Web Components Introduction

A quick guide on how to create and use Web Components

Marcus Fihlon

June 23, 2016

Scrum Master | Software Engineer | Lecturer | Speaker

DISCLAIMER

The following presentation has been approved for open audiences only. Hypersensitivity to occasional profanity requires covering ears.

All logos, photos etc. used in this presentation are the property of their respective copyright owners and are used here for educational purposes only. Any and all marks used throughout this presentation are trademarks of their respective owners.

The presenter is not acting on behalf of CSS Insurance, either as an official agent or representative. The views expressed are those solely of the presenter.

Marcus Fihlon and CSS Insurance disclaims all responsibility for any loss or damage which any person may suffer from reliance on this information or any opinion, conclusion or recommendation in this presentation whether the loss or damage is caused by any fault or negligence on the part of presenter or otherwise.

NOTES AND PHOTOS

You can take notes and photos if you want.
Or you can focus on the presentation and live coding.

All my slides and source files are available online.
A link and a QR code are on the last slide.

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

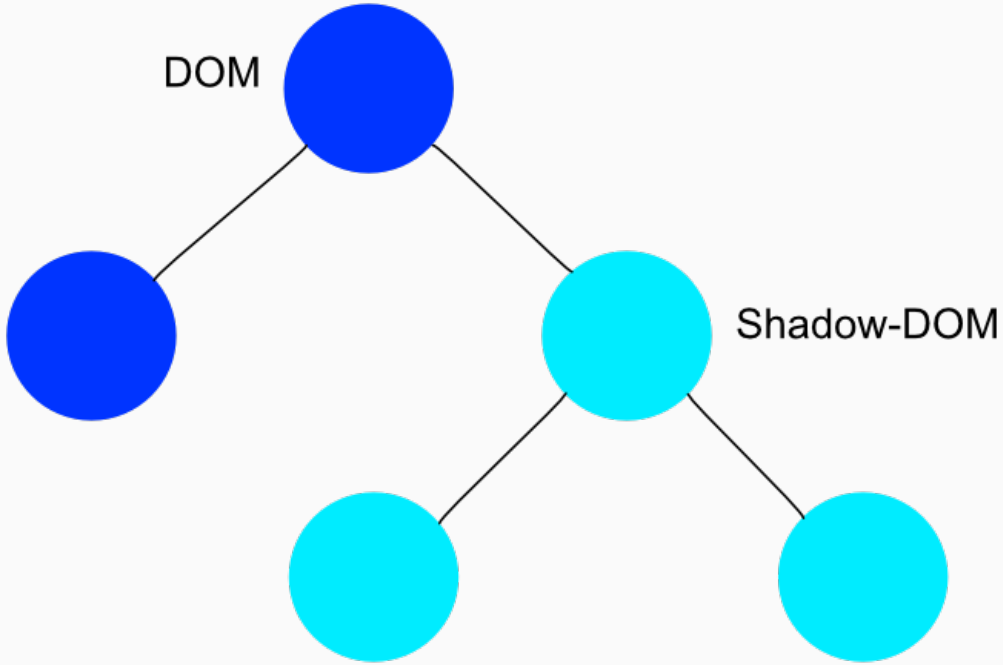
TEMPLATES

- 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

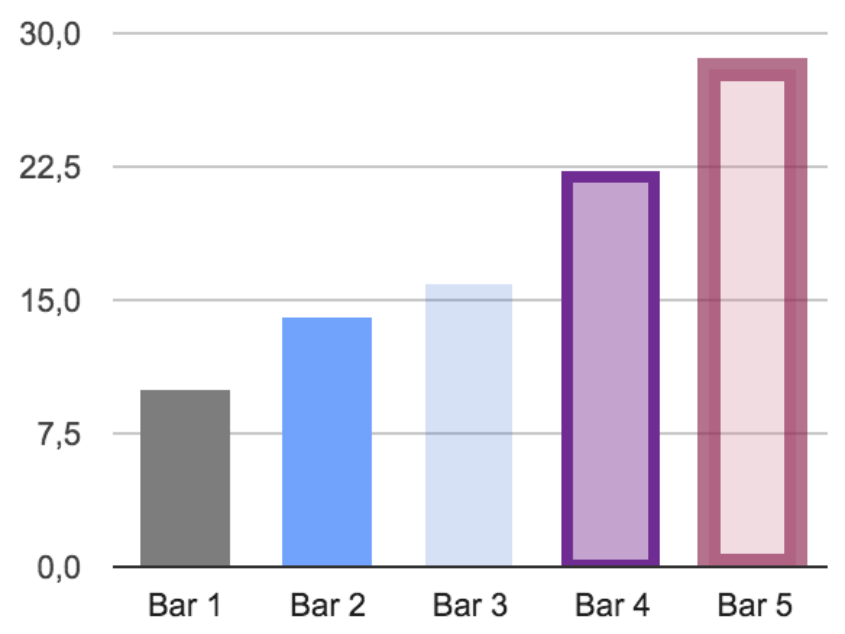
- Connect template and shadow DOM
- Define reusable components
- Create own tags to produce readable HTML
  → own tags need to include a hyphen
- Apply styles inside of the custom element
- Use JavaScript for interaction
- Throws lifecycle events:
  → created, ready, attached, detached, attributeChanged

```
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

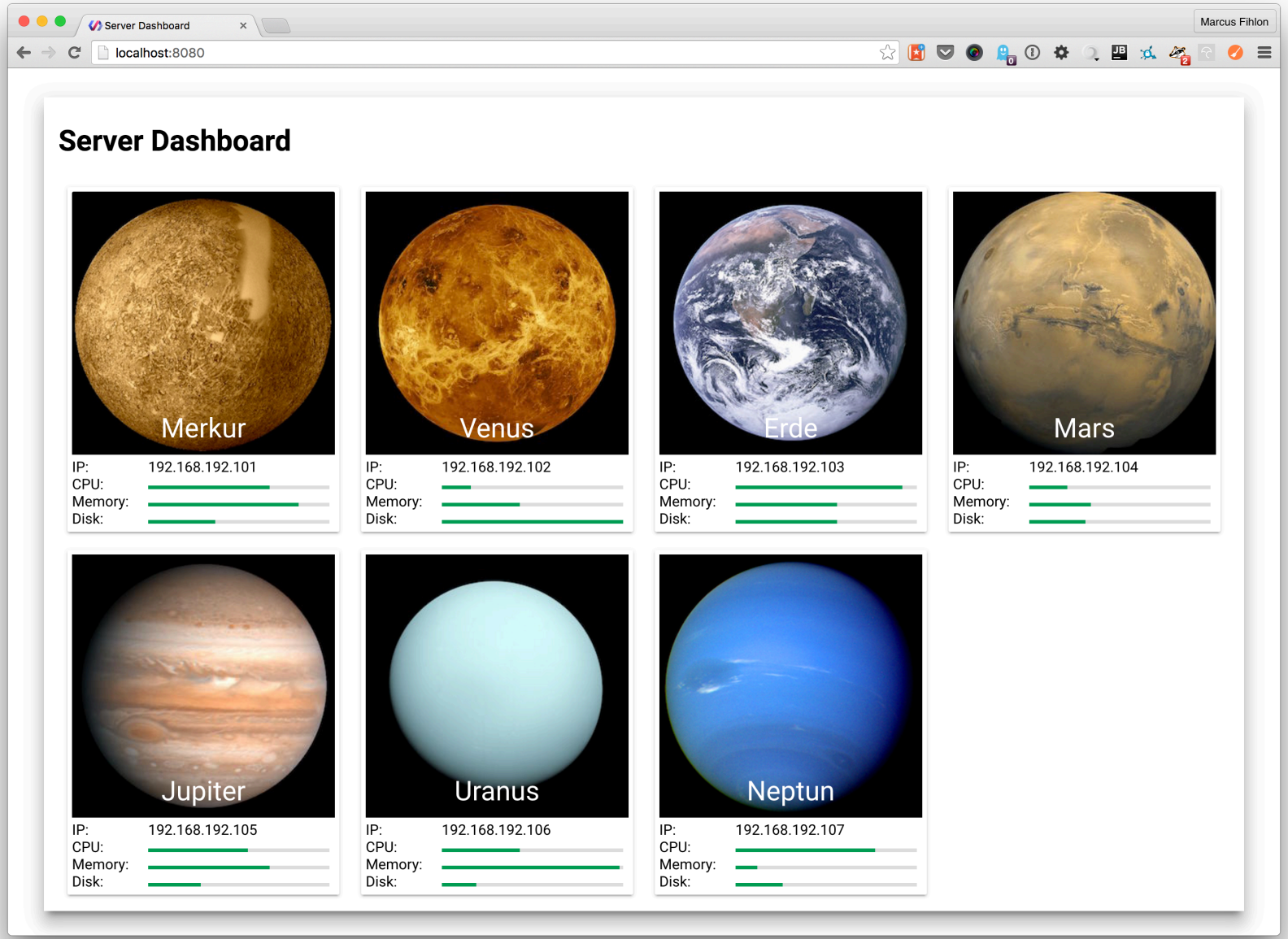
```
1 bower install webcomponentsjs
2 npm install webcomponents.js
```

Libraries

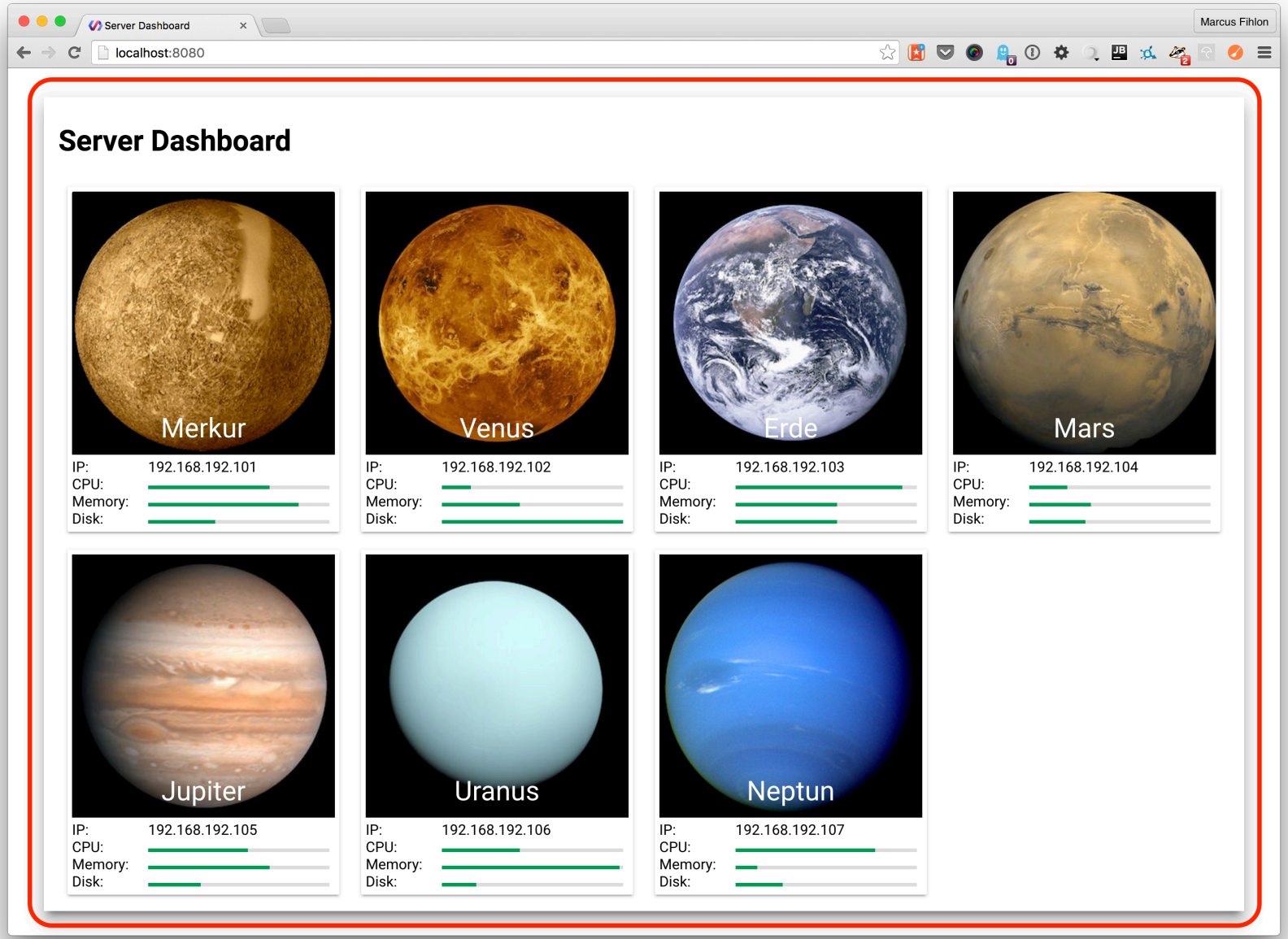
- Polymer
- X-Tag
- Bosonic

LIVE CODING

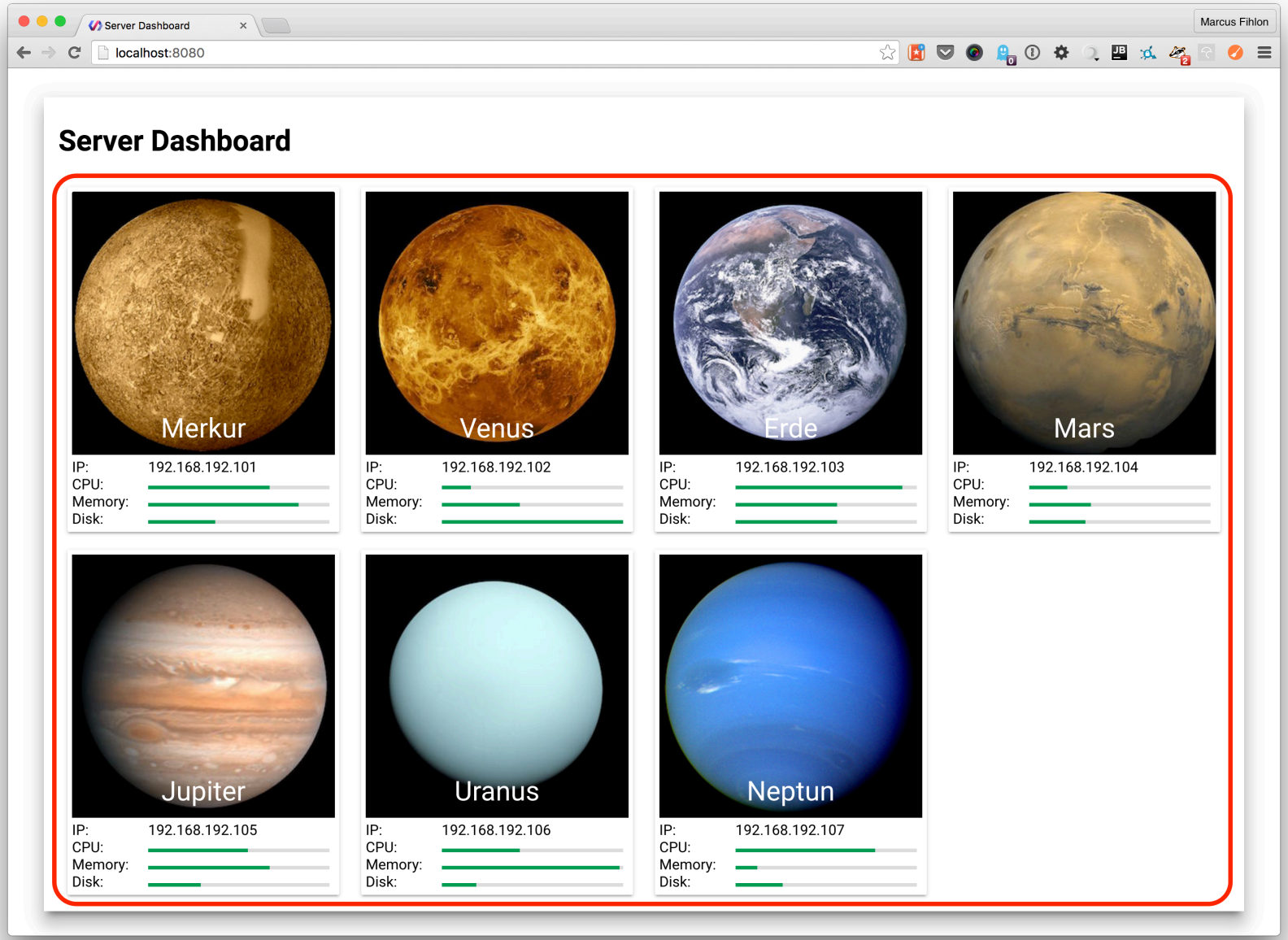
SCREENSHOT OF DEMO APPLICATION



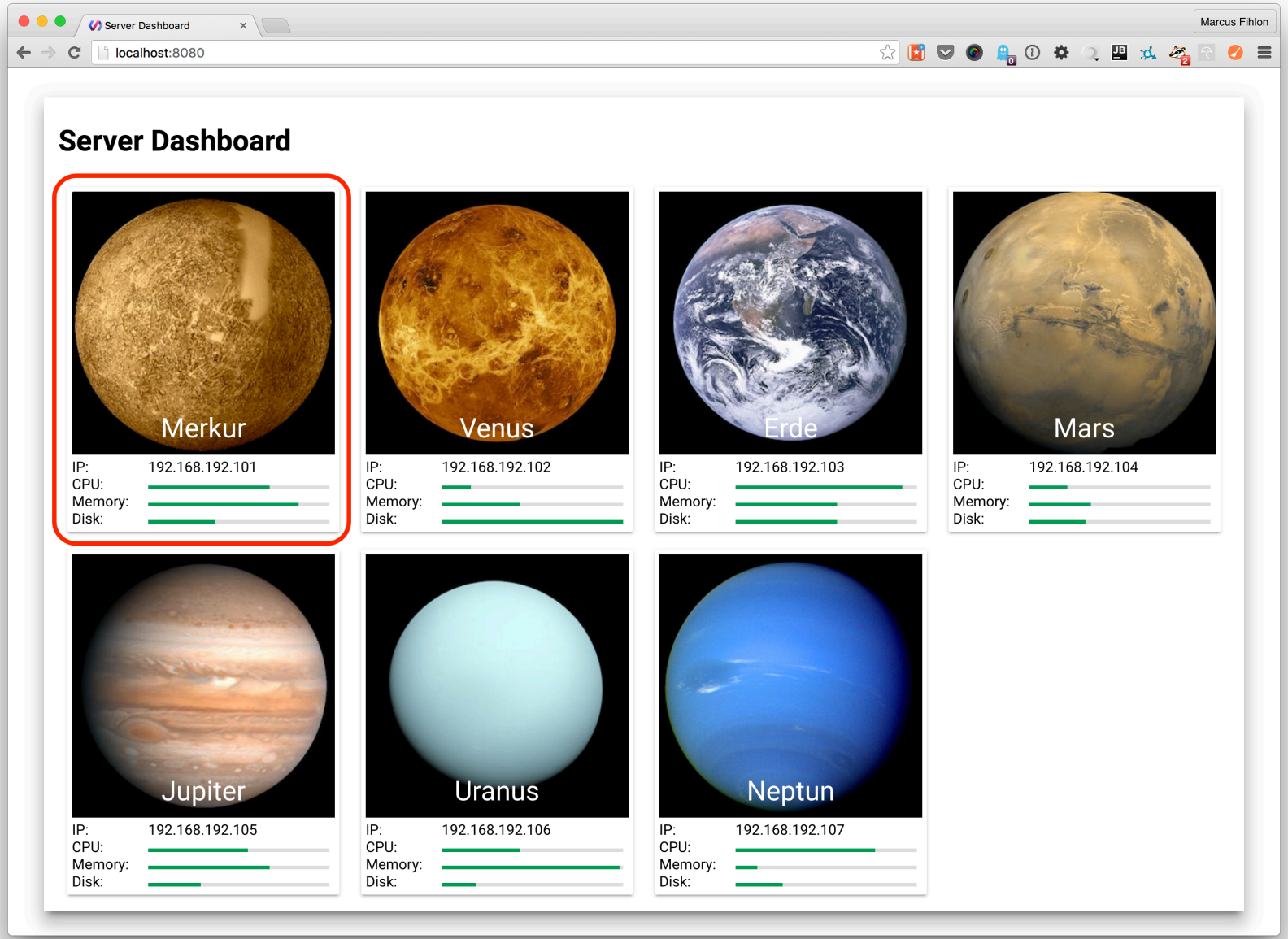
COMPONENTS OF DEMO APPLICATION



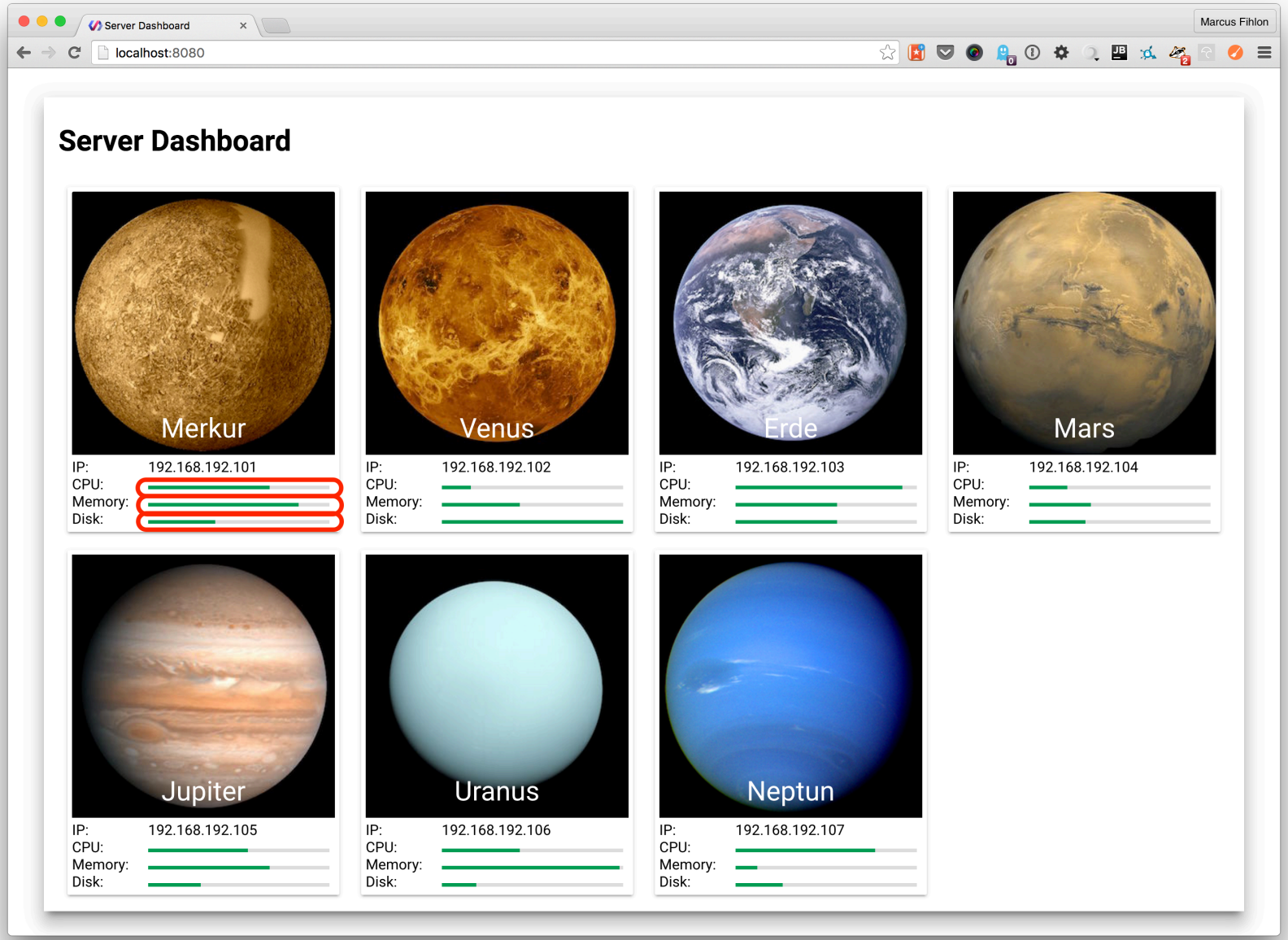
COMPONENTS OF DEMO APPLICATION



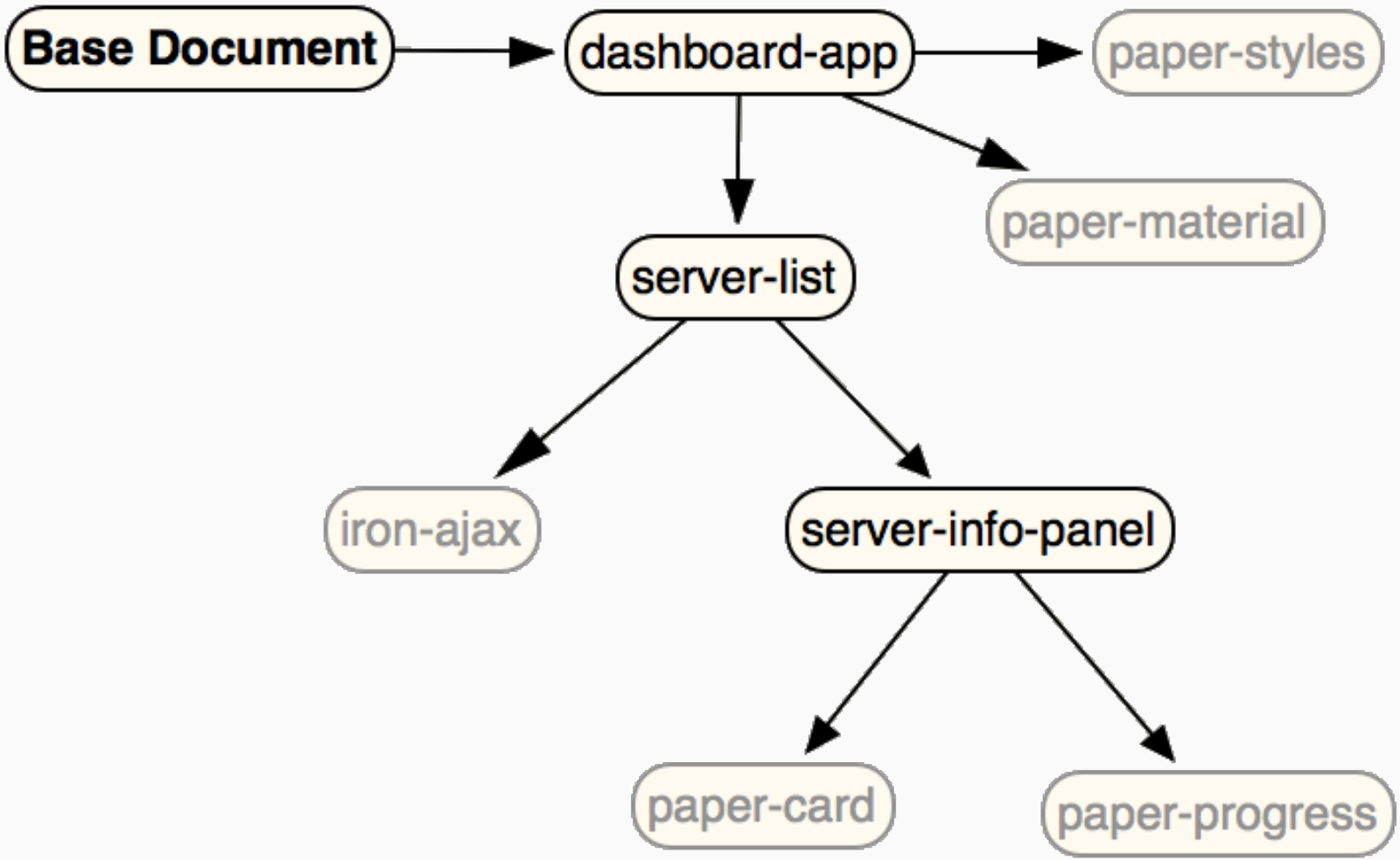
COMPONENTS OF DEMO APPLICATION



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**

COMMANDS

- Install the Polymer command line client

```
npm install -g polymer-cli
```

- Initialize a Polymer project

```
polymer init
```

- Serve a Polymer project

```
polymer serve
```

- Build a Polymer project

```
polymer build
```

- Test a Polymer project

```
polymer test
```

LINKS

- W3C Web Components Specification
  `https://w3.org/standards/techs/components`
- W3C Introduction to Web Components
  `http://w3.org/TR/components-intro/`
- Informations about Web Components
  `http://webcomponents.org`
- Directory of custom elements
  `https://customelements.io`
- Polymer Project
  `https://www.polymer-project.org`

Thank You! Questions?

http://bit.ly/html-wc