Aufgabe: Kartenapp mit LeafletJS

Heute lernst du, wie man Code, den andere bereits mal geschrieben haben, für deine eigene Webseite verwenden und benutzen kannst. Solche Programme, die du für deine Entwicklung nutzen kannst, heißen *Library* (*Bibliothek*) oder *Framework* . Wir schauen uns heute leafletjs.com an um eine Webseite zu entwickeln.

Geh zuerst auf http://sciencecampinformatik.github.io/2016/ und schau dir auf der Map-Demo-Seite an, wie es aussehen soll.

Fremden Code zu benutzen ist meistens am Anfang sehr verwirrend. Dafür ist es umso großartiger, wenn es dann funktioniert und man plötzlich tolle, neue Funktionen hat.
Wie immer: Wenn Du irgendwo Probleme hast und nicht weiterkommst, frag uns einfach.

Erste Erfahrungen mit Leaflet JS

Leaflet ist ein sogennantes Open Source Projekt. Open Source Projekte sind Projekte, deren Code öffentlich ist und bei denen jeder, der will, mitarbeiten kann. In dieser Hinsicht ist es ein wenig wie Wikipedia, wo jeder der möchte Beiträge schreiben kann.

Bevor Du Leaflet in deiner Website benutzen kannst musst du den Leaflet-Code in deiner Website einbinden. Sonst weiß der Browser nicht, wo er den Code findet.

Binde Leaflet in deine Webseite ein

i. Die Dateien, die den Leaflet-Code enthalten befinden sich schon auf dem USB-Stick, den Du von uns bekommen hast. Du findest sie unter `bibliothek/leaflet`
ii. In diesem Ordner findest Du die Datei `leaflet.js`, die enthält allen Javascript Code von Leaflet.
iii. Außerdem gibt es ein Stylesheet `leaflet.css`, das für die Leaflet-Elemente ein Aussehen definiert.
iv. Die Datei `leaflet-src.js` brauchst Du nicht. (Sie enthält den selben Code wie `leaflet.js`, ist allerdings nicht komprimiert und daher noch lesbar)
v. Jetzt bist Du bereit Leaflet in den Header deiner Website einzubinden. Du brauchst den Javascript-Code und das CSS-Stylesheet. Das Einbinden der beiden Dateien funktioniert genau so, wie Du auch schon deine eigenen Javascript- und CSS-Dateien eingebunden hast: (es können beliebig viele Javascript- und CSS-Dateien in eine Website eingebunden werden)

```
<header>
  ...
  <!-- Binde den Leaflet-Javascript-Code ein -->
  <script src="bibliothek/leaflet/leaflet.js"/>
  <!-- Binde das Leaflet-Stylesheet ein -->
  <link rel="stylesheet" type="text/css" href="bibliothek/leaflet/leaflet.css
"/>
  ...
</header>
```

Leaflet sollte jetzt korrekt eingebunden sein und kann benutzt werden!

Jetzt brauchst Du zuerst einen Bereich auf deiner Website, in der die Karte angezeigt werden soll. Erstelle dazu in deiner HTML-Datei ein neues `div`-Element und gebe ihm die id “map”:

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

In diesem Bereich wird später die Karte auftauchen!

Da der eingebundene Javascript-Code von Leaflet von sich aus überhaupt nichts tut, brauchst Du noch ein bisschen eigenen Javascript-Code.
Auf der Startseite von leafletjs.com ist ein Beispiel Code gegeben.

Probiere den Beispiel Code aus

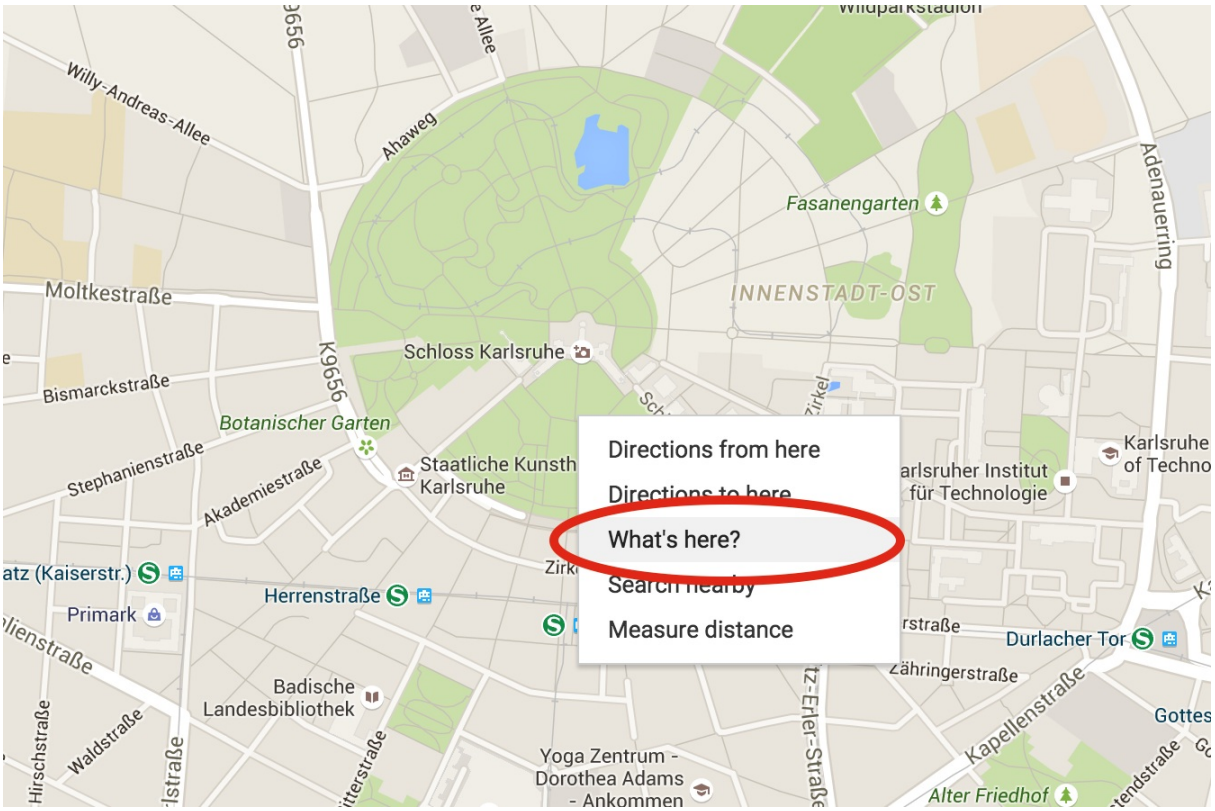
i. Erstelle eine Datei namens `map.js`, am besten im Unterordner `js`.
ii. Schreibe den Beispielcode in `map.js`.
iii. Binde `map.js` in deine Website hinter dem Karten-`div` Element ein:

```
<body>
  ...
  <div id="map">
  </div>
  <script src="js/map.js"/>
  ...
</body>
```

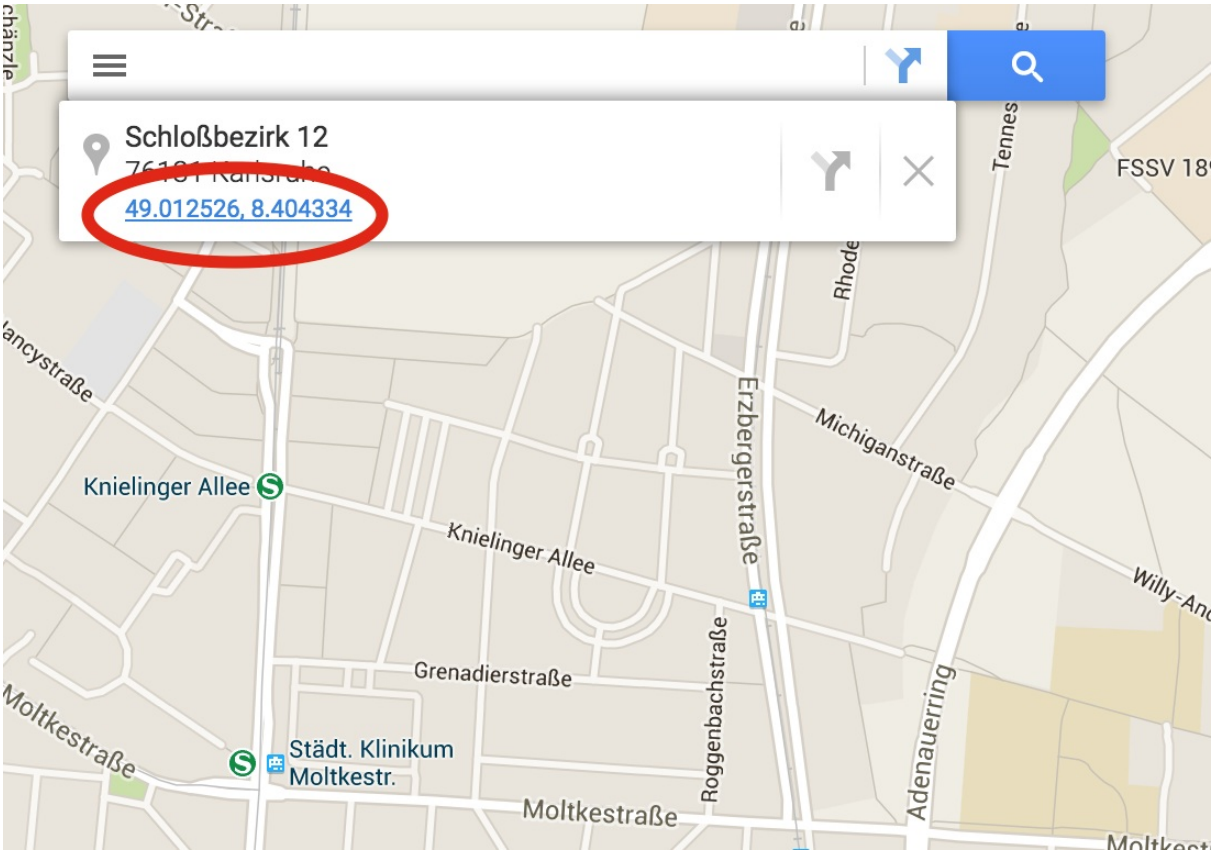
Jetzt sollte Deine Website eine Karte anzeigen!
Wenn das soweit funktioniert, wollen die Karte ein wenig anpassen.

Verändere den Code so, dass die Karte Karlsruhe anzeigt.

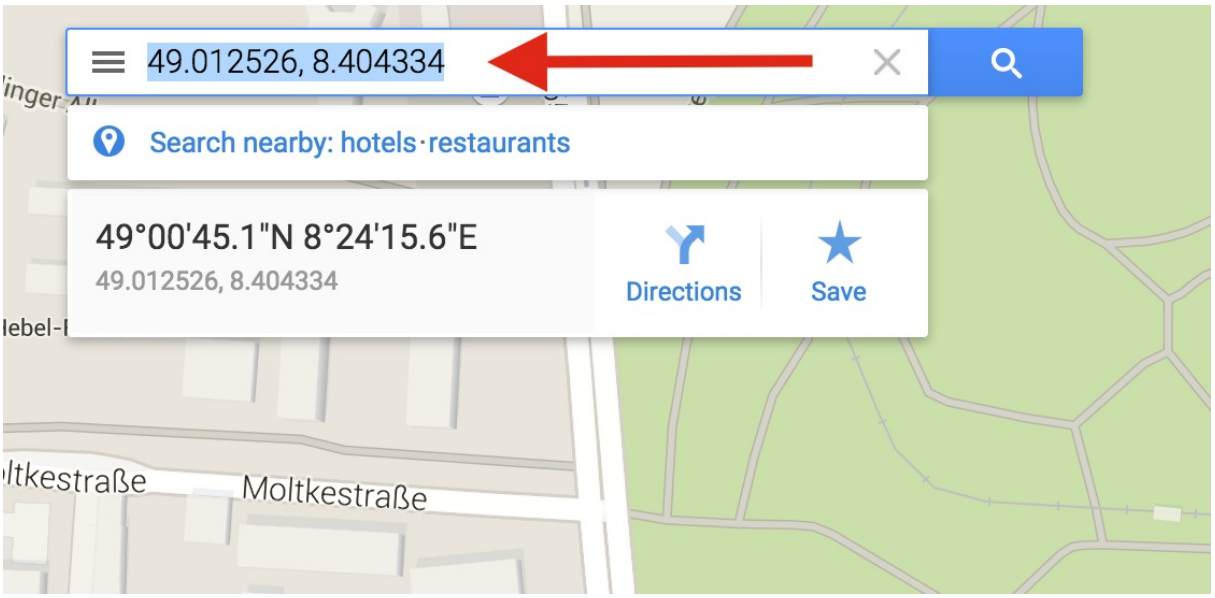
i. Lösche den Marker in London. Wir brauchen diesen nicht.
   - Schau dir den Beispielcode genau an.
   - Welcher Teil erstellt den Marker und das Popupfenster?
   - Wenn du es gefunden hast, entferne den Code.
ii. Zentriere nun die Karte auf Karlsruhe
   - Welche Funktion könnte dafür verantwortlich sein?
   - Suche in der Docs von Leaflet nach dieser Funktion und versuche zu verstehen, wie sie funktioniert. (Das ist nicht einfach zu verstehen. Frag nach Hilfe!)
iii. Um Leaflet sinnvoll zu benutzen brauchst du immer wieder Koordinaten von Orten. Du kannst dafür GoogleMaps benutzen
   - Mache einen Rechts-klick auf den Ort, dessen Koordinate du willst
   - Klicke auf *Whats' here?*



- Klicke auf die Koordinate, die nun am Suchfenster erscheint



- Kopiere dir die Koordinate



iv. Verändere nun den Code so, dass die Karte auf Karlsruhe zentriert ist.

Setze nun verschiedene Marker auf die Seite.

i. Schau dir nochmal den Code an, den wir vorhin gelöscht haben.
ii. Verwende ihn wieder im Marker auf die Karte zu setzen.
   - Falls du die gleichen Marker wie in der Demo haben möchtest, findest du hier die Koordinaten:
     - var infobau = [49.013817, 8.419790];
     - var schlosspark = [49.013149, 8.404364];
     - var klinikum = [49.015609, 8.374574];
     - var gameforge = [49.019921, 8.440989];
     - var his = [49.010194, 8.418930];

Tipp: Schreibe die Koordinaten nicht direkt in den Marker rein, sondern definiere erst Variablen mit den Koordinaten und benutze dann diese.

Navigationsangaben

Nun wollen wir auch die Route zwischen zwei Orten berechnen. Leaflet selbst bietet nicht die Möglichkeit, Routen zu berechnen. Daher benutzen wir wieder eine andere Library. Sie nennt sich *Leaflet Routing Machine*. Der Code der Library befindet sich ebenfalls bereits auf eurem USB-Stick unter `bibliothek/leaflet-routing-machine.

Binde Leaflet Routing Machine ein.

i. Gehe auf die Webseite http://www.liedman.net/leaflet-routing-machine/
ii. Suche den Abschnitt *Getting Started* und binde die Library entsprechend ein
   - Im Beispiel auf der Webseite wird zusätzlich zu Leaflet Routing Machine auf Leaflet eingebunden - da wir Leaflet bereits eingebunden haben, brauchen wir nur 2 der 4 Codezeilen. Welche sind das?
iii. Teste Leaflet Routing Machine, indem du den Beispielcode kopierst und bei dir im JavaScript-Code einfügst.
iv. Ändere den Beispielcode, sodass eine Route zwischen Markern berechnet wird, die du bereits gesetzt hast.

Verschiedene Routen anzeigen

Auf der Demoseite siehst du, dass in der 3. Version eine Route zwischen dem Infobau und einem anderen

Marker berechnet wird. Wenn du auf den Button “Nächste Route” klickst, wird eine andere Route startend vom Infobau angezeigt.

Versuche dies nachzubauen

i. Du brauchst einen Button, der eine Funktion in JavaScript aufruft.
ii. Schreibe diese Funktion, die aufgerufen wird. Lass die Funktion zunächst erstmal leer.
iii. Bestimme einen Marker, der immer dein Startpunkt sein wird.
iv. Erstelle ein Array, welches verschiedene Koordinaten hat, zu dem die Route berechnet wird.
v. Schreibe die Funktion, sodass immer ein neues Ziel gewählt wird.
vi. Erweitere die Funktion nun, sodass die Route zwischen Start und dem neuen Ziel berechnet wird.

Tipp: Ein neues Ziel im Array zu bestimmen, wenn man auf einen Button klickt, funktioniert sehr ähnlich wie in der Bildergalerie.

Sonstiges

Du kannst auch andere Kartendesigns für deine Webseite wählen. Diese Designs nennt man für Karten “Vector Tiles”. Normalerweise kosten Vector Tiles Geld, weil es sehr teuer ist, die vielen Bilder zu hosten. Einige sind allerdings kostenlos. Ihr dürft allerdings auf keinen Fall die Attribution dieser Tiles löschen.

Eine Attribution ist ein Hinweis, von wem die Tiles eigentlich kommen, die du benutzt. Außerdem wird in der Attribution erklärt unter welchen rechtlichen Bedingungen du die Vector Tiles benutzen darfst. Eine Attribution ist quasi ein fairer Tausch - du darfst die Vector Tiles kostenlos benutzen, dafür musst du aber auch angeben, von wem die Tiles kommen. Die Attribution siehst du unten rechts in deiner Karte.

Dies ist ein üblicher Vorgang in der Open Source Welt. Hier sind mögliche Tiles, die du benutzen darfst. Probiere sie aus und wähle dein Lieblingsdesign:

```
L.tileLayer('http://a.tile.openstreetmap.fr/hot/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
 contributors | Tiles by <a href="http://hot.openstreetmap.org/">HOT</a>'
}).addTo(map);
```

```
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
 contributors'
}).addTo(map);
```

```
L.tileLayer('http://otile1.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpg', {
    attribution: 'Tiles Courtesy of <a href="http://www.mapquest.com/" target
="_blank">MapQuest</a> <img src="http://developer.mapquest.com/content/osm/mq
_logo.png">, &copy; Data by <a href="http://openstreetmap.org">OpenStreetMap<
/a>, under <a href="http://www.openstreetmap.org/copyright">ODbL</a>.'
}).addTo(map);
```

```
L.tileLayer('http://c.tile.stamen.com/toner/{z}/{x}/{y}.png', {
    attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>,
 under <a href="http://creativecommons.org/licenses/by/3.0">CC BY 3.0</a>. Da
ta by <a href="http://openstreetmap.org">OpenStreetMap</a>, under <a href="ht
tp://www.openstreetmap.org/copyright">ODbL</a>.'
```

```
}).addTo(map);
```

```
L.tileLayer('http://c.tile.stamen.com/watercolor/{z}/{x}/{y}.png', {
    attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>,
 under <a href="http://creativecommons.org/licenses/by/3.0">CC BY 3.0</a>. Da
ta by <a href="http://openstreetmap.org">OpenStreetMap</a>, under <a href="ht
tp://www.openstreetmap.org/copyright">ODbL</a>.'
}).addTo(map);
```

Falls du nicht weiterkommst, kannst du uns gerne fragen oder auf der Demoseite von sciencecampinformatik.github.io/2016 nachschauen.