Multimedia im Netz

Online Multimedia

Winter semester 2015/16

Tutorial 13 – Major Subject

Today's Agenda

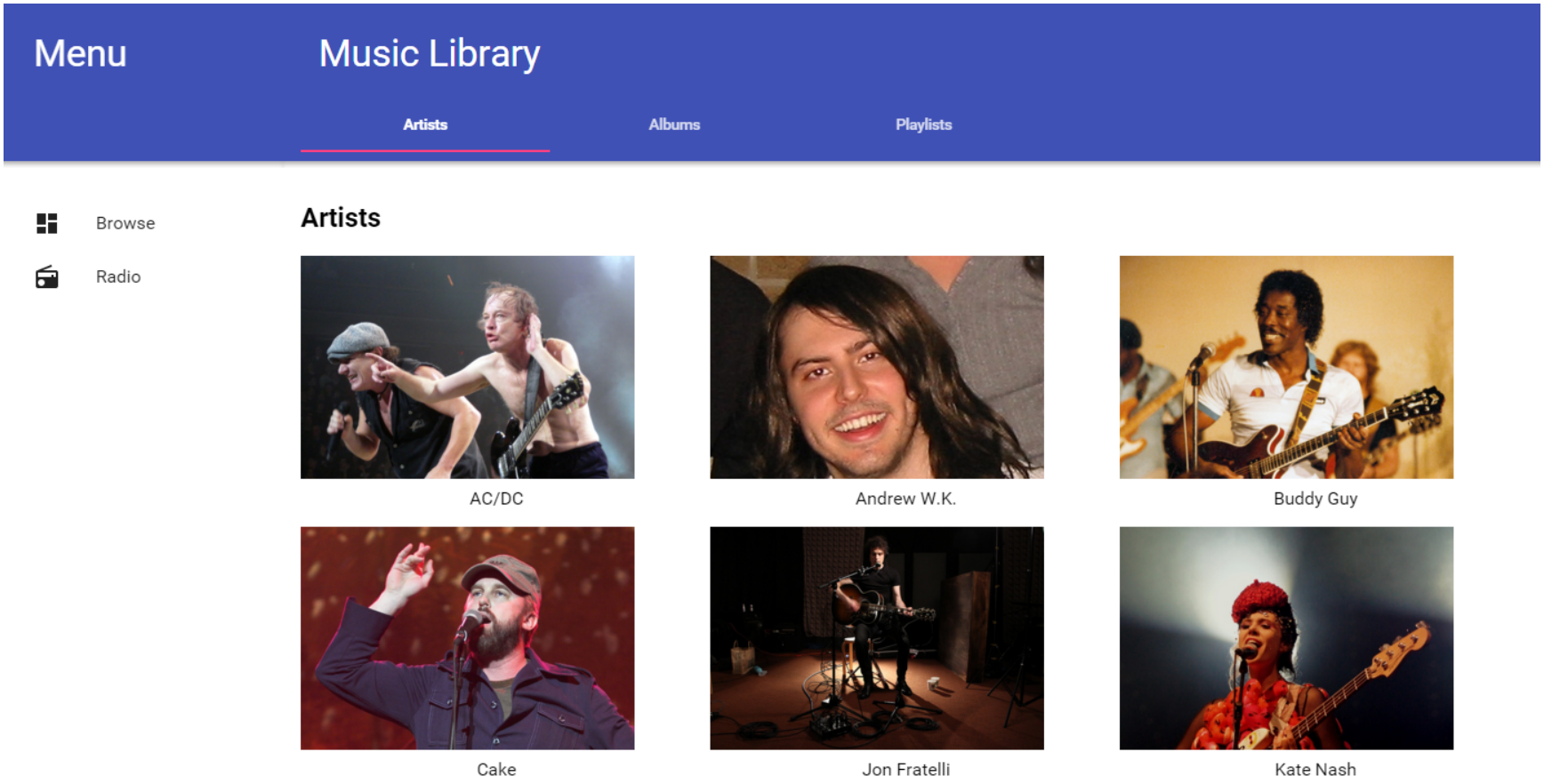
- Custom Elements with Polymer
  - Structure
  - Usage
- Advanced Databinding
  - Binding to object properties
  - Template helper elements
- AJAX with Polymer
- Quiz

Last Week...

- What is databinding?
- What syntax do you need to achieve databinding?
- What four concepts do Web-Components cover?

Custom Elements

Assignment 12



Assignment 12 - Approach

```
<section>
    <h1>Artists</h1>
    <div class="artists">
        <div class="artist">
            <iron-image src="eodm.jpg"></iron-image>
            <span class="name">EODM</span>
        </div>
        <div class="artist">
            <iron-image src="ramones.jpg"></iron-image>
            <span class="name">Ramones</span>
        </div>
        <div class="artist">
            <iron-image src="kinks.jpg"></iron-image>
            <span class="name">Kinks</span>
        </div>
    </div>
</section>
```

Goal: Custom Elements

```
<section>
  <h1>Artists</h1>
  <div class="artists">

    <my-artist name="eodm" img="eodm.jpg"></my-artist>

    <my-artist name="ramones" img="ramones.jpg"></my-artist>

    <my-artist name="kinks" img="kinks.jpg"></my-artist>

  </div>
</section>
```

Custom Elements

- Advantages:
  - More declarative approach than using nested classes etc.
  - Encapsulation
  - Separation of concerns
  - Readable

- **Custom element names need to contain a dash (-), e.g. my-artist, some-thing, cool-dude**

- We want to be able to create a re-usable “artist” component

Basic Structure - my-artist.html

```
<link rel="import"
      href="../bower_components/polymer/polymer.html">

<dom-module id="my-artist">
    <template>
        <style></style>
        <div>
            Some Content
        </div>
    </template>
    <script>
        Polymer({
            is : 'my-artist'
        })
    </script>
</dom-module>
```

Basic Structure - my-artist.html

```
<link rel="import"
      href="../bower_components/polymer/polymer.html">

<dom-module id="my-artist">
    <template>
        <style></style>
        <div>
            Some Content
        </div>
    </template>
    <script>
        Polymer({
            is : 'my-artist'
        })
    </script>
</dom-module>
```

Basic Structure - my-artist.html

```
<link rel="import"
      href="../bower_components/polymer/polymer.html">

<dom-module id="my-artist">
    <template>
        <style></style>
        <div>
            Some Content
        </div>
    </template>
    <script>
        Polymer({
            is : 'my-artist'
        })
    </script>
</dom-module>
```

Using the <my-artist> element

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Using Custom Elements</title>
  <script src="../bower_components/webcomponentsjs/webcomponents-
                  lite.min.js">
  </script>
  <link rel="import" href="my-artist.html">

</head>
<body>

  <my-artist></my-artist>

</body>
</html>
```

Databinding & Styling

```
<dom-module id="my-artist">
    <template>
        <style>
            :host{
                color: red;
            }
        </style>
        <div>{{content}}</div>
    </template>
    <script>
        Polymer({
            is : 'my-artist',
            ready : function(){
                this.content = 'Oops, I bound it again.'
            }
        })
    </script>
</dom-module>
```

Declared Properties

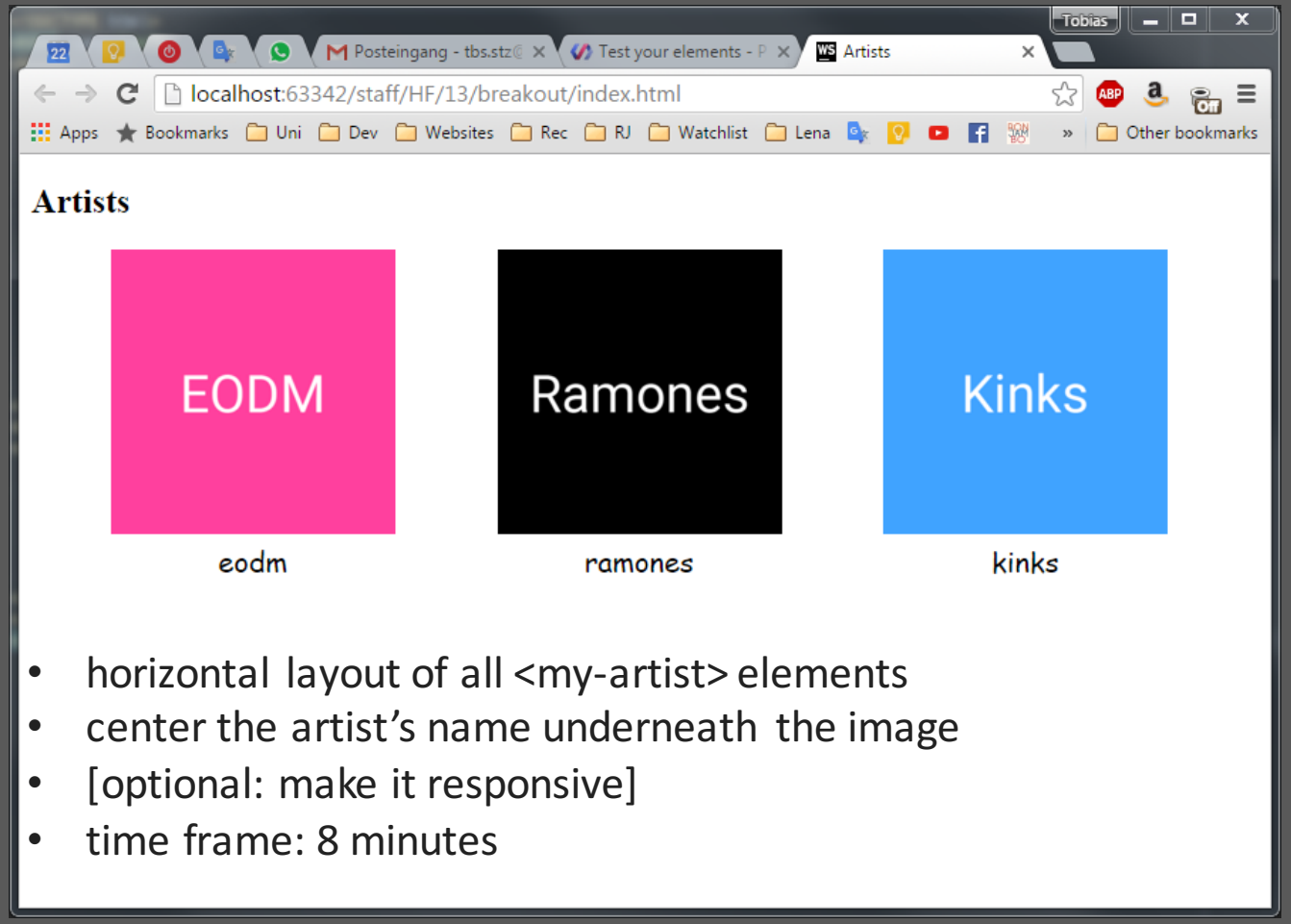
```
<link rel="import"
      href="../bower_components/polymer/polymer.html">
<link rel="import"
      href="../bower_components/iron-image/iron-image.html">

<dom-module id="my-artist">
    <template>
        <iron-image src="{{img}}"></iron-image>
        <span>{{name}}</span>
    </template>
    <script>
        Polymer({
            is : 'my-artist',
            properties : {
                name : String,
                img : String
            }
        })
    </script>
</dom-module>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Artists</title>
    <script
       src="../bower_components/webcomponentsjs/webcomponents-lite.min.js">
    </script>
    <link rel="import" href="../bower_components/polymer/polymer.html">
    <link rel="import" href="my-artist.html">
</head><body>
<section>
    <h1>Artists</h1>
    <div class="artists">
        <my-artist name="eodm" img="eodm.png"></my-artist>
        <my-artist name="ramones" img="ramones.png"></my-artist>
        <my-artist name="kinks" img="kinks.png"></my-artist>
    </div>
</section>
</body></html>
```

declared properties → element attributes

Breakout: Styling



- horizontal layout of all <my-artist> elements
- center the artist's name underneath the image
- [optional: make it responsive]
- time frame: 8 minutes

Advanced Databinding

Binding to sub-properties

- Polymer allows us to bind output to object properties or sub-properties
- Implication: if the property changes, but not the entire object, all bound content will be updated
- Example:

```
<template is="dom-bind" id="app">
    <div>Position: <span>{{person.position}}</span></div>
    <div>First: <span>{{person.first}}</span></div>
    <div>Last: <span>{{person.last}}</span></div>
</template>
<script>
    var app = document.querySelector('#app');
    app.person = {
        first : 'Jake',
        last : 'Peralta',
        position: 'Detective'
    };
</script>
```

https://www.polymer-project.org/1.0/docs/devguide/data-binding.html#binding-to-sub-properties

Databinding Helper Element: DOM repeat

- `<template is="dom-repeat">` binds to an array:

```
<template is="dom-repeat" items="{{employees}}">
  <div>First name: <span>{{item.first}}</span></div>
  <div>Last name: <span>{{item.last}}</span></div>
</template>
```

`employees` in this case is an array like this:

```
[{first:'Hank',last:'Williams'},
 {first:'Sara',last:'Monah'}, …]
```

- Simplifies iteration, reduces mark-up
- Similar to “foreach” syntax → foreach **item** in **array**

Artist example

```
<template is="dom-bind" id="app">
  <h1>Artists</h1>
  <div class="artists">
    <template is="dom-repeat" items="{{artists}}">
       <my-artist name="{{item.name}}" img="{{item.img}}"></my-artist>
    </template>
  </div>
</template>
<script>
    var app = document.querySelector('#app');
    app.artists = [
        {name: 'EODM', img: 'eodm.png'},
        {name: 'Ramones', img: 'ramones.png'},
        {name: 'Kinks', img: 'kinks.png'}
    ]
</script>
```

Dom-repeat: named repeats with “as”

- Inside the dom-repeat template, the current item is bound to the variable name `item`
- We can change that with the **`as`** property of the template:

```
<template is="dom-repeat"
          items="{{artists}}"
          as="artist">
    <my-artist
            name="{{artist.name}}"
            img="{{artist.img}}"></my-artist>
</template>
```

for arrays like this:

```
app.artists = [
    {name: 'EODM', img: 'eodm.png'},
    {name: 'Ramones', img: 'ramones.png'},
    {name: 'Kinks', img: 'kinks.png'}
]
```

AJAX with Polymer

The <iron-ajax> component

- Declarative approach to AJAX: specify ***what*** you expect from the server, instead of how to get it
- Component: **<iron-ajax>** with the most prevalent attributes:
  - `url`: the target URL of the request
  - `method`: the HTTP method, default GET
  - `params`: GET parameters in JSON format.
  - `auto`:
    if this attribute is set, the request will fire if parameters change or
  - `lastResponse`:
    contains the response object of the last request
- By the way: all camelCase properties need to be used with a dash inside markup: `lastResponse` → `last-response`

<iron-ajax> Basic Example

ajax-basic.html

```
<template is="dom-bind">
  <iron-ajax
     auto
     url="artists.json
     last-response="{{artists}}">
  </iron-ajax>

  <template
     is="dom-repeat"
     items="{{artists}}">

     <div>
       <img src="{{item.img}}"
            width="100">
       <span>{{item.name}}</span>
     </div>

  </template>
</template>
```

artists.json

```
[
  {
    "name": "EODM",
    "img": "eodm.png"
  },
  {
    "name": "Kinks",
    "img": "kinks.png"
  },
  {
    "name": "Ramones",
    "img": "ramones.png"
  }
]
```

<iron-ajax> Example: Spotify Search

```
<template is="dom-bind">
  <iron-ajax
     auto
     url="https://api.spotify.com/v1/search"
     params='{"type":"artist", "q":"eagles"}'
     last-response="{{spotifyResponse}}"></iron-ajax>

     <template
       is="dom-repeat"
       items="{{spotifyResponse.artists.items}}">

                 <div>{{item.name}}</div>

     </template>
</template>
```

Breakout: More Artist Details

- Make the artist's name a link to their spotify page (`artist.external_urls.spotify`)
- Loading feedback:
  - import the <paper-spinner> element
  - bind its "active" property to the <iron-ajax>'s "loading" property
- Homework (expert level):
  - Read into the "Computed Properties" chapter
  - Use a computed property to display the artist's picture
  - You need to create a custom element for this.

Round-Up Quiz

1. True or false:
   a) A custom element's "declared property" will become its attribute, e.g. `<my-element propertyxyz= "...">`
   b) "mySuperProperty" is a valid property name
   c) Databinding automatically works with sub-properties.
   d) Element names need to contain a dash.
2. Spot the 3 errors:

```
<iron-ajax
   url="artists.json"
   auto
   lastResponse="[[artists]]"></iron-ajax>
<template
   is="dom-bind"
   items="{{artists}}">

      <div>{{item}}</div>
</template>
```

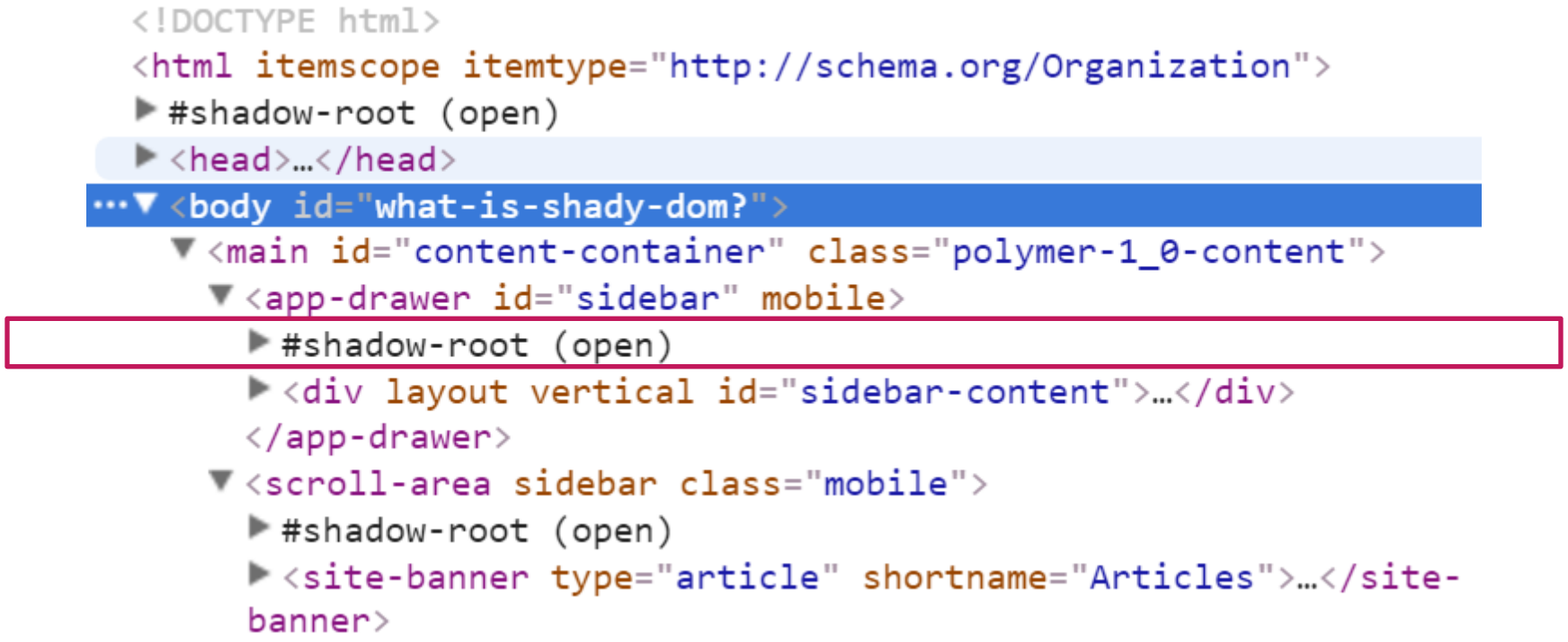
Thanks!

What are your questions?

Local DOM → Shadow DOM / Shady DOM

- Once the element is used, Local DOM inside the <template> tags becomes part of the host's "Shadow DOM"
- Current definition of Shadow DOM (W3C):
  "a method of combining multiple DOM trees into one hierarchy and how these trees interact with each other within a document, thus enabling better composition of the DOM"
- Implications:
  - Element IDs inside shadow DOM do not collide with other DOM(s)
  - More convenient CSS scopes
  - Clean JavaScript namespace
- Polymer (also) uses the lightweight version "Shady DOM"

Shadow DOM in the Developer Console



Screenshot of element inspector on current version of
https://www.polymer-project.org/1.0/articles/shadydom.html

Links

- What the heck is Shadow DOM? http://glazkov.com/2011/01/14/what-the-heck-is-shadow-dom/
- Introduction to Shadow DOM http://webcomponents.org/articles/introduction-to-shadow-dom/
- Styling Polymer https://www.polymer-project.org/1.0/docs/devguide/styling.html

Things to look into if you are interested

- Referential transparency