Web Components Introduction

A quick guide on how to use Web Components

Marcus Fihlon

May 28, 2016

Scrum Master | Software Engineer | Lecturer | Speaker

You don't need to take pictures of the slides!

Michael Sohn / AP

ABOUT ME

- **Scrum Master**
  CSS Insurance
- **Software Engineer**
  CSS Insurance / Open Source Software
- **Lecturer**
  TEKO Swiss Technical College
- **Speaker**
  Conferences / User Groups / Meetups

www.fihlon.ch | github.com/McPringle | hackergarten.net

AGENDA

Intro

Specifications

Goodies

Status

Live Coding

Wrap-up

INTRO

“Web Components are a set of standards currently being produced by Google engineers as a W3C specification that allow for the creation of reusable widgets or components in web documents and web applications. The intention behind them is to bring component-based software engineering to the World Wide Web. The components model allows for encapsulation and interoperability of individual HTML elements.”

Wikipedia

- New W3C Standard
- Allows reuse of components
- The standard is divided into four specifications:
  - Templates
  - Shadow DOM
  - Custom Elements
  - Imports
- A Web Component uses well-known technologies:
  - HTML
  - CSS
  - JavaScript
- No need of a framework or library
  - Except an optional polyfill to support older browsers

SPECIFICATIONS

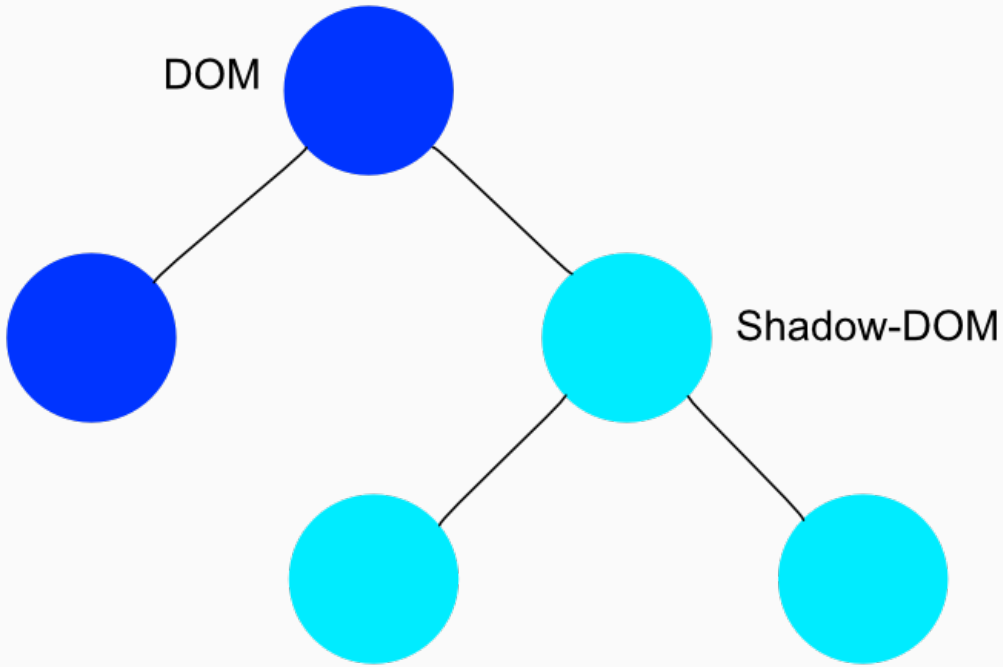
TEMPLATES

- Defines HTML parts to be reused any number of times
- Define reusable parts directly inside of HTML documents
- Is defined by the new `<template>` tag
- Can be added to the DOM using JavaScript
- Unlimited number of templates possible

```
1 <template id="my-template">
2     <div>
3         <img src="ok.png" width="20" height="20" />
4     </div>
5 </template>
```

SHADOW DOM

- Create an independent sub-DOM
- Not accessable from “outside” of the sub-DOM
- Avoids DOM collisions between components
- No side-effects of CSS or JavaScript between components
- Can be added to the DOM using JavaScript
- Unlimited number of Shadow DOMs possible



CUSTOM ELEMENTS

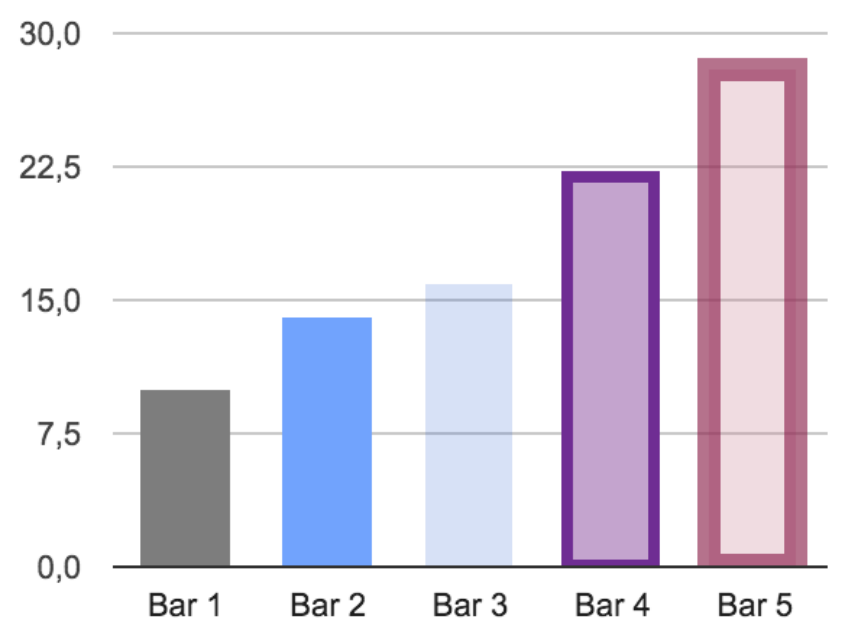
- Connect template and shadow DOM
- Define reusable components
- Create own tags to produce readable HTML
  → own tags need to include a hyphen
- Apply styles inside of the custom element
- Use JavaScript for interaction
- Throws lifecycle events:
  → created, ready, attached, detached, attributeChanged

```
1 <google-hangout-button />
```

IMPORTS

- Outsourcing of HTML parts
- Create own HTML files for components (higher reusability)
- Add components to HTML documents using imports

```
1 <link rel="import" href="google-chart.html">
2 <google-chart type="column" data="chart.json" />
```



GOODIES

CSS VARIABLES

```
1  :root {
2      --main-text-color: grey;
3  }
4
5  p {
6      color: var(--main-text-color, black);
7  }
```

CSS MIXINS

```
:root {
    --form-styles: {
        border: 1px dotted grey;
        font-size: 0.8em;
        margin: 1.2em;
    }
}

form {
    @apply(--form-styles);
}
```

STATUS

STATUS

	Chrome	Opera	Firfox	Safari	IE/Edge
Templates	✔	✔	✔	✔	✔
Shadow DOM	✔	✔	▲	▲	▲
Custom Elements	✔	✔	▲	▲	▲
Imports	✔	✔	▲	✘	✘

Polyfills

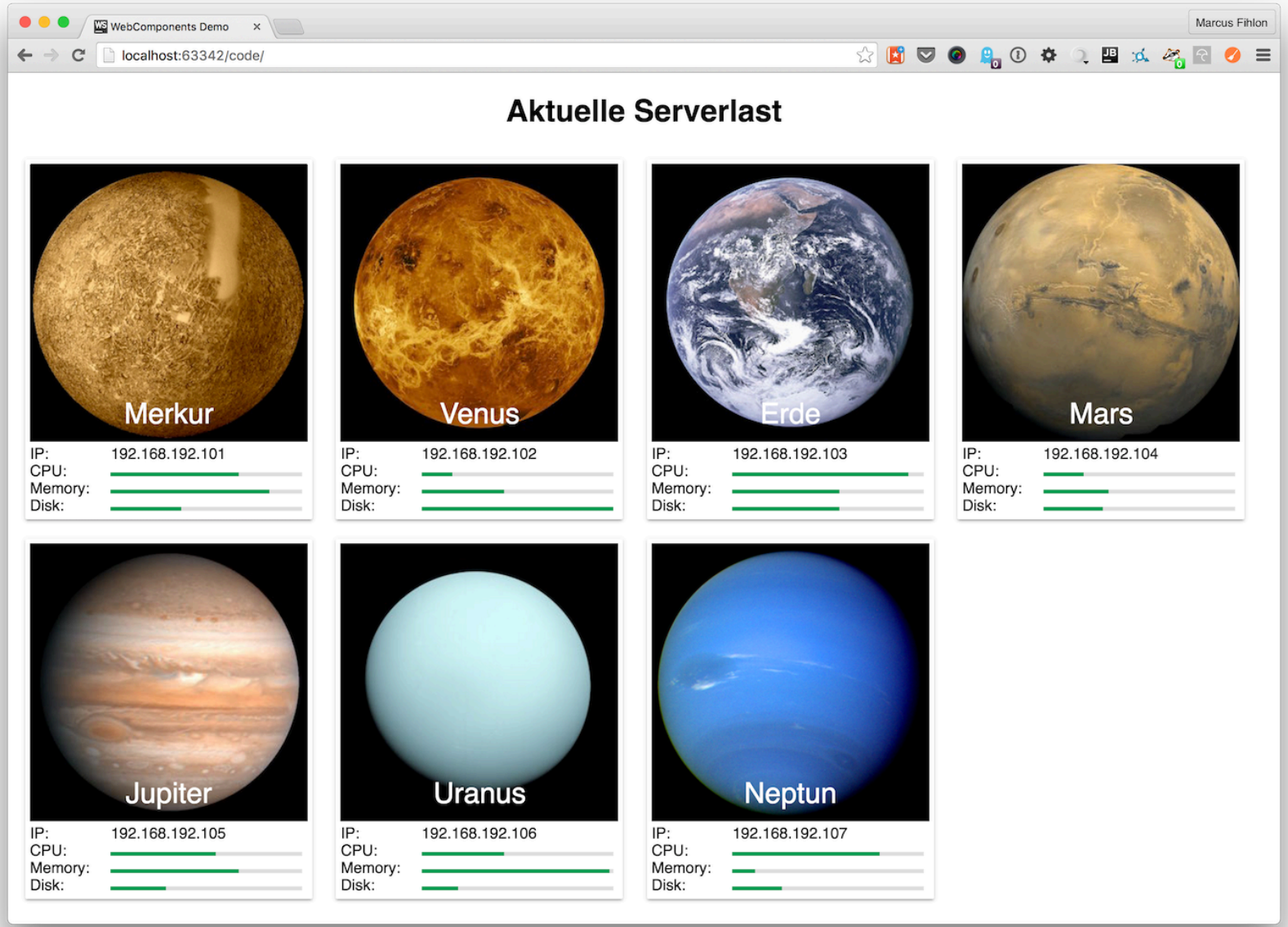
```
1 bower install webcomponentsjs
2 npm install webcomponents.js
```

Libraries

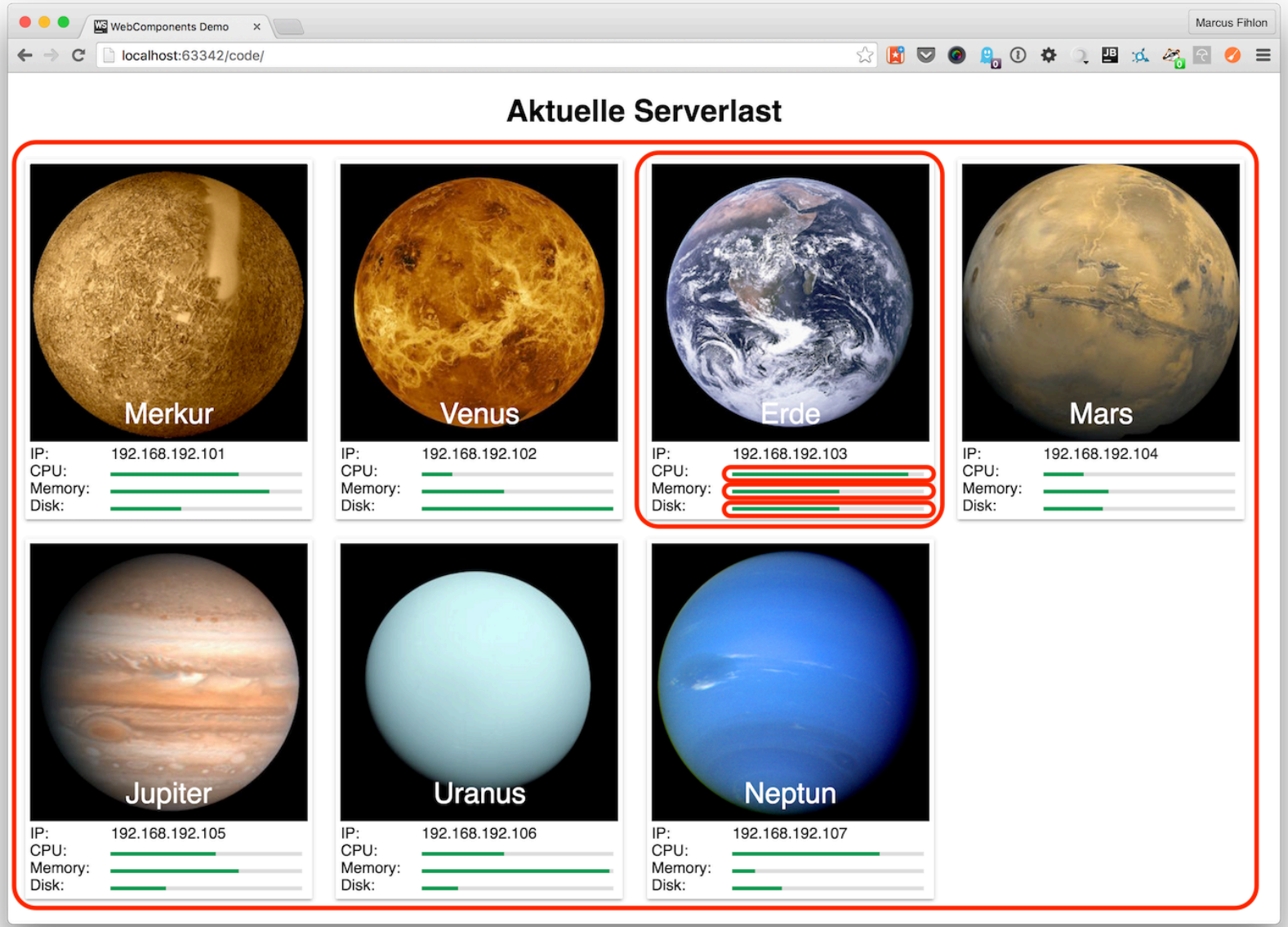
- Polymer
- X-Tag
- Bosonic

LIVE CODING

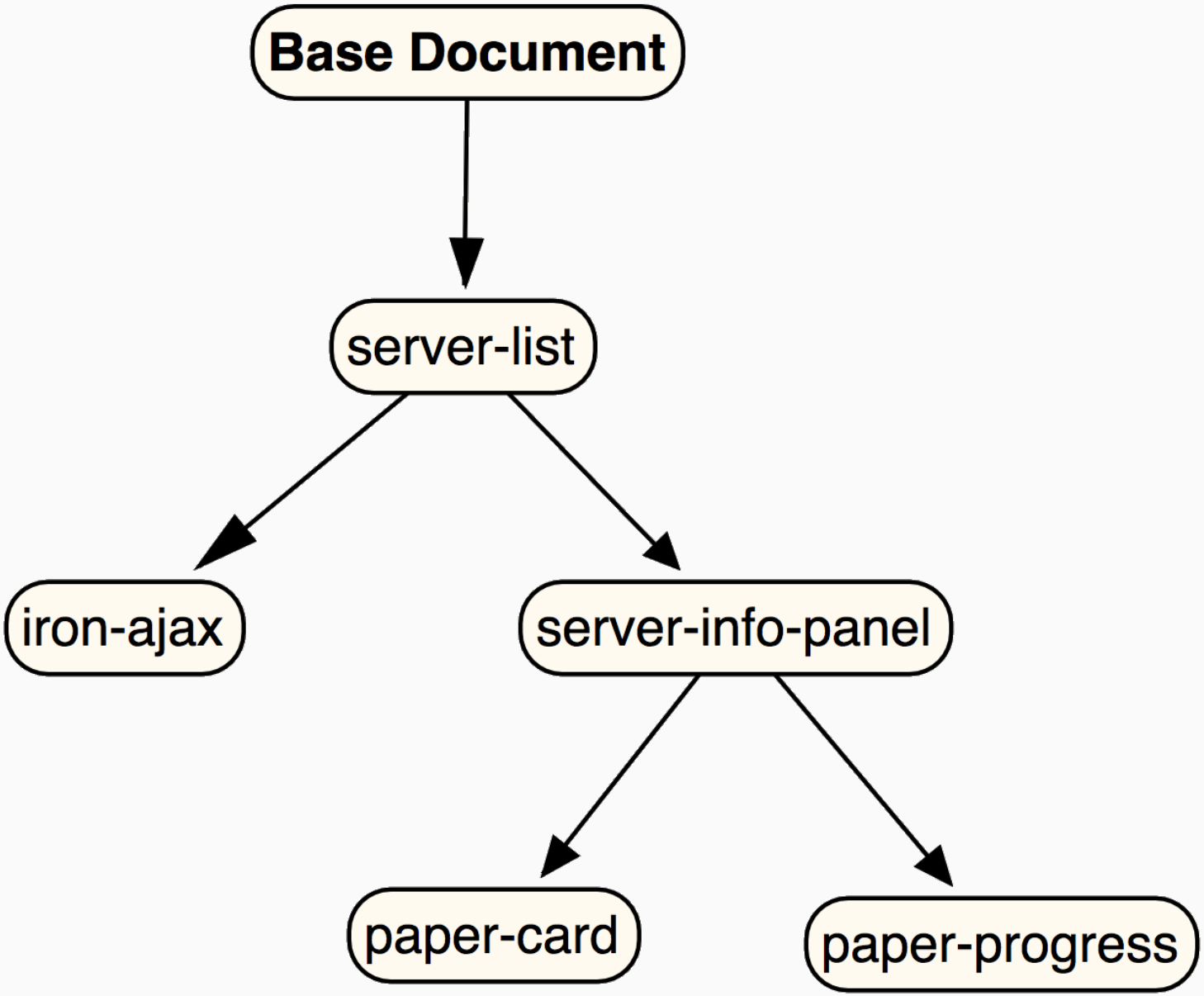
SCREENSHOT OF DEMO APPLICATION



COMPONENTS OF DEMO APPLICATION



Component structure of demo application



WRAP-UP

Web Components...

- are **declarative** and **reuseable**
- are **combinable** and **extensible**
- are **interoperational** – DOM = common demoninator
- allow **encapsulation** – scoping
- increase **productivity** and **accessibility**
- are **standard**
- support **thinking in components**

Thank You! Questions?

http://bit.ly/html-wc