

Multimedia im Netz
Online Multimedia
Winter semester 2015/16

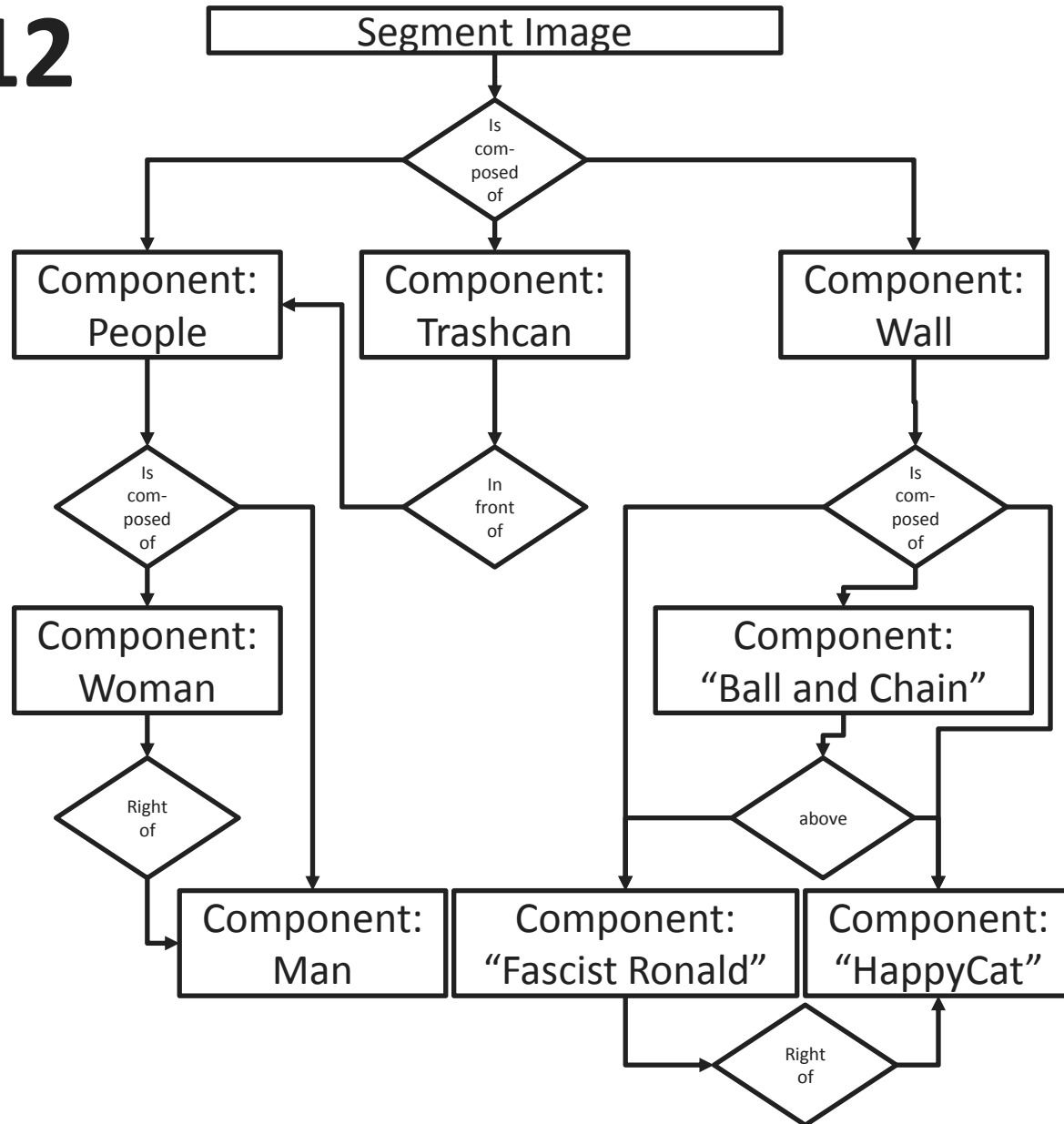
Tutorial 12 – Major Subject



Today's Agenda

- Imperative vs. Declarative programming
- WebComponents with Polymer
 - Getting Started / Code-Along
 - Using Components
 - Google Map Components
 - Databinding in Polymer
- Quiz

Assignment 12



Imperative vs. Declarative

- Imperative: Specify *how* to do something
- Declarative: Specify *what* should be done
other definition: a programming paradigm that expresses the logic of a computation without describing its control flow
- Often (not always), you'll find these concepts alongside declarative programming
 - Functional programming
 - Reactive programming
 - Databinding (see following slides)

<http://latentflip.com/imperative-vs-declarative/>

Example: Imperative Input Handling

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"><title>Imperative vs Declarative Event
  Handlers</title>
</head>
<body>
  <input id="name" placeholder="enter your name" />
  <button id="imperative" disabled>OK</button>
  <div>Your name: <span id="output"></span></div>

  <script>

  var userName = '';
  var imperativeButton = document.getElementById('imperative');

  function updateUI(){
    document.getElementById('output').innerHTML = userName;
    imperativeButton.disabled = userName.length == 0;
  }

  document.getElementById('name').addEventListener('input',function(){
    userName = this.value;
    updateUI();
  });
  </script>
</body></html>
```

Example: Imperative Input Handling

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"><title>Imperative vs Declarative Event
  Handlers</title>
</head>
<body>
  <input id="name" placeholder="enter your name" />
  <button id="imperative" disabled>OK</button>
  <div>Your name: <span id="output"></span></div>

  <script>

  var userName = '';
  var imperativeButton = document.getElementById('imperative');

  function updateUI(){
    document.getElementById('output').innerHTML = userName;
    imperativeButton.disabled = userName.length == 0;
  }

  document.getElementById('name').addEventListener('input', function(){
    userName = this.value;
    updateUI();
  });
  </script>
</body></html>
```

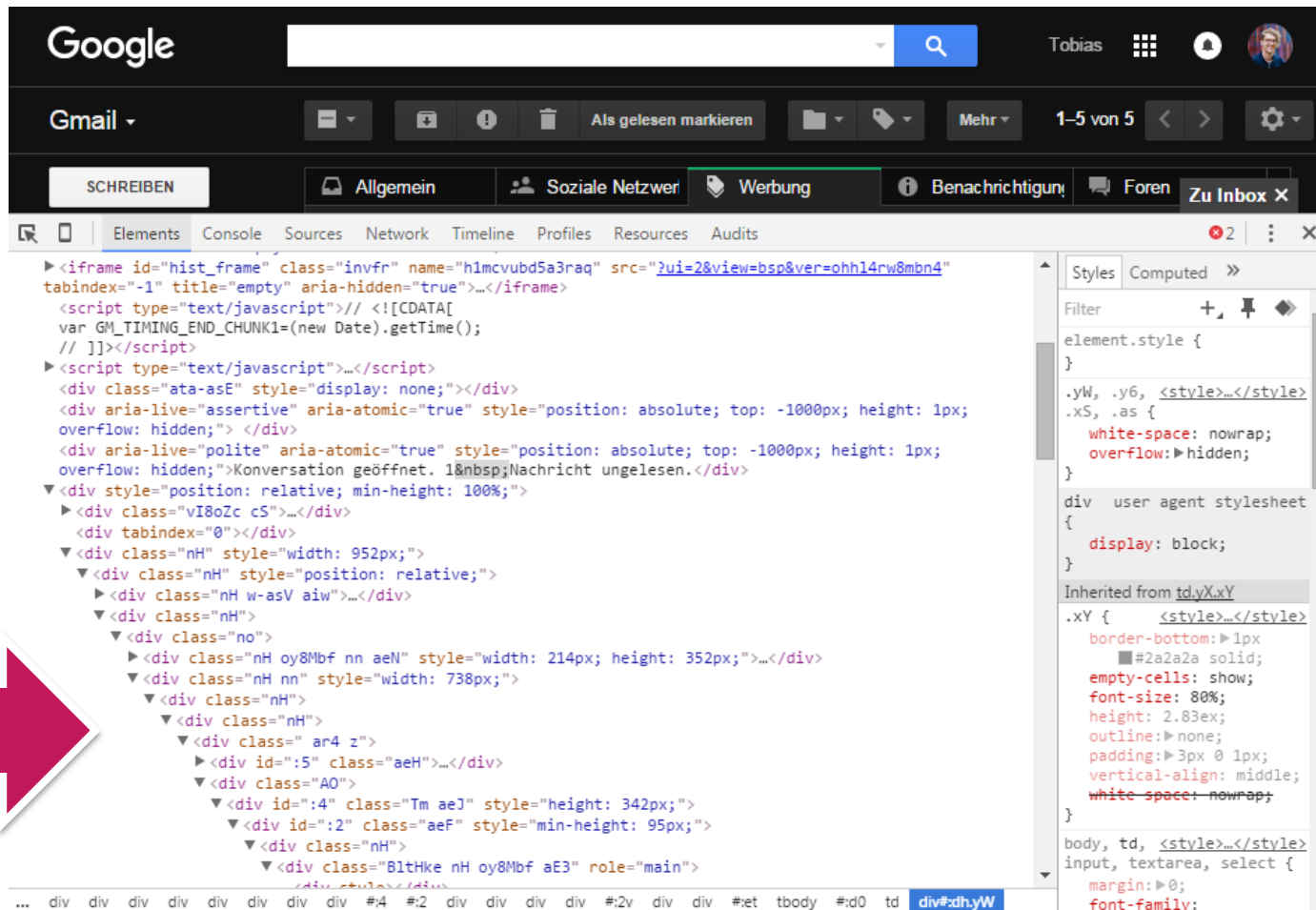
Declarative Equivalent with AngularJS (1.4)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Declarative Example</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/an
gular.min.js"></script>
</head>
<body ng-app>
  <input placeholder="enter your name" ng-model="userName"/>
  <button ng-disabled="!userName">OK</button>
  <div>Your name: <span id="output">{{userName}}</span></div>
</body>
</html>
```

Databinding

- Model-View-Binder:
 - Variant of the Model-View-ViewModel Pattern
 - Goal: Simplify event driven programming
 - Separation of user interface and business logic
 - More declarative programming
- Declarative aspect: Declare ***what*** data you want in the UI, rather than *how* to get it.
- Many JavaScript frameworks currently are driven by this paradigm.
 - Angular
 - Polymer
 - KnockoutJS
 - ReactJS

A Problem



The screenshot shows a Google search page with the Chrome DevTools console and styles pane open. The console displays the following HTML structure:

```
>>> <iframe id="hist_frame" class="invfr" name="h1mcvubd5a3raq" src="ui=2&view=bsp&ver=ohh14rw8mbn4"
tabindex="-1" title="empty" aria-hidden="true">...</iframe>
  <script type="text/javascript">// <![CDATA[
    var GM_TIMING_END_CHUNK1=(new Date).getTime();
    // ]]></script>
  <script type="text/javascript"></script>
  <div class="ata-asE" style="display: none;"></div>
  <div aria-live="assertive" aria-atomic="true" style="position: absolute; top: -1000px; height: 1px;
overflow: hidden;"></div>
  <div aria-live="polite" aria-atomic="true" style="position: absolute; top: -1000px; height: 1px;
overflow: hidden;">Konversation geöffnet. 1&nbsp;Nachricht ungelesen.</div>
  <div style="position: relative; min-height: 100%;">
    >>> <div class="vI8oZc cS">...</div>
    >>> <div tabindex="0"></div>
    >>> <div class="nH" style="width: 952px;">
      >>> <div class="nH" style="position: relative;">
        >>> <div class="nH w-asV aiw">...</div>
        >>> <div class="nH">
          >>> <div class="no">
            >>> <div class="nH oy8Mbf nn aeN" style="width: 214px; height: 352px;">...</div>
            >>> <div class="nH nn" style="width: 738px;">
              >>> <div class="nH">
                >>> <div class="nH">
                  >>> <div class=" ar4 z">
                    >>> <div id=":5" class="aeH">...</div>
                    >>> <div class="AO">
                      >>> <div id=":4" class="Tm aeJ" style="height: 342px;">
                        >>> <div id=":2" class="aeF" style="min-height: 95px;">
                          >>> <div class="nH">
                            >>> <div class="Blthke nH oy8Mbf aeE" role="main">
                              >>> <div style="border-bottom: 1px solid #2a2a2a; font-size: 80px; height: 2.83ex; padding: 3px 0 1px; vertical-align: middle; white-space: nowrap;">
                                ...
                              </div>
                            </div>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

The styles pane on the right shows the following styles for the selected element:

```
element.style {
}
.yW, .y6, <style>...</style>
.xS, .as {
  white-space: nowrap;
  overflow: hidden;
}
div user agent stylesheet
{
  display: block;
}
Inherited from td.yX.yY
.yY {
  border-bottom: 1px solid #2a2a2a;
  font-size: 80px;
  height: 2.83ex;
  padding: 3px 0 1px;
  vertical-align: middle;
  white-space: nowrap;
}
body, td, <style>...</style>
input, textarea, select {
  margin: 0;
  font-family:
```

A large red arrow points to the console output, highlighting the HTML structure.

A Solution: Custom HTML Elements

```
<hangout-module>
  <hangout-chat from="Paul, Addy">
    <hangout-discussion>
      <hangout-message from="Paul" profile="profile.png"
        datetime="2013-07-17T12:02">
        <p>Feelin' this Web Components thing.</p>
        <p>Heard of it?</p>
      </hangout-message>
    </hangout-discussion>
  </hangout-chat>
  <hangout-chat>...</hangout-chat>
</hangout-module>
```

<http://www.html5rocks.com/en/tutorials/webcomponents/customelements/>

Web Components



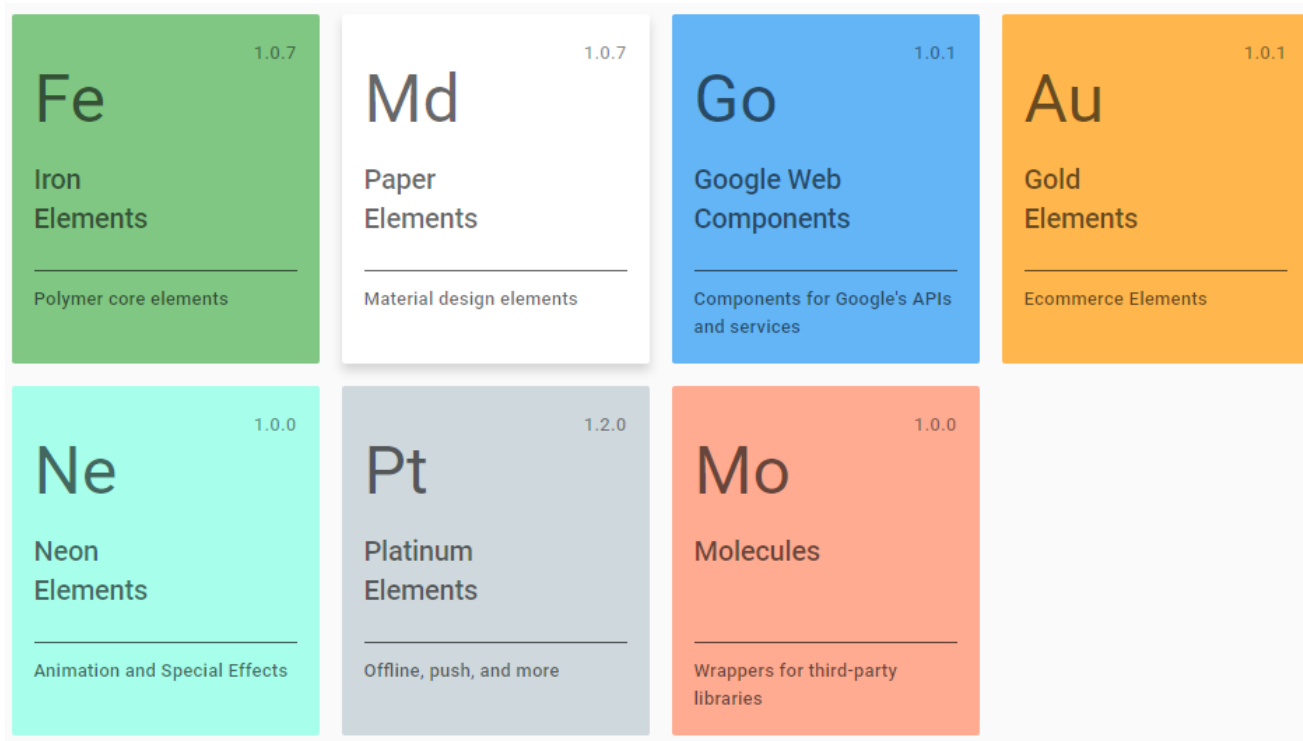
- Approach to more declarative web programming style.
- Goal: re-use “things”, that we would have to write over and over (reducing boilerplate code on the web)
- Driven by Google and also Mozilla ([x-tag](#))
- Concepts:
 - Custom Elements
 - HTML Imports
 - Templates
 - Shadow DOM

Image: <http://webcomponents.org/>

Polymer



- Material Design
- Large library of custom elements (the [Element Catalog](#))



Getting Started with Polymer: Tools

- **Option A: Your favorite Text Editor / IDE**
 - Install [bower](#)
 - Use the `bower.json` file we provide on GitHub
 - Run `bower install`
 - Create an html file and start working with Polymer
- **Option B: Use Chrome Dev Editor**
 - Download here: <https://chrome.google.com/webstore/detail/chrome-dev-editor-develop/pnoffddplpippgcfjdhbmhkofpnaalpg>
 - Create a new project, use the template “**Javascript web app (using Polymer paper elements)**”
 - Wait until bower dependencies are set up
 - Run the project. See changes in the browser immediately.



Using Custom Web Components

```
<!doctype html>
<html>
<head>
  <title>Polymer Basics</title>
  <script
    src="bower_components/webcomponentsjs/webcomponents-lite.min.js">
  </script>

  <link rel="import"
    href="bower_components/paper-button/paper-button.html">

  <link rel="stylesheet" href="styles.css">
</head>

<body>

<paper-button raised>Hi there!</paper-button>

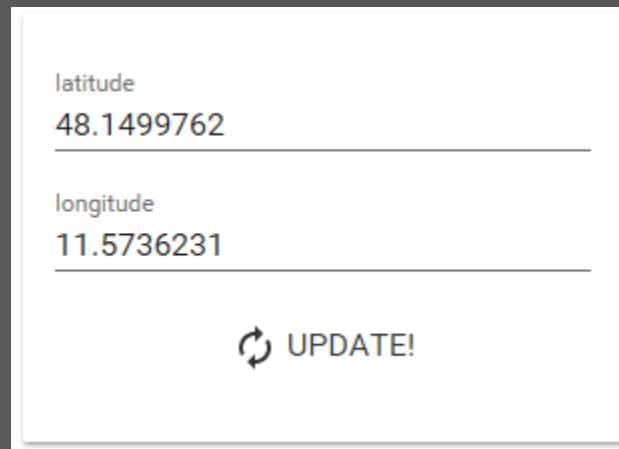
</body></html>
```

Goal of today's Code-Along



Breakout: Setup - More components

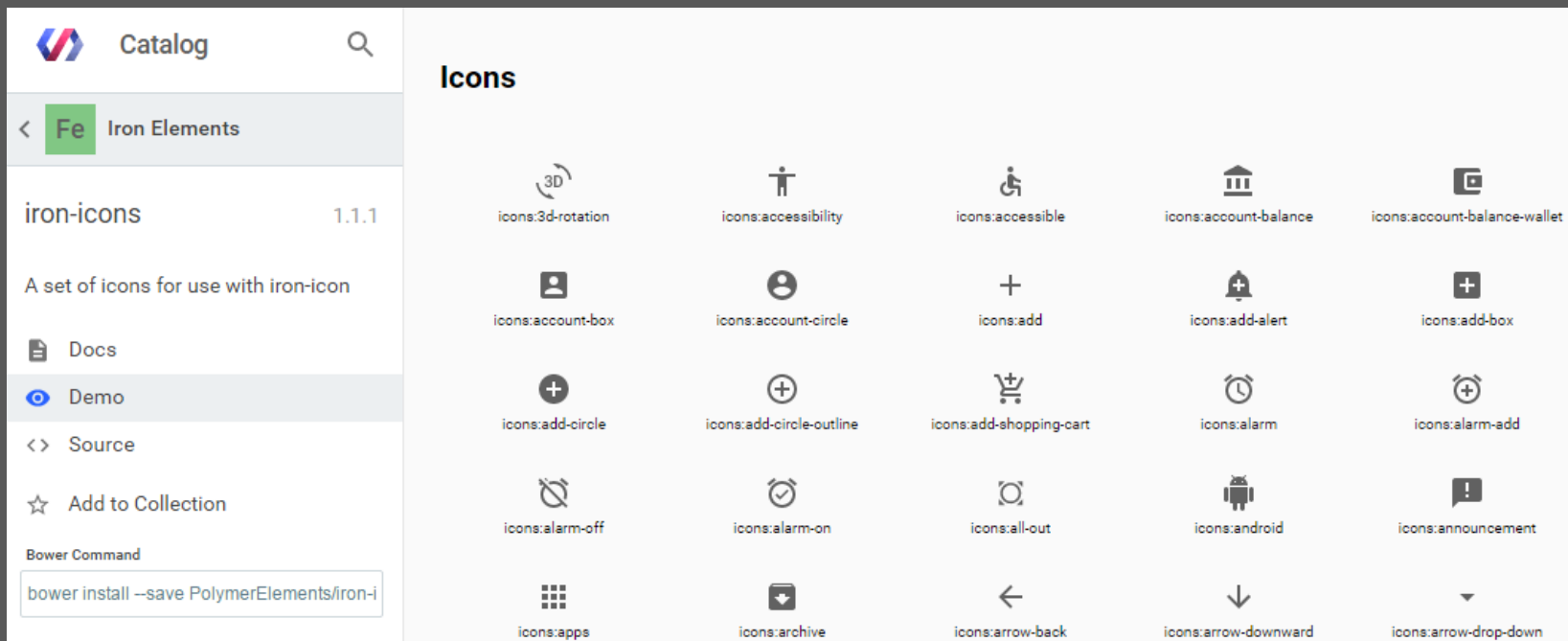
- Add the following:
 - `<paper-card>`
 - `<paper-input>`
 - `<iron-icons>`
- Create a box with those four components:



A screenshot of a web form with a white background and a thin grey border. It contains two input fields. The first field is labeled "latitude" and contains the value "48.1499762". The second field is labeled "longitude" and contains the value "11.5736231". Below the input fields is a button with a circular refresh icon and the text "UPDATE!".

Resources

- Each component is documented, e.g. iron-icons
<https://elements.polymer-project.org/elements/iron-icons>
- To see the component in action, click on “Demo” on the left:



The screenshot displays the Polymer Elements Catalog interface. On the left, a sidebar shows the navigation menu with 'Fe Iron Elements' selected, and 'Demo' highlighted. Below the menu, the 'iron-icons' component is listed with version '1.1.1' and a description: 'A set of icons for use with iron-icon'. A 'Bower Command' section shows the command: `bower install --save PolymerElements/iron-i`. The main content area, titled 'Icons', displays a grid of 25 icons, each with a label such as 'icons:3d-rotation', 'icons:accessibility', 'icons:accessible', 'icons:account-balance', 'icons:account-balance-wallet', 'icons:account-box', 'icons:account-circle', 'icons:add', 'icons:add-alert', 'icons:add-box', 'icons:add-circle', 'icons:add-circle-outline', 'icons:add-shopping-cart', 'icons:alarm', 'icons:alarm-add', 'icons:alarm-off', 'icons:alarm-on', 'icons:all-out', 'icons:android', 'icons:announcement', 'icons:apps', 'icons:archive', 'icons:arrow-back', 'icons:arrow-downward', and 'icons:arrow-drop-down'.

Sample Solution...

```
<paper-card>
  <paper-input
    label="latitude"
    type="number"
    value="48.1499762"></paper-input>
  <paper-input
    label="longitude"
    type="number"
    value="11.5736231"></paper-input>
  <paper-button>
    <iron-icon icon="autorenew"></iron-icon>
    Update!
  </paper-button>
</paper-card>
```

Adding a Map Component

- Modify the bower.json file:

```
"dependencies": {  
  "iron-elements": "PolymerElements/iron-elements#^1.0.7",  
  "paper-elements": "PolymerElements/paper-elements#^1.0.7",  
  "google-map": "GoogleWebComponents/google-map#~1.1.7"  
}
```

- Those of you who use bower can run
`bower install --save GoogleWebComponents/google-map`

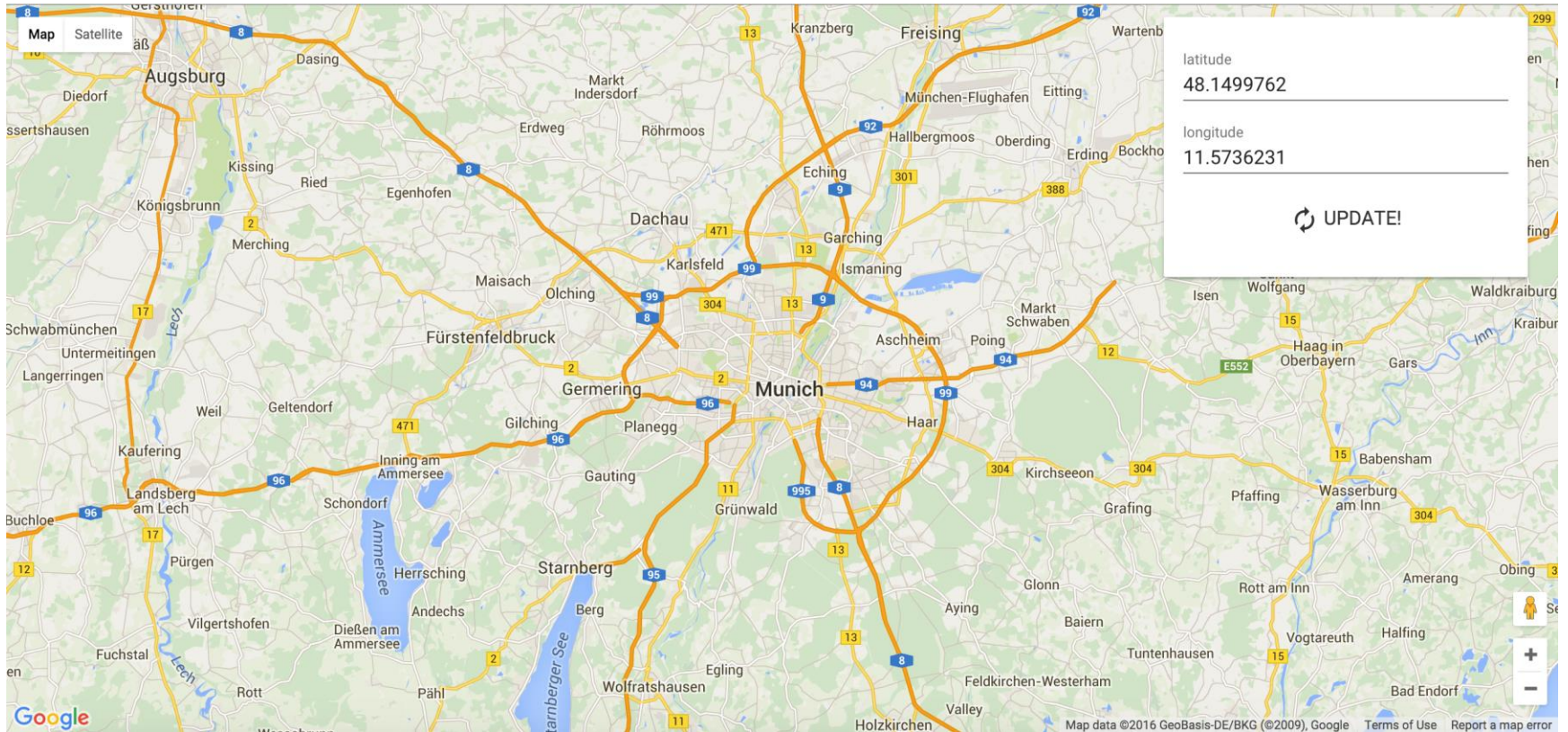
- Include and use these components:

```
<link rel="import"  
      href="bower_components/google-map/google-map.html">  
<link rel="import"  
      href="bower_components/google-map/google-map-marker.html">
```

- Create the element:

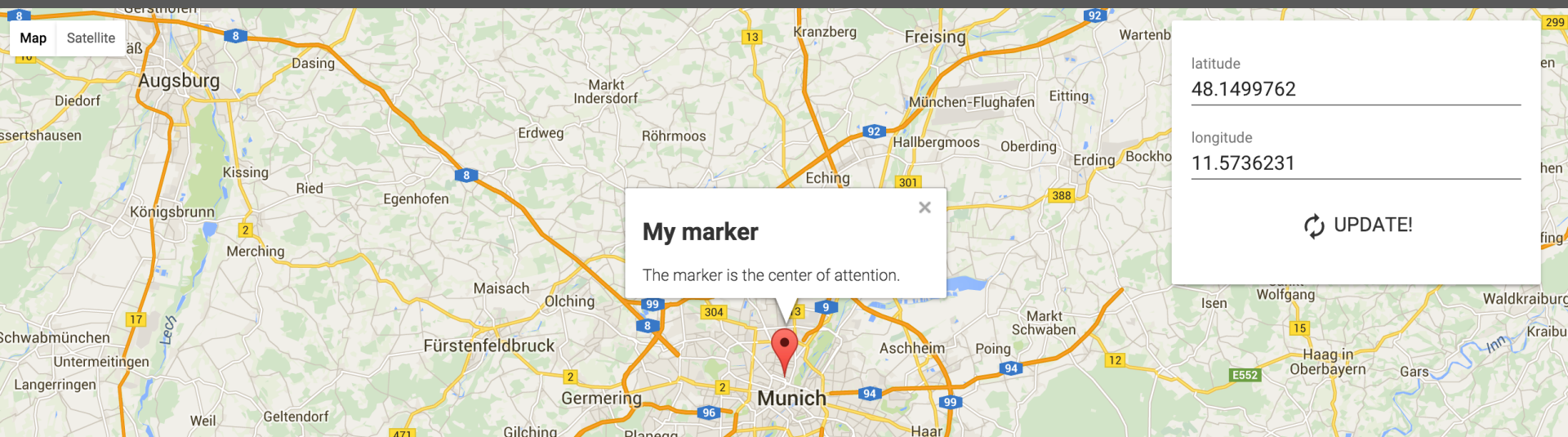
```
<google-map  
  latitude="48.1499762"  
  longitude="11.5736231">  
</google-map>
```

Result: A Map!



Breakout: Add a <google-map-marker>

- Use the <google-map-marker> component to display a marker on the map
 - use coordinates that are shown on the current map (48.1499762, 11.5736231)
 - Create a <h2> and <div> inside it. Those are displayed inside the info window.



Sample Solution

```
<google-map
  latitude="48.1499762"
  longitude="11.5736231">
  <google-map-marker latitude="48.1499762"
    longitude="11.5736231">
    <h2>My marker</h2>
    <div>
      The marker is the
      center of attention.
    </div>
  </google-map-marker>
</google-map>
```

Breakout: Event Listeners

- Attach an event listener for the “tap” event to the update button.
- Adjust the latitude and longitude of the map depending on the values inside the <paper-input> elements
- Adjust the latitude and longitude of the <google-map-marker>

Sample Solution

```
<script>

var marker = document.querySelector('google-map-marker'),
    button = document.querySelector('paper-button'),
    latitudeInput = document.getElementById('latitude'),
    longitudeInput = document.getElementById('longitude'),
    googleMap = document.querySelector('google-map');

button.addEventListener('tap', function(){
    var latitude = latitudeInput.value;
    var longitude = longitudeInput.value;

    googleMap.latitude = latitude;
    googleMap.longitude = longitude;

    marker.latitude = latitude;
    marker.longitude = longitude;
});

</script>
```


Thoughts on Sample Solution

- Imperative programming paradigm
 - Components have certain states which we take care of
→ latitude/longitude
 - We specify **how** the events should be handled, i.e. each single step.
- JavaScript is required
- Goal: more declarative way of updating the map and marker.

Databinding in Polymer

- Similar to what we saw earlier in AngularJS
- Advantage: more declarative, no JavaScript on our side necessary (but it is used under the hood)
- **Square brackets [[]]**: One-way databinding
 - host notifies target
- **Curly brackets {{ }}**: Automatic databinding
 - host notifies target (downwards)
 - target notifies host (upwards), if target allows this.
- Databinding needs a binding scope:
`<template is="dom-bind">`
Everything in here can use this databinding scope.
`</template>`

<https://www.polymer-project.org/1.0/docs/devguide/data-binding.html>

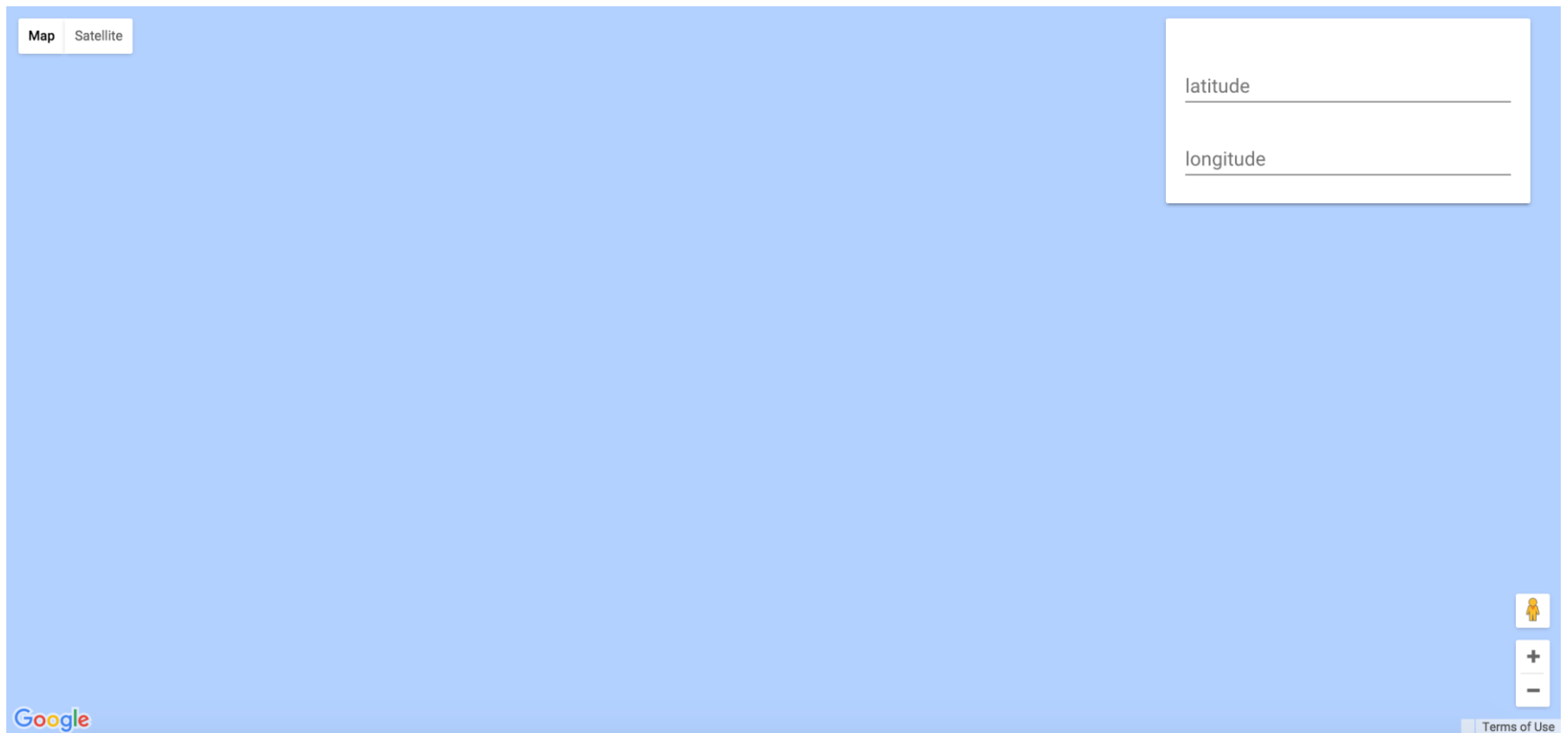
Modifying our working sample (1)

```
<paper-card>
  <paper-input
    label="latitude"
    type="number"
    value="{{latitude}}"
    id="latitude"
  ></paper-input>
  <paper-input
    label="longitude"
    type="number"
    value="{{longitude}}"
    id="longitude"
  ></paper-input>
</paper-card>
```

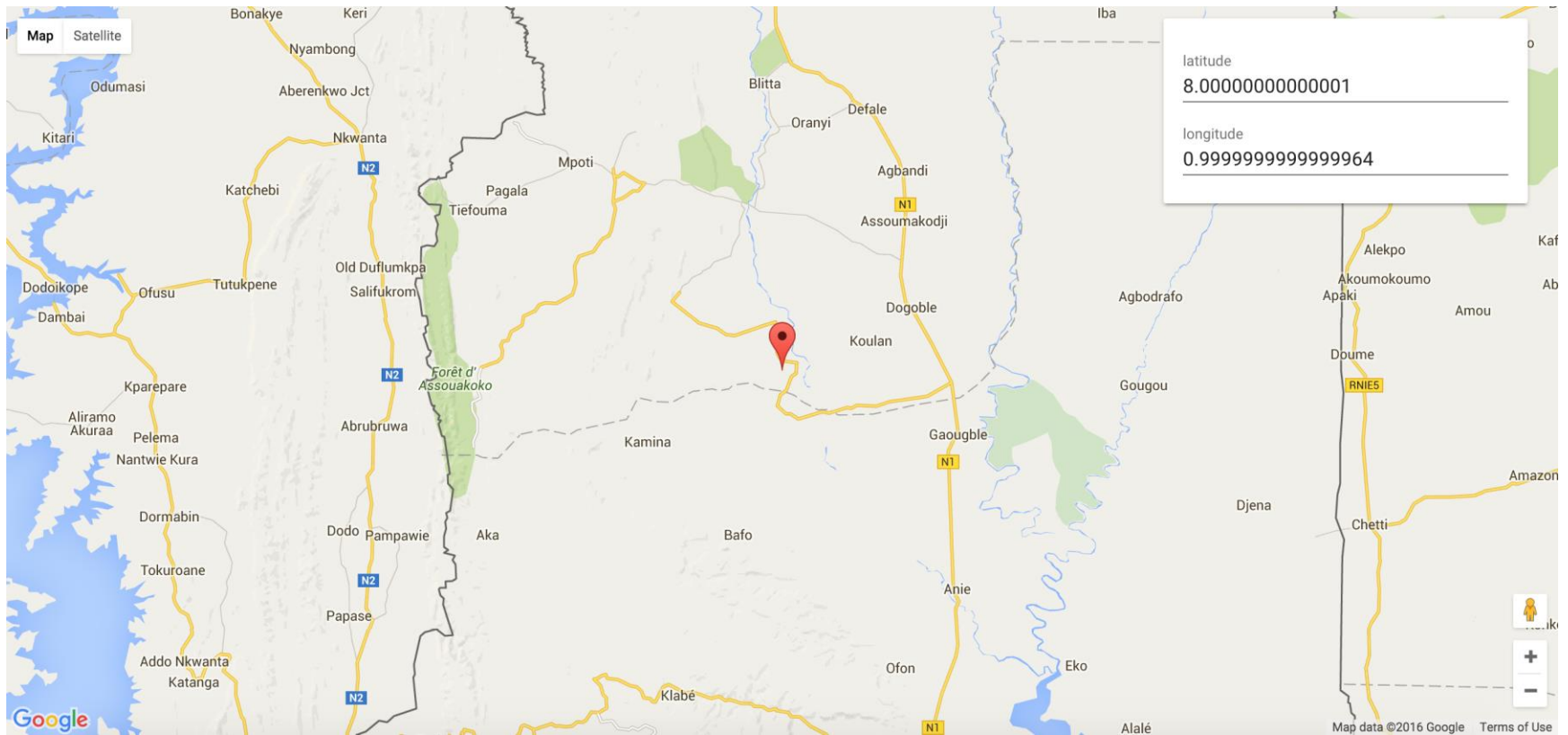
Modifying our working sample (2)

```
<google-map
  latitude="{{latitude}}"
  longitude="{{longitude}}">
  <google-map-marker latitude="{{latitude}}"
    longitude="{{longitude}}"
    animation="DROP">
    <h2>My marker</h2>
    <div>The marker is the
      center of attention.
    </div>
  </google-map-marker>
</google-map>
```

Result...

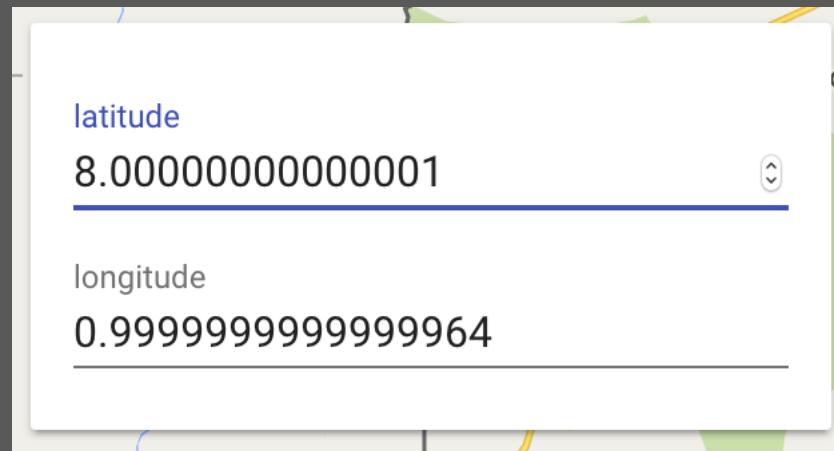


Result after typing inside the input boxes



Breakout: Fix Rounding Issue

- If you type in the <paper-input> fields, you'll notice that the value is automatically adjusted, preventing you from typing more than one digit:



- Fix this problem. Do not use JavaScript.

Breakout: Extend the Interaction

- Re-establish the default values for the marker.
 - give the <template> an id="app".
 - the latitude and longitude are accessible as properties from the template
 - You *do* need JavaScript for that.
- Make the marker draggable without using JavaScript.

Assignment

Menu

Music Library

Artists

Albums

Playlists

 Browse

 Radio

Artists



AC/DC



Andrew W.K.



Buddy Guy



<https://youtu.be/3fTV5JnPntw>

Assignment

- Create a small music-library webpage using lots of elements from the polymer element catalog.
- To create the solution shown in the video we used:
iron-flex-layout, iron-icons, iron-icons/av-icons, iron-image, iron-pages, paper-button, paper-card, paper-drawer-panel, paper-header-panel, paper-icon-button, paper-input, paper-item/paper-icon-item, paper-listbox, paper-menu, paper-styles/default-theme, paper-styles/typography, paper-tabs, paper-tabs/paper-tab, paper-toolbar



Round-Up Quiz

1. Which of these languages are 100% declarative?
 1. HTML
 2. JavaScript
 3. Java
 4. SQL?
2. What is the correct binding annotation for one-way binding?
3. Which 4 concepts do WebComponents encompass?

Thanks!

What are your questions?

Links

- <https://remysharp.com/2010/10/08/what-is-a-polyfill>
- <http://w3c.github.io/webcomponents/explainer/>
- Polymer tutorials on YouTube:
<https://www.youtube.com/playlist?list=PLLnPHn493BHGhoGAb2PRKzv4Zw3QoatK->
- More details on declarative programming:
<https://www.youtube.com/watch?v=XSeMyqoMNNk>
- Motivation for declarative programming (python):
<https://www.youtube.com/watch?v=nRDC6GtfB4g>