

# HTML5 Web Components: следующий шаг к модульности вашего проекта

Андрей Рахманов, Enaza  
DevConf 2015, 19 июня 2015 г.



<http://www.devconf.ru>

### Свойства

Название: castle\_europe

Размер: 841 x 706 px

Поворот: 0°

Центр вращения: 420 x 353 px

Видимость: 100 %

Блокировка:

Отображать на устройстве:

Позиционирование

Отступы

Сверху: 64 px

Справа: 0 px

Снизу: 0 px

Слева: -101 px

### Свойства изображения

Масштаб: 153 x 153 %

Выбрать

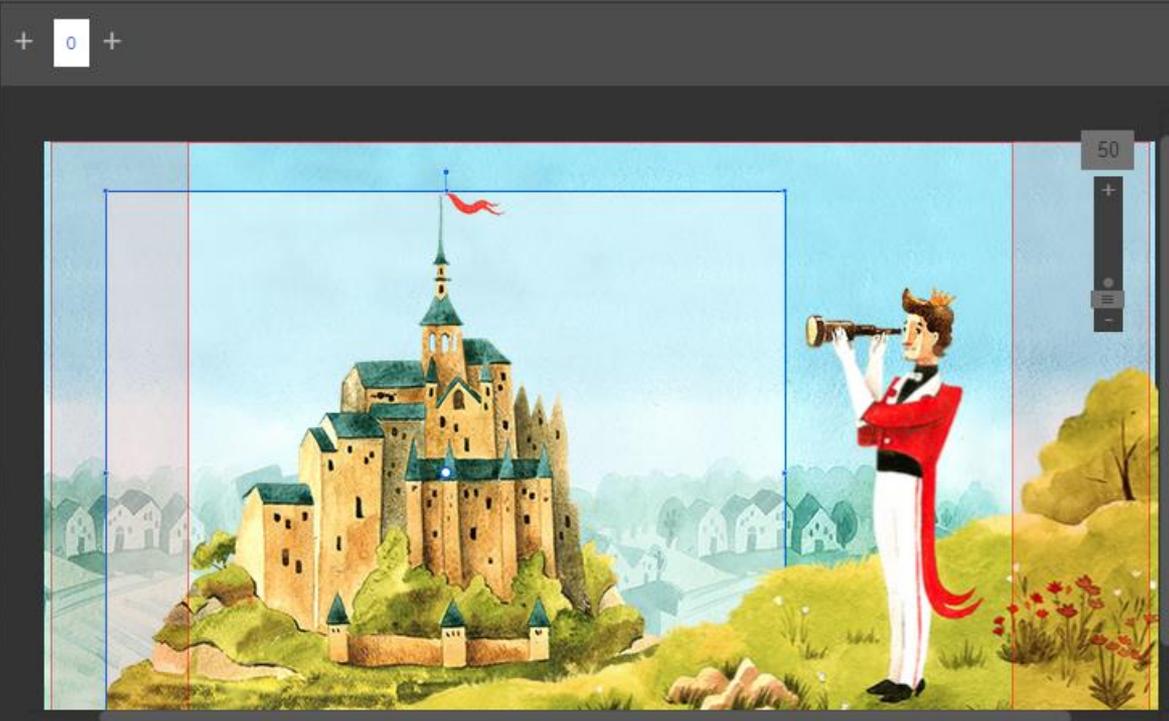
Привязать действие

Состояния

Невидимый

Видимый

Создать состояние



Сохранено

Действия | + | События

Состояние	Действие	Таймер
Япония в Европу	+ Создать цепочку	0   1   2   3
Европа в Россию	castle_japan	[Progress bar]
Россия в Японию	castle_europe	[Progress bar]

Панель действий

### Библиотека файлов

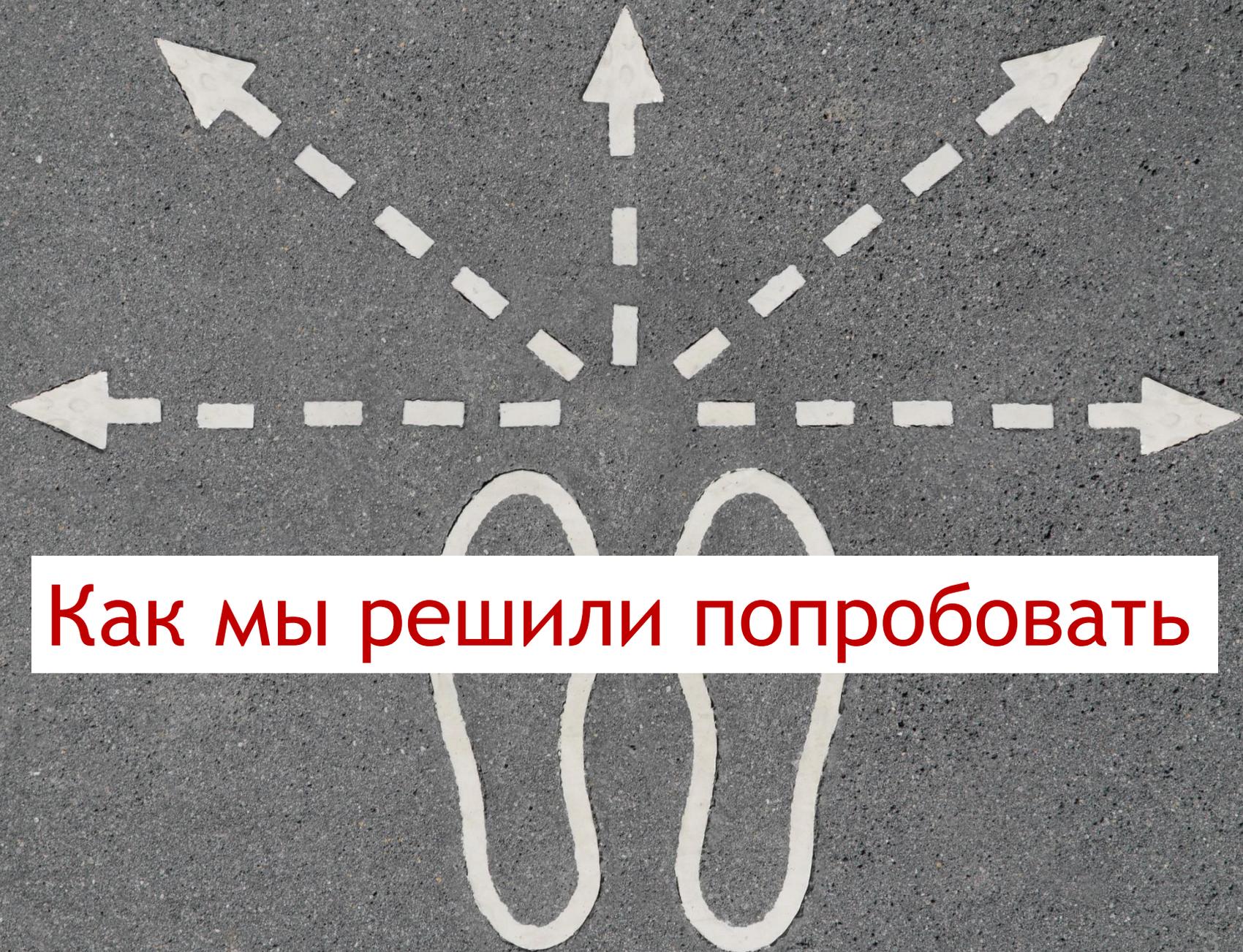
castle\_russia.png  
550 x 462 px  
8 bit

Имя

- sound\_europa.mp3
- prince.png
- sound\_russia.mp3
- castle\_europe.png
- sound\_japan.mp3
- castle\_russia.png**
- castle\_japan.png
- back.jpg

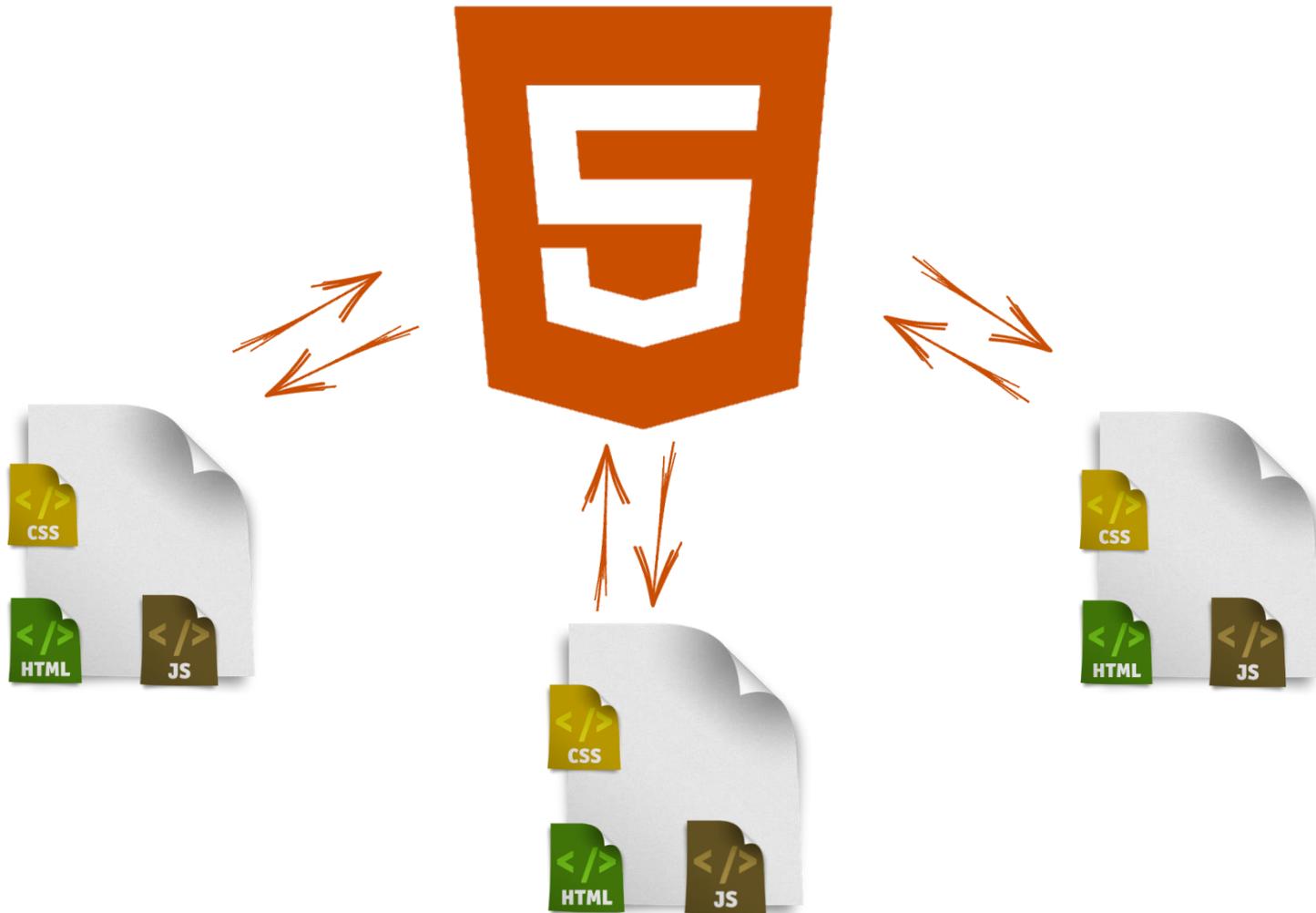
Главы, страницы, компоненты

- Мастер-страницы
- Верхняя
- Нижняя
- Глава 1
  - 0- Страница 1
    - prince
    - castle\_japan
    - castle\_europe**
    - castle\_russia
    - back



**Как мы решили попробовать**

## Single-Page App



```

<style>
  #map {
    height: 400px;
  }
</style>

<div id="map"></div>

<script src="http://maps.googleapis.com/maps/api/js?callback=mapReady"></script>
<script>
  var marker = null;

  function getCurrentLocation(callback) {
    navigator.geolocation.watchPosition(callback);
  }

  function addMarker(opts, info) {
    var marker = new google.maps.Marker(opts);

    var infoWindow = new google.maps.InfoWindow({content: info});

    google.maps.event.addListener(marker, 'click', function() {
      infoWindow.open(opts.map, marker);
    });

    return marker;
  }

  function mapReady() {
    var container = document.querySelector('#map');
    //...
  }
  //...

```

```
<style>
  #map {
    height: 400px;
  }
</style>

<div id="map"></div>

<script src="http://maps.googleapis.com/maps/api/js?callback=mapReady"></script>
<script>
  var marker = null;

  function getCurrentLocation(callback) {

    var marker = new google.maps.Marker(opts);

    var infoWindow = new google.maps.InfoWindow({content: info});

    google.maps.event.addListener(marker, 'click', function() {
      infoWindow.open(opts.map, marker);
    });

    return marker;
  }

  function mapReady() {
    var container = document.querySelector('#map');
    //...
  }
  //...
```

**<google-map></google-map>**



# BACKBONE.JS

# Поиск подходящего решения

- Плохо сочетать Backbone.js с другими фреймворками
- Надо использовать native-реализацию или библиотеку



`<dom-module>`

`<style>`

`</style>`



`<template>`

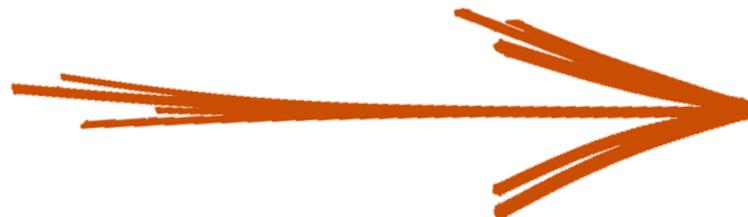
`</template>`

`</dom-module>`



`<script>`

`</script>`



# Составные части Web Components

- Templates
- HTML Imports
- Custom Elements
- Shadow DOM

## Templates



## HTML Imports



## Custom Elements



## Shadow DOM



Native: Пока весьма печально...

## Templates



## HTML Imports



## Custom Elements



## Shadow DOM



Polyfills: Уже лучше!

# Какие варианты есть на текущий момент?



Native



Bosonic



X-Tag (Mozilla)

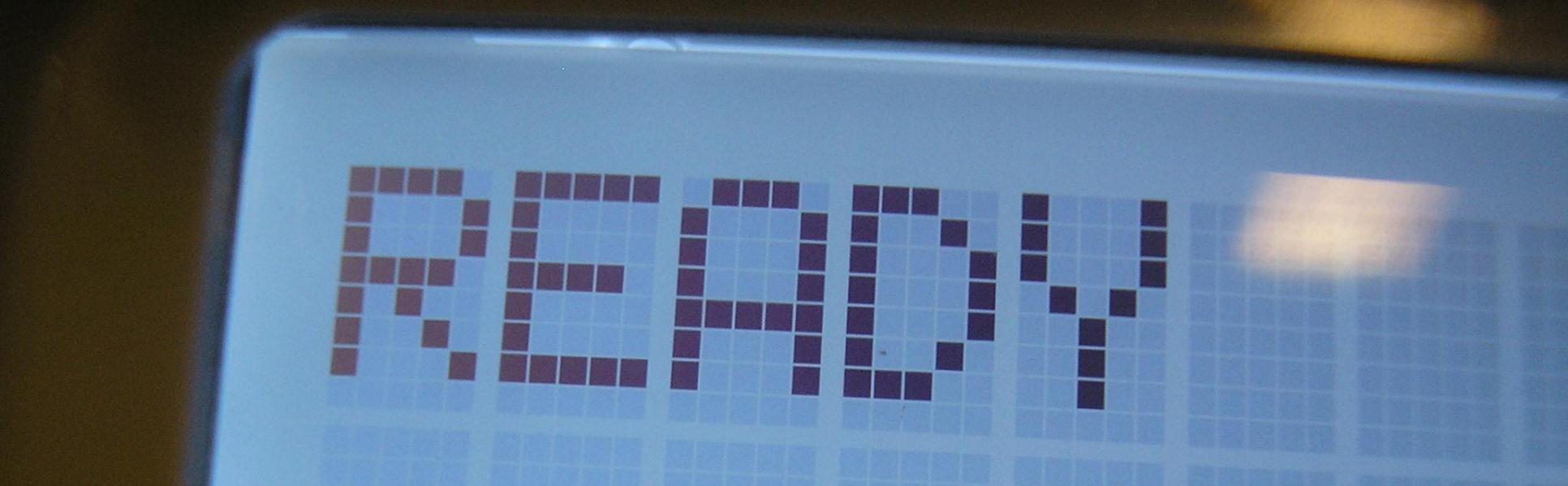


Polymer (Google)

# Мы выбрали Polymer

- Декларативное описание элементов
- Удобный «синтаксический сахар» над нативной реализацией



A close-up photograph of a digital display, likely a scoreboard or a game screen, showing the word "READY" in a pixelated, blocky font. The letters are composed of dark blue or black pixels on a lighter blue background. The display is slightly out of focus, and there are some reflections and light artifacts on the surface.

READY

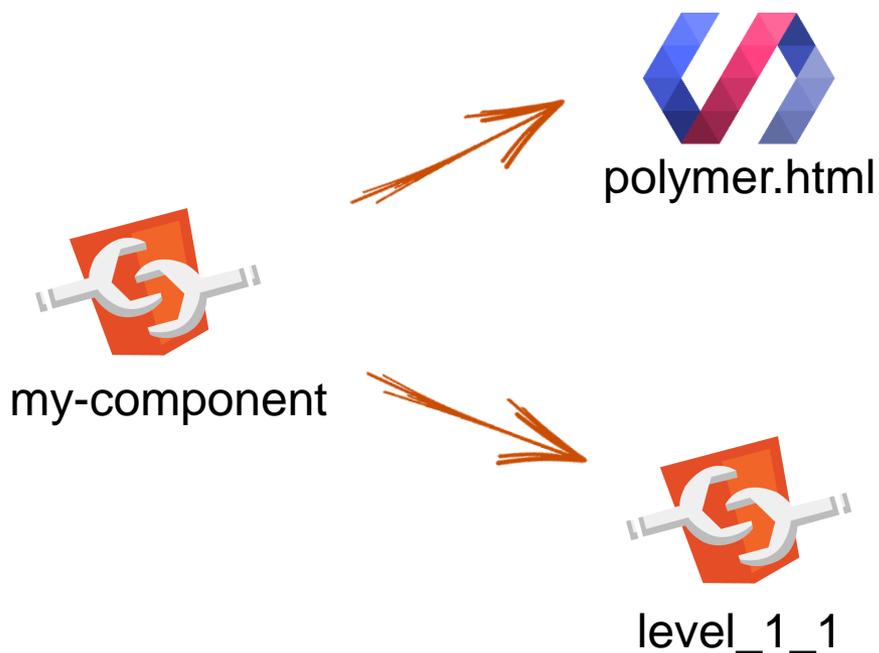
На пороге интеграции  
в существующий проект

## Подключаем один компонент

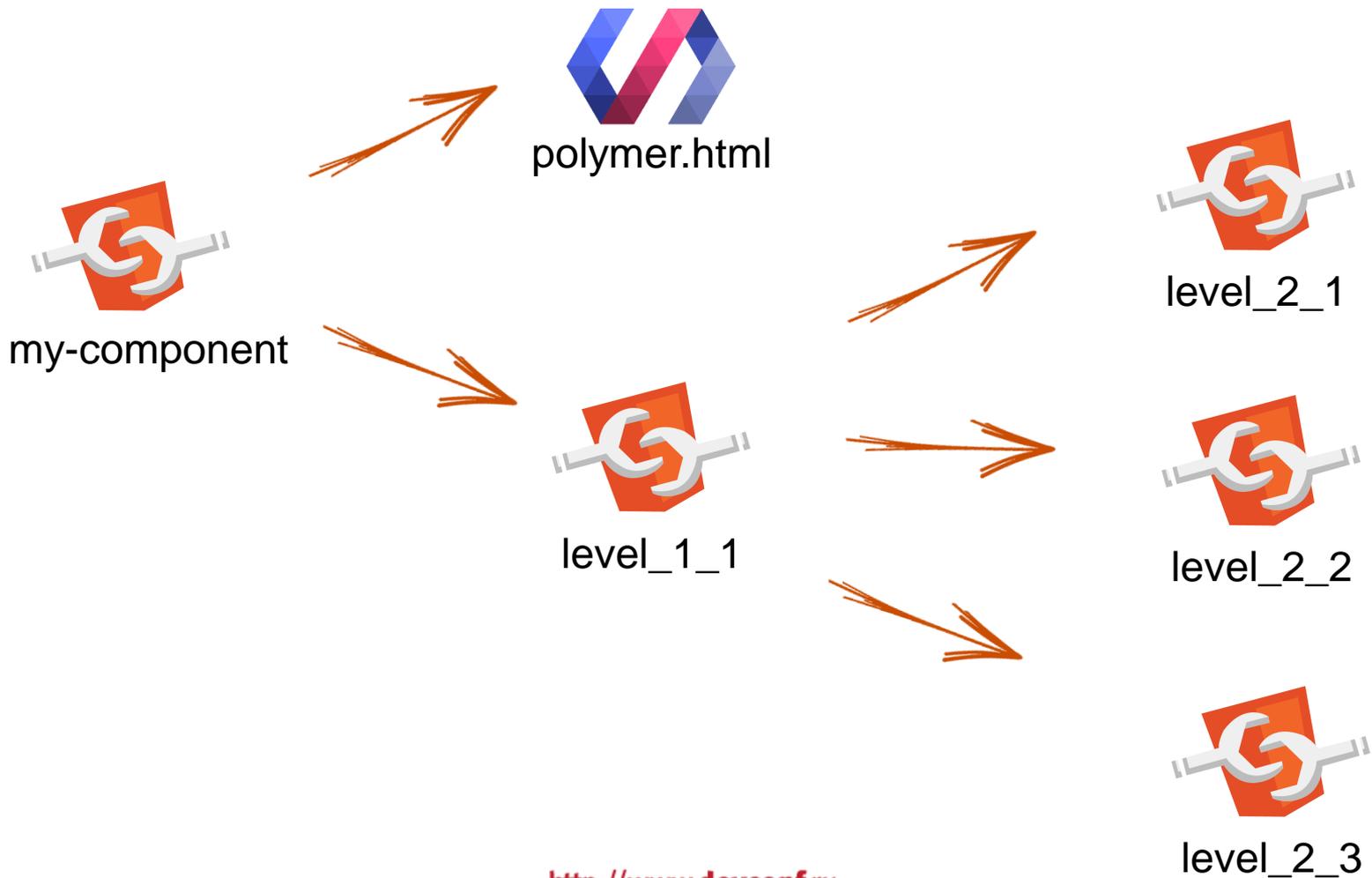


my-component

# Подключаем один компонент



## Подключаем один компонент



```
<link rel="import" href="simple_component.html">
```

а за НИМ ПОЙДУТ...

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

```
<link rel="import" href="level_1_1.html">
```

```
<link rel="import" href="level_2_1.html">
```

```
<link rel="import" href="level_2_2.html">
```

```
<link rel="import" href="level_2_3.html">
```

**1 import == 6 запросов**

# Out-of-box: Polymer Vulcanize

Реализация:

```
var vulcan = require('vulcanize');

vulcan.setOptions({
  /* конфигурация сборки */
});

vulcan.process(target, function(err, inlinedHtml) {
  /* запись готового html в файл all_in_one.html */
});
```

Использование:

```
<link rel="import" href="all_in_one.html">
```

component\_2.html

component\_3.html

component\_1.html

styles.css

polymer.html

script.js



# Подключение: HTML Import сборки

```
<html>  
  <!-- содержимое index.html -->  
  
  <link rel="import" href="components.html">  
</html>
```



1 лишний запрос  
не так страшен



Сборка  
кэшируется

# Взаимодействие с используемым стеком: Jade, Stylus и Browserify

# Стек технологий



## Стек технологий



## Стек технологий



## Как делаем мы

**jade**  
Node Template Engine



*stylus*



**JS**



CommonJS  
 Browserify



# Чего ожидает от нас Polymer

```

<dom-module>
  <style>
    </style>
  <template>
    </template>
</dom-module>

```



Файл с компонентом

```

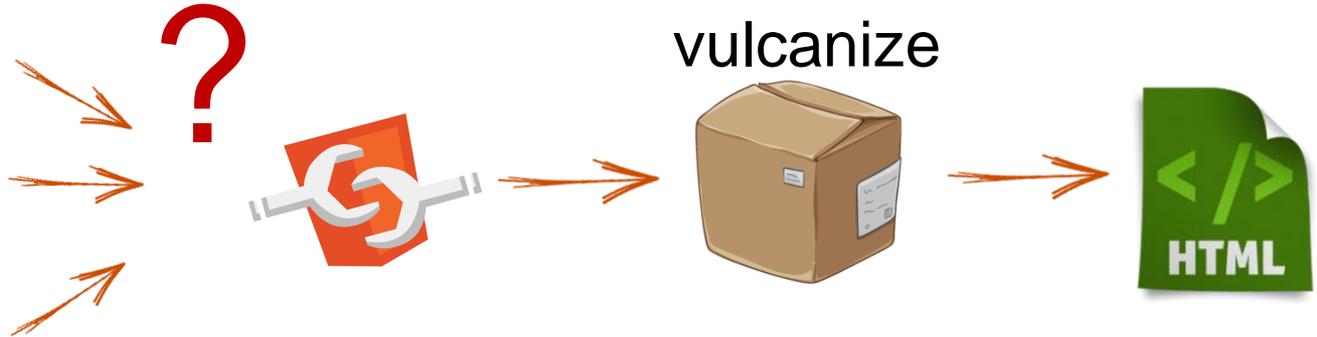
<script>
</script>

```



**iade**  
Node Template Engine

*stylus*



**JS**



Вот кто нам поможет!

# Gulp Task: Jade -> HTML

```
var searchTemplate = dirname + '/*.jade';

gulp.task(taskId, function () {
  return gulp.src(searchTemplate)
    .pipe(buffer())
    .pipe(jade({}))
    .pipe(gulp.dest(dirname));
});
```

# Gulp Task: Stylus -> CSS

```
var searchTemplate = dirname + '/*.styl';

gulp.task(taskId, function () {
  return gulp.src(searchTemplate)
    .pipe(buffer())
    .pipe(stylus({}))
    .pipe(gulpif(env == 'production', minifyCSS()))
    .pipe(gulp.dest(dirname));
});
```

# Gulp Task: склеим 3 файла в 1

```
var searchTemplates = [  
  dirname + '/prebuild-*.html',  
  dirname + '/prebuild-*.css',  
  dirname + '/prebuild-*.js'  
];  
  
gulp.task(taskId, function () {  
  return gulp.src(searchTemplates)  
    // сборка в нужном порядке  
    .pipe(order(['*.html', '*.css', '*.js']))  
    .pipe(buffer())  
    .pipe(concat(resultname))  
    .pipe(gulp.dest(dirname));  
});
```

**jade**  
Node Template Engine

*stylus*

JS



first.html



**jade**  
Node Template Engine

*stylus*

JS



second.html



**jade**  
Node Template Engine

*stylus*

JS



third.html



**jade**  
Node Template Engine

*stylus*

JS

first.html



register\_components.html

**jade**  
Node Template Engine

*stylus*

JS

second.html



**html imports:**

first.html  
second.html  
third.html

vulcanize



**jade**  
Node Template Engine

*stylus*

JS

third.html



```
<!-- подключение библиотеки Polymer -->
<link rel="import"
  href="./bower_components/polymer/polymer.html">

<!-- динамический реестр используемых компонентов -->
<link rel="import"
  href="./client/wc-up-input-autoresize/up-input-autoresize.html">

<link rel="import"
  href="./client/wc-up-textarea-count/up-textarea-count.html">

<link rel="import"
  href="./client/wc-up-select/up-select.html">
```

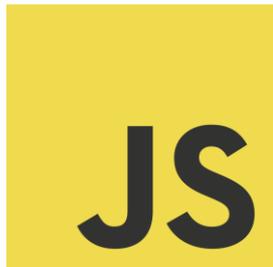
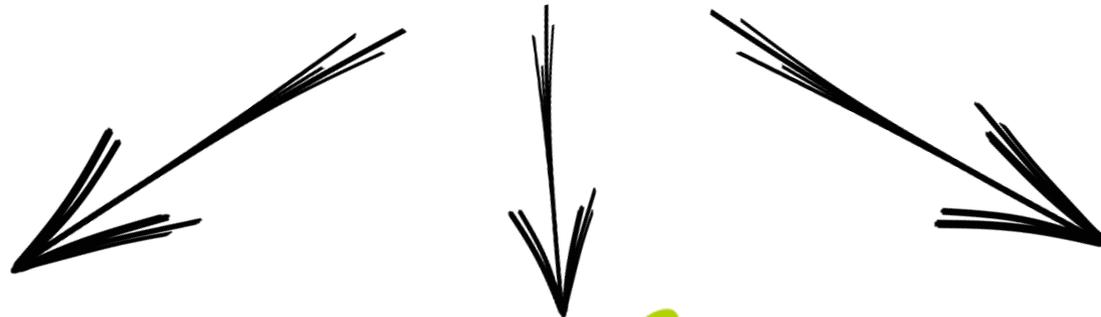
# Этот реестр формирует Gulp

## Генерация шаблонов



YEOMAN

```
npm install -g generator-poly-jadestyl  
yo poly-jadestyl add my-web-component
```



**jade**  
Node Template Engine

```
dom-module(id="my-web-component")  
  template  
    p Test content
```

**JS**

```
Polymer({  
  is: 'my-web-component',  
  
  ready: function () {  
  }  
})
```

*stylus*

```
:host  
  background red
```

## Резюме

- Web Components успешно интегрируются с имеющимися инструментами
- Gulp способен решить множество задач по сборке Web Components
- Создание нового компонента сводится к одной строке кода для генератора

# Где применить веб-компоненты?



# Декомпозиция Single-Page App

На низком уровне

Расширение стандартных элементов управления

На высоком уровне

Переиспользуемые логические блоки

Низкий уровень

# Расширение стандартных элементов управления

# Расширение стандартных элементов управления

- `<input is="up-autoresizable" />`  
- автоподстройка длины
- `<input is="up-colorpicker" />`  
- выбор цвета с альфа-каналом  
- указание цвета числом
- `<up-textarea-count></..>`  
- подсчет числа оставшихся символов

## <input is="up-autoresizable"/>

```
var $ = require('jquery');           | зависимости через  
      require('jquery.autosize.input'); | Browserify
```

```
Polymer({  
  is: 'up-autosizable',             | наследование  
  extends: 'input',                 | от input  
  
  // момент добавления в DOM  
  attached: function () {           | подключение  
    $this.autosizeInput();           | плагина  
  },  
  
  detached: function () {           | отключение  
    $this.autosizeInput('destroy');  | плагина  
  }  
})
```

```
<input is="up-colorpicker"/>
```

```
Polymer({  
  is: 'up-colorpicker',           | наследование  
  extends: 'input',              | от input  
  
  //... properties  
  
  ready: function () {  
    $(this).spectrum();  
  },  
  
  // примерить цвет  
  handleMove: function () {     | кастомное  
    $(this).trigger('preview', color); | событие  
  }  
})
```

```
<input is="up-colorpicker" alpha="1" />
```



## <up-textarea-count>

```

Polymer({
  is: 'up-textarea-count',

  get value() {
    return this.$.textarea.value;
  },

  set value(value) {
    this.$.textarea.value = value;
  },

  listeners: {
    'textarea.keyup': 'calcLength',
    'textarea.change': 'calcLength'
  },

  calcLength: function () {
    this.counter = this.value.length;
  }
  // ... прочая логика
})

```

| Public API  
 | для взаимодействия  
 |  
 |  
 |

```
<up-textarea-count maxlength="140" rows="10" cols="20">  
  2015  
</up-textarea-count>
```

Высокий уровень

# Переиспользуемые логические блоки

<внешний блок (атрибуты...)>

<внутренний блок 1 (атрибуты...)>

<поле 1>

<поле 2>

<внутренний блок 2 (атрибуты...)>

<поле 1>

<поле 2>

# <up-inspector>

```
<up-inspector label="Свойства"...
```

```
<up-inspector-list label="Позиционирование"
```

```
<!-- элементы управления -->
```

```
<!-- элементы управления -->
```

```
<up-inspector-list label="Отступы"
```

```
<!-- элементы управления -->
```

```
<!-- элементы управления -->
```

Свойства	
Название	Квадрат
Размер	45 x 52 px
Поворот	0°
Центр вращения	22 x 26 px
Видимость	100%
Блокировка	<input type="checkbox"/>
▼ Позиционирование	
<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
▼ Отступы	
Сверху	43 px
Справа	0 px
Снизу	0 px
Слева	41 px
▶ Растягивание	
Свойства квадрата	
Фон	<input type="color"/>
▼ Контур	
Цвет	<input type="color"/>
Толщина	1 px
Закругление	0 px

```
<up-inspector name="properties" label="Свойства">  
  <up-inspector-list name="position" label="Позиционирование">  
    <!-- элементы управления -->  
  </up-inspector-list>  
  
  <up-inspector-list name="offset" label="Отступы">  
    <!-- элементы управления -->  
  </up-inspector-list>  
</up-inspector>
```

## Резюме

- Web Components сильны на обоих уровнях
- И разметка, и логика превращаются в набор логических блоков
- Полученный результат удобно поддерживать

Так здорово ли работать  
с веб-компонентами  
в реальном проекте?





**Определенно, ДА!**

# Наиболее частые проблемы

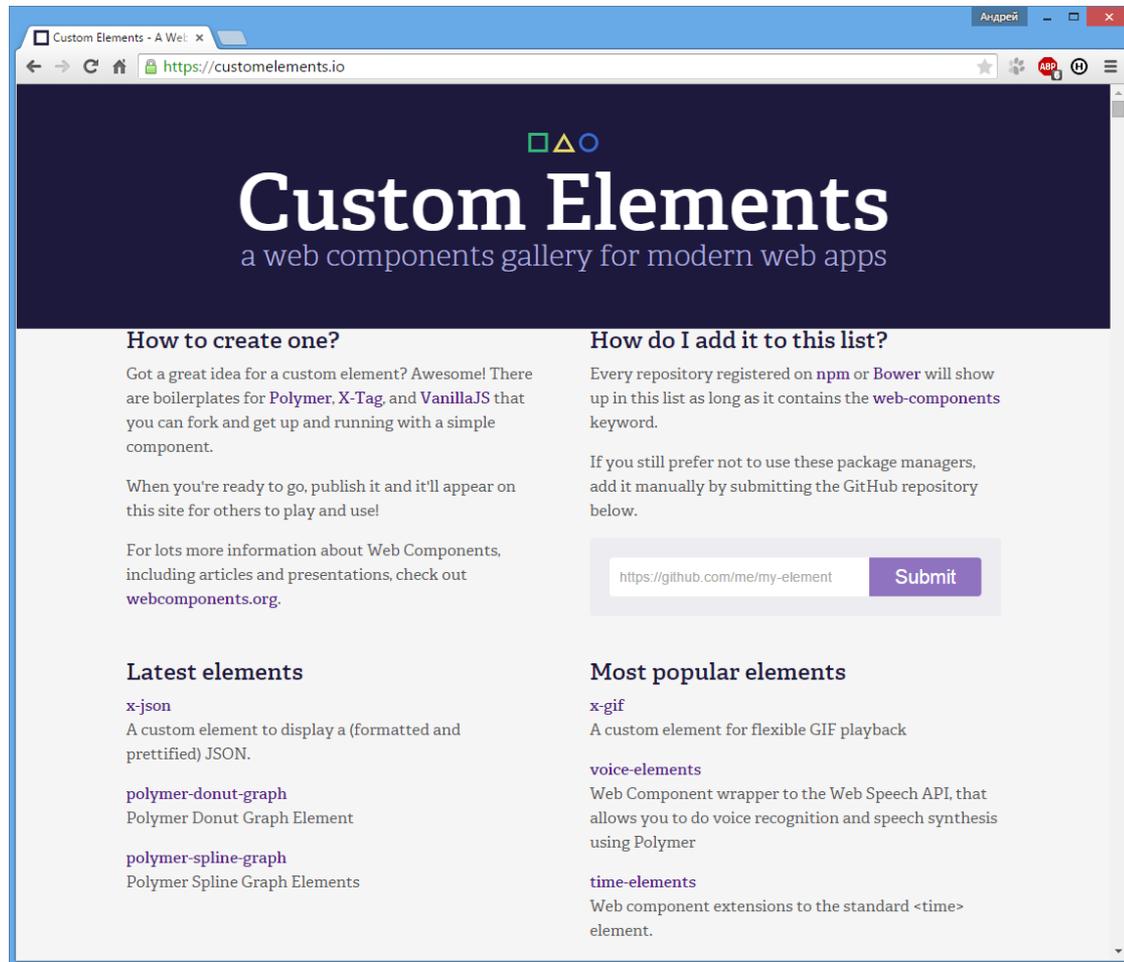
- Не все документировано
- Возможны изменения API
- Мало информации на stackoverflow по свежим версиям библиотек (...но это временно)

# Полезные ресурсы

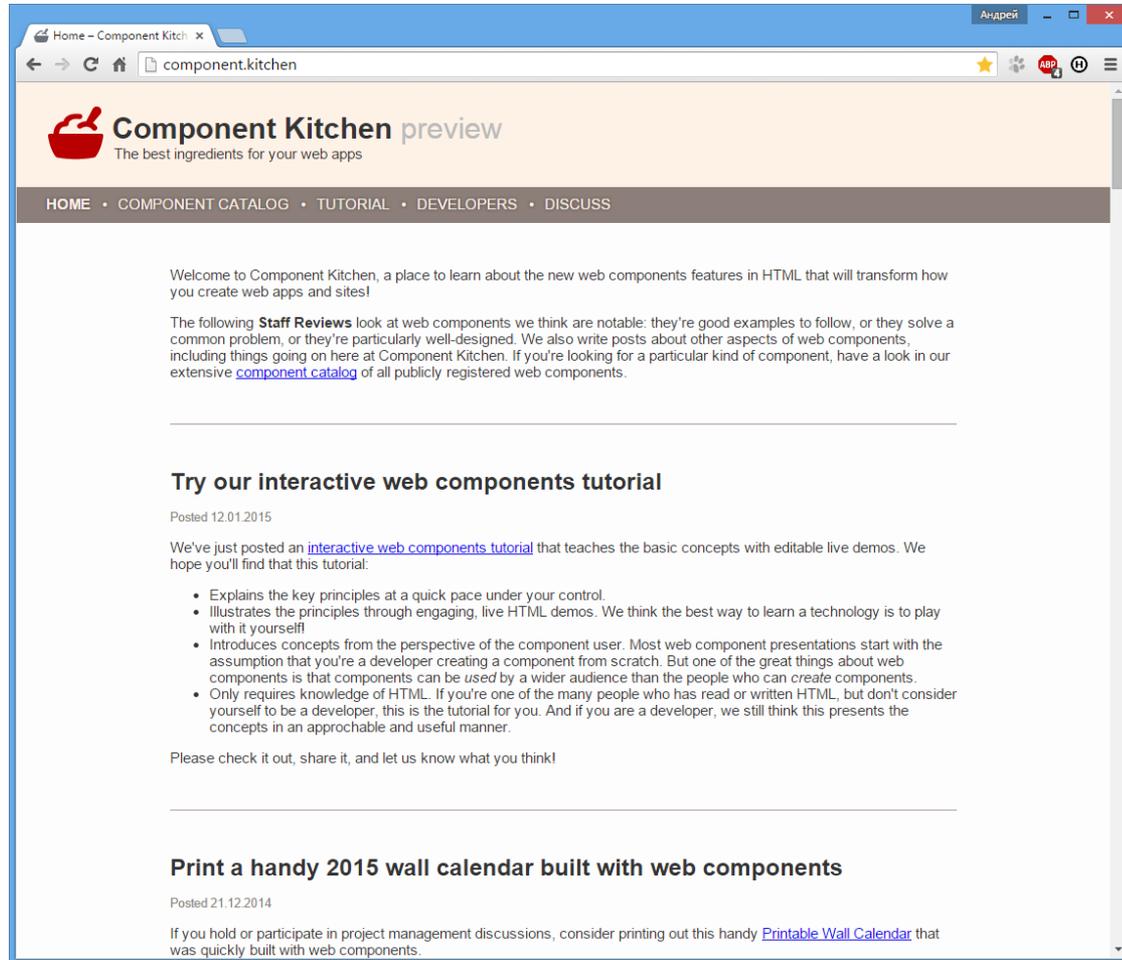
# webcomponents.org

The screenshot shows the webcomponents.org website in a browser window. The browser's address bar displays 'webcomponents.org'. The website's navigation menu includes 'HOME', 'POLYFILLS', 'ARTICLES', 'PRESENTATIONS', 'PODCASTS', and 'RESOURCES'. A search bar with 'Google Custom Search' is visible. The main content area features a large logo of two interlocking wrenches on a red square, with the text 'WebComponents.org' and the tagline 'a place to discuss and evolve web component best-practices'. Below this, there are three main sections: 'POLYFILLS', 'DISCOVER', and 'ARTICLES'. The 'POLYFILLS' section includes a description of webcomponent.js polyfills, installation instructions for Bower and npm, and a download button for 'webcomponents.js' (version 0.6.1). The 'DISCOVER' section lists 'CUSTOMELEMENTS.IO', 'BUILT WITH POLYMER', and 'COMPONENT KITCHEN'. The 'ARTICLES' section features 'WHY WEB COMPONENTS?' with a 'Read More >' link. At the bottom, there are sections for 'BROWSER SUPPORT' and 'SPECS'.

# customelements.io



# component.kitchen



# Pluralsight - HTML5

## Web Component Fundamentals

HTML5 Web Component Fundamentals

Learn how to use the Shadow DOM, Custom Elements, Templates, and Imports to create reusable web components.

by Cory House

Table of contents [Expand all](#) [Start free trial now](#)

▶ Five Problems, One Solution	23:52
▶ Templates	27:50
▶ Custom Elements	43:34
▶ Shadow DOM Fundamentals	34:19
▶ Shadow DOM Insertion Points & Events	53:34
▶ Shadow DOM Styling	51:34
▶ Imports	56:00
▶ Native Alternatives	10:43
▶ Resources	2:05

Course content

- Table of contents
- Description
- Exercise files
- Assessment
- Discussion

More info

Level: **Beginner**

Rating: ★★★★★

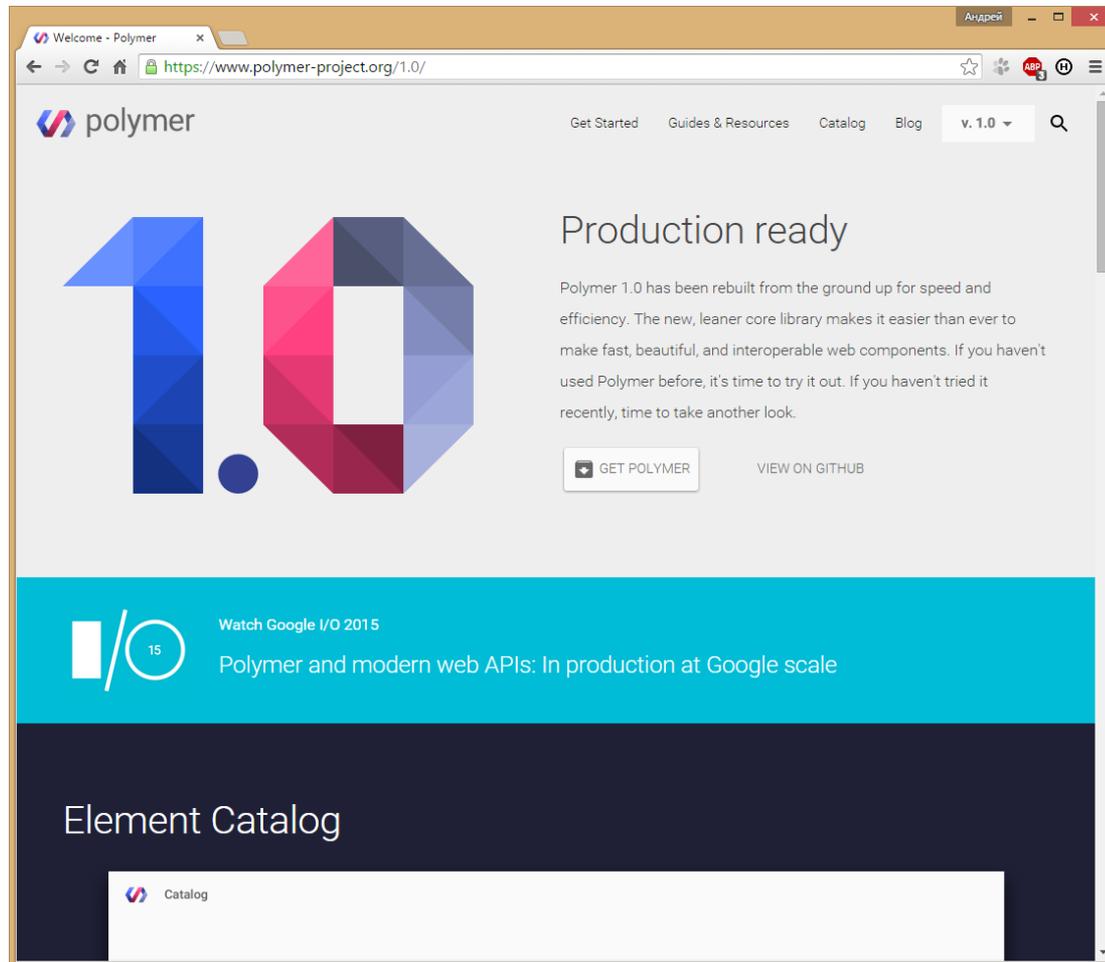
Duration: **5h 3m**

Released: **10 Jan 2015**

Features:

Feedback & Support

# polymer-project.org



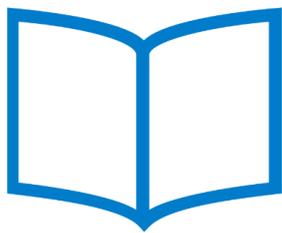
## Резюме

- Интеграция в реальный проект - уже возможна
- Инструменты - активно развиваются
- Библиотеки уже сегодня имеют широкие возможности
- Вы получаете модульность, о которой можно только мечтать!

Андрей Рахманов  
rahmanov@enaza.ru

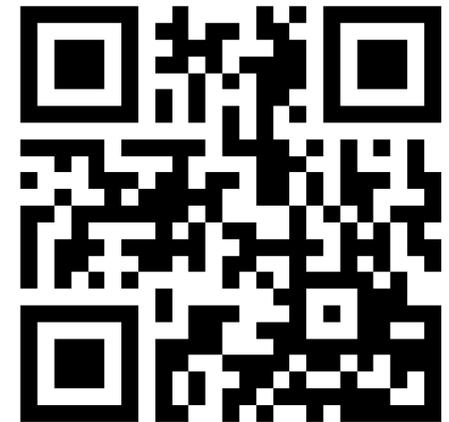
Дмитрий Чертков  
chertkov@enaza.ru

Проект UnderPage  
[goo.gl/McfoKL](https://goo.gl/McfoKL)



UnderPage

Презентация  
[goo.gl/xBTtuu](https://goo.gl/xBTtuu)



Попробуйте Web Components  
Это вкусно!