js file you can use when deploying your site to GitHub Pages or to another server.

Basic elements of a Leaflet.js map

There are a few basic elements that you'll use on regular basis for drawing on a map. Here's a look at how to make each one of those:

UI Layers

popups

To get started learning popups, let's make the map show a popup with the latitude and longitude of place we click on the map.

To create a popup, use the `L.popup()` method:

```
var popup = L.popup();
```

Next we need to listen for the `click` event on the map:

```
map.on('click', onClick);
```

We've passed a function named `onClick` to the event listener, so let's create that function:

```
function onClick(e){
    popup
    .setLatLng(e.latlng)
    .setContent("You clicked the map at " + e.latlng.toString())
    .openOn(map);
}
```

markers

To create a marker, use the `L.marker()` method:

```
var marker = L.marker([47.63, -122.32]);
marker.addTo(map);
```

The above code will add a marker to the center of your map, as we're using the some lat/long for this marker as the map view.

Let's add a popup to that marker.

First, set some html content to a variable:

```
var markerHtml = 'this is a marker. pretty great, right?';
```

Next, bind a popup to the marker:

```
marker.bindPopup(markerHtml);
```

Now when you click on the marker, the popup will show up the the html you specified.

To make the popup open up by default, try this:

```
marker.bindPopup(popupContent).openPopup();
```

Now when you look at your map again, you'll see that the popup is open by default.

Vector layors

Vectory layers are the things you can draw on top of the map. Some of the basic types: lines, rectangles, circles, and polygons. In this section we'll make one of each of those vector layer types.

Lines

We'll use the `L.polyline` method for drawing lines on the map. it takes an array of lat/long coordinates, and an options object[30]. This method extends the base `L.Path` method, so you can also use any path options[31].

Here's an example:

```
var polylinePath = [
  [47.607204675859045, -122.3298454284668],
  [47.60182268729636, -122.32521057128906],
  [47.60095457276622, -122.32349395751952],
  [47.5998260023411, -122.32237815856934],
  [47.59842248977284, -122.32124090194701],
  [47.59655590441905, -122.32023239135741],
  [47.59175167310035, -122.32126235961914],
  [47.588278460128734, -122.32057571411133],
  [47.58329978624572, -122.31997489929199],
  [47.579478622338286, -122.3192024230957],
  [47.57756793579513, -122.3196315765381]
```

[30]http://leafletjs.com/reference.html#polyline-options

[31]http://leafletjs.com/reference.html#path-options

```
];

var polyline = L.polyline(polylinePath);
polyline.addTo(map);
```

This is a line along part of the I-5 highway in the Seattle area. Try it out in your map. Try changing the path to see what lines you can make. Remember that we made that popup that shows the lat/long when you click on the map. Use that to come up with coordinates that you want to place in your lat/lng array.

Rectangles

Creating a rectangle is very similar to creating a polyline, but takes two coordinates that represent two corners of the rectangle.

Here's an example that loosely outlines Lake Union in Seattle:

```
var rectBounds = [[47.64614, -122.34555], [47.6274, -122.32839]];

var rectangle = L.rectangle(rectBounds);
rectangle.addTo(map);
```

Circles

When creating a circle, we use the `L.circle` method and give one lat/lng coordinate as the center point, followed by the radius.

Here's an example with a circle centered on the Seattle neighborhood Fremont (also known as the center of the universe):

```
var circle = L.circle([47.65136, -122.35087], 300)
circle.addTo(map);
```

Polygons

A polygon can be any arbitrary shape, good for outlining buildings, parks, etc. To create a polygon, we use `L.polygon`.

Here's an example that roughly outlines the downtown Seattle area:

```
L.polygon([
  [47.61311, -122.34967],
  [47.61594, -122.33156],
  [47.60466, -122.32169],
  [47.60211, -122.33551]
]).addTo(map);
```

What's next

This gets you started with leaflet.js. You've learned about getting a map to show up on a page and drawing basic shapes. Just by composing shapes and popup markers in this way you can make a useful map very easily.

Using alternate tilesets

The Leaflet tutorials all use CloudMade tiles for the examples. Those tiles look great, but it's common to want a little more control over the style of the map.

For the examples in this chapter we'll build on the example created in chapter 3.

In this chapter we'll take a look at these options for Leaflet tiles:

- Using Stamen tiles[32]
- Using Mapbox tiles[33]
- Using alternate CloudMade tiles[34]

But first we'll do a quick refresher on the file structure to make sure you have everything set up.

Get set up

If you already created the example project from chapter 3 and have it ready to go, cd into that directory and skip or skim this set up section. Otherwise, if you need to recreate the project or feel like you could benefit from going through the code again, check out this section before moving to the alternate tilesets.

Here is what your full index.html file should look like:

```html
<!doctype html>
<html>
<head>
  <title>leaflet example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="node_modules/leaflet/dist/leaflet.css">
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="map"></div>
```

[32]http://maps.stamen.com
[33]https://www.mapbox.com
[34]http://maps.cloudmade.com/editor

```
  <script src="bundle.js"></script>
</body>
</html>
```

Here is what should be in your style.css file:

```
#map {
  height: 300px;
}
```

And this is what your app.js file should look like:

```
// require leaflet.js
var L = require('leaflet');

// specify the path to the leaflet images folder
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';

// initialize the map
var map = L.map('map', {
  scrollWheelZoom: false
});

// set the position and zoom level of the map
map.setView([47.63, -122.32], 11);

// set an attribution string
var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetM\
ap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-\
BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';

// set the tiles the map will use
var tiles = 'http://{s}.tile.cloudmade.com/!!! YOUR API KEY !!!/997/256/{z}/{x}/{\
y}.png';

// create a tileLayer with the tiles, attribution
var layer = new L.StamenTileLayer('watercolor');

// add the tile layer to the map
layer.addTo(map);
```

If you lost your CloudMade API key, you can find it by logging in to cloudmade.com[35], or you can also make a new API key there.

The contents of your package.json file should be very similar to this:

```
{
  "name": "leaflet-basics",
  "description": "introduction to using leaflet with browserify",
  "version": "0.0.0",
  "main": "app.js",
  "scripts": {
    "start": "beefy index.js:bundle.js --live",
    "bundle": "browserify index.js -o bundle.js"
  },
  "author": "",
  "license": "MIT",
  "dependencies": {
    "leaflet": "~0.7.2"
  },
  "devDependencies": {
    "beefy": "~0.7.1",
    "browserify": "~3.24.10"
  }
}
```

The important parts of the package.json file are the `scripts`, `dependencies`, and `devDependencies` sections. Make sure that at least those fields are intact in your package.json file.

If you don't have a node_modules folder in your project directory, run this command to install the project dependencies:

```
npm install
```

Now we should be ready!

Install the leaflet-providers module

In order to use these alternate tile providers, we're going to use a module named leaflet-providers[36].

With this module we can use tile layers from:

[35] http://cloudmade.com
[36] https://github.com/leaflet-extras/leaflet-providers

- Mapbox
- CloudMade
- Stamen
- OpenStreetMap
- MapQuestOpen
- Esri
- OpenWeatherMap
- Nokia

Mapbox, CloudMade, and Nokia require registration. Learn more at the leaflet-providers GitHub page[37].

Install leaflet-providers

```
npm install --save leaflet-providers
```

Using Stamen tiles

Stamen is a design company that makes all kinds of beautiful things, including maps.

Take a look at their Toner[38], Terrain[39], and Watercolor[40] map tiles. They're stunning. See more at maps.stamen.com[41].

For this example, we'll use the Watercolor tileset.

To use the Watercolor tiles, require the `leaflet-providers` module in your app.js file:

```
require('leaflet-providers');
```

> Note that we're not setting this module to a variable like with other modules. It's because the leaflet-providers module is a plugin to Leaflet that adds a `provider()` method to `L.tileLayer`. By requiring the module without setting it to a variable the module can modify variables in the global scope when required.

Next, set the layer variable to the new Stamen Watercolor tiles:

[37] https://github.com/leaflet-extras/leaflet-providers
[38] http://maps.stamen.com/toner/#14/37.7777/-122.4073
[39] http://maps.stamen.com/terrain/#14/37.7777/-122.4073
[40] http://maps.stamen.com/watercolor/#14/37.7777/-122.4073
[41] http://maps.stamen.com

```
var layer = L.tileLayer.provider('Stamen.Watercolor').addTo(map);
```

Your full app.js file should now look like this:

```
// require leaflet.js
var L = require('leaflet');
require('leaflet-providers');

// specify the path to the leaflet images folder
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';

// initialize the map
var map = L.map('map', {
  scrollWheelZoom: false
});

// set the position and zoom level of the map
map.setView([47.63, -122.32], 11);

// create a tileLayer with the tiles, attribution
var layer = L.tileLayer.provider('Stamen.Watercolor').addTo(map);

// add the tile layer to the map
layer.addTo(map);
```

Note that we removed these lines:

```
// set an attribution string
var attribution = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetM\
ap</a> contributors, <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-\
BY-SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>';

// set the tiles the map will use
var tiles = 'http://{s}.tile.cloudmade.com/!!! YOUR API KEY !!!/997/256/{z}/{x}/{\
y}.png';
```

They are unnecessary because the `L.tileLayer.provider('Stamen.Watercolor')` method takes care of the tiles url and attribution string for us.

Check out your Watercolor map!

If you run `npm start` and go to http://localhost:9966, you should see an awesome watercolor map! Looks good, right?

Using Mapbox tiles

You'll want to use Mapbox tiles[42]. They are beautiful.

```
L.tileLayer.provider('MapBox.YourUsername.MapID').addTo(map);
```

Here's an example map with new styles I made in about 5 minutes using the simple editor on mapbox.com:

```
var layer = L.tileLayer.provider('MapBox.sethvincent.h60m4iih');
```

Once you add in the above line to replace the CloudMade tileset, your app.js file should look like this:

```
// require leaflet.js
var L = require('leaflet');
require('leaflet-providers');

// specify the path to the leaflet images folder
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';

// initialize the map
var map = L.map('map', {
  scrollWheelZoom: false
});

// set the position and zoom level of the map
map.setView([47.63, -122.32], 11);

// create a tileLayer with the tiles, attribution
var layer = L.tileLayer.provider('MapBox.sethvincent.h60m4iih');

// add the tile layer to the map
layer.addTo(map);
```

When you make a map at mapbox.com, look for the map ID in your projects list after you log in, or, if you're editing a map, the ID is found be clicking the gear icon – it'll be at the top of that menu.

In an upcoming chapter we'll work on an example where you'll design completely custom tiles using Mapbox's desktop app TileMill[43], upload the tiles to the Mapbox servers, and use the tiles in a Leaflet map!

[42]https://www.mapbox.com
[43]https://www.mapbox.com/tilemill/

Using alternate CloudMade tiles

Using alternate tiles from CloudMade is very similar using the leaflet-providers module. The main differences are that you must specify an `apiKey` and `styleID`.

Here's an example:

```
var layer = L.tileLayer.provider('CloudMade', {
  apiKey: '!!! YOUR API KEY !!!',
  styleID: '2172'
});
```

The `styleID` of `2172` is for a fairly pleasant tile theme named Orange Yellow.

To find and create new styles for CloudMade tiles go to maps.cloudmade.com/editor[44].

You'll see the `styleID` in the bottom right corner of each thumbnail.

The full app.js file when using the CloudMade provider should look something like this:

```
// require leaflet.js
var L = require('leaflet');
require('leaflet-providers');

// specify the path to the leaflet images folder
L.Icon.Default.imagePath = 'node_modules/leaflet/dist/images/';

// initialize the map
var map = L.map('map', {
  scrollWheelZoom: false
});

// set the position and zoom level of the map
map.setView([47.63, -122.32], 11);

// create a tileLayer with the tiles, attribution
var layer = L.tileLayer.provider('CloudMade', {
  apiKey: '!!! YOUR API KEY !!!',
  styleID: '2172'
});

// add the tile layer to the map
layer.addTo(map);
```

[44]http://maps.cloudmade.com/editor

Other tilesets

Most likely these three tile providers will provide tiles that fit your needs, but if you're looking for more, check out the leaflet-providers repository on GitHub: github.com/leaflet-extras/leaflet-providers[45].

[45]https://github.com/leaflet-extras/leaflet-providers

Useful Leaflet.js plugins

Leaflet.markercluster

Cluster your markers so that when zoomed out it's easier to see what's going on. You can see this plugin in action on Craigslist maps.

Project repository: github.com/Leaflet/Leaflet.markercluster[46]

Leaflet.label

Give nice labels to your markers in addition to the built in pop-ups provided by the core Leaflet library.

Project repository: github.com/Leaflet/Leaflet.label[47]

Leaflet.awesome-markers

Use markers with the Font Awesome icon set[48]. The latest version works with Bootstrap 3[49] and Font Awesome 4.0.

Project repository: github.com/lvoogdt/Leaflet.awesome-markers[50]

Leaflet.TextPath

Show some text along a Polyline layer on your map. Particularly interesting in combination with Font Awesome or another icon font. Useful for marking paths.

Project repository: github.com/makinacorpus/Leaflet.TextPath[51]

[46]https://github.com/Leaflet/Leaflet.markercluster
[47]https://github.com/Leaflet/Leaflet.label
[48]http://fontawesome.io/
[49]http://getbootstrap.com/
[50]https://github.com/lvoogdt/Leaflet.awesome-markers
[51]https://github.com/makinacorpus/Leaflet.TextPath

leaflet-hash

Use leaflet-hash to create urls to map views. Just like Google Maps, you'll be able to share a specific position and zoom level of the map.

Project repository: github.com/mlevans/leaflet-hash[52]

L.GeoSearch

L.GeoSearch provides basic geocoding functionality using Esri, Google, or OpenStreetMap as the geocoding provider.

Project repository: github.com/smeijer/L.GeoSearch[53]

Leaflet.draw

This plugin adds controls for drawing shapes on a Leaflet map.

Project repository: github.com/Leaflet/Leaflet.draw[54]

leaflet-search

Use leaflet-search to quickly filter/search through layers on the map.

Project repository: github.com/stefanocudini/leaflet-search[55]

[52]https://github.com/mlevans/leaflet-hash
[53]https://github.com/smeijer/L.GeoSearch
[54]https://github.com/Leaflet/Leaflet.draw
[55]https://github.com/stefanocudini/leaflet-search

A Work in progress

I'm releasing this book a few chapters at a time in part to get feedback from you about what topics you'd like to see in the book.

The planned upcoming chapters:

- Data visualization styles you can use in your maps
- Customizing your maps
  - Map tiles
    - CloudMade
    - MapBox
    - Stamen
  - Markers
  - Controls
- Drawing shapes on maps
- Advanced layer types
- GeoJSON & Leaflet.js
  - TopoJSON
- Using remote data in your maps
  - Google Spreadsheet and Tabletop.js
  - Twitter
  - Instagram
  - LocalWiki
- Using D3.js with Leaflet.js
- Additional resources directory

By purchasing the book you're basically subscribed to the upcoming chapter releases. The above list is somewhat revisable. I'm interested in hearing your thoughts on what chapters would be valuable to you. Email me at seth@superbigtree.com to let me know what types of Leaflet.js content you would find useful.

Changelog

v0.2.0 - February 3, 2014

- Add Using alternate tilesets chapter
- Make the “Basic elements of a Leaflet.js map” section its own chapter
- Many small typo fixes

v0.1.0 - November 18, 2013

- Add introduction chapter
- Add simplest possible map you can make chapter
- Add getting started with leaflet.js using node.js, npm, and browserify chapter
- Add Useful leaflet.js plugins chapter
- Add note about how book is a work in progress