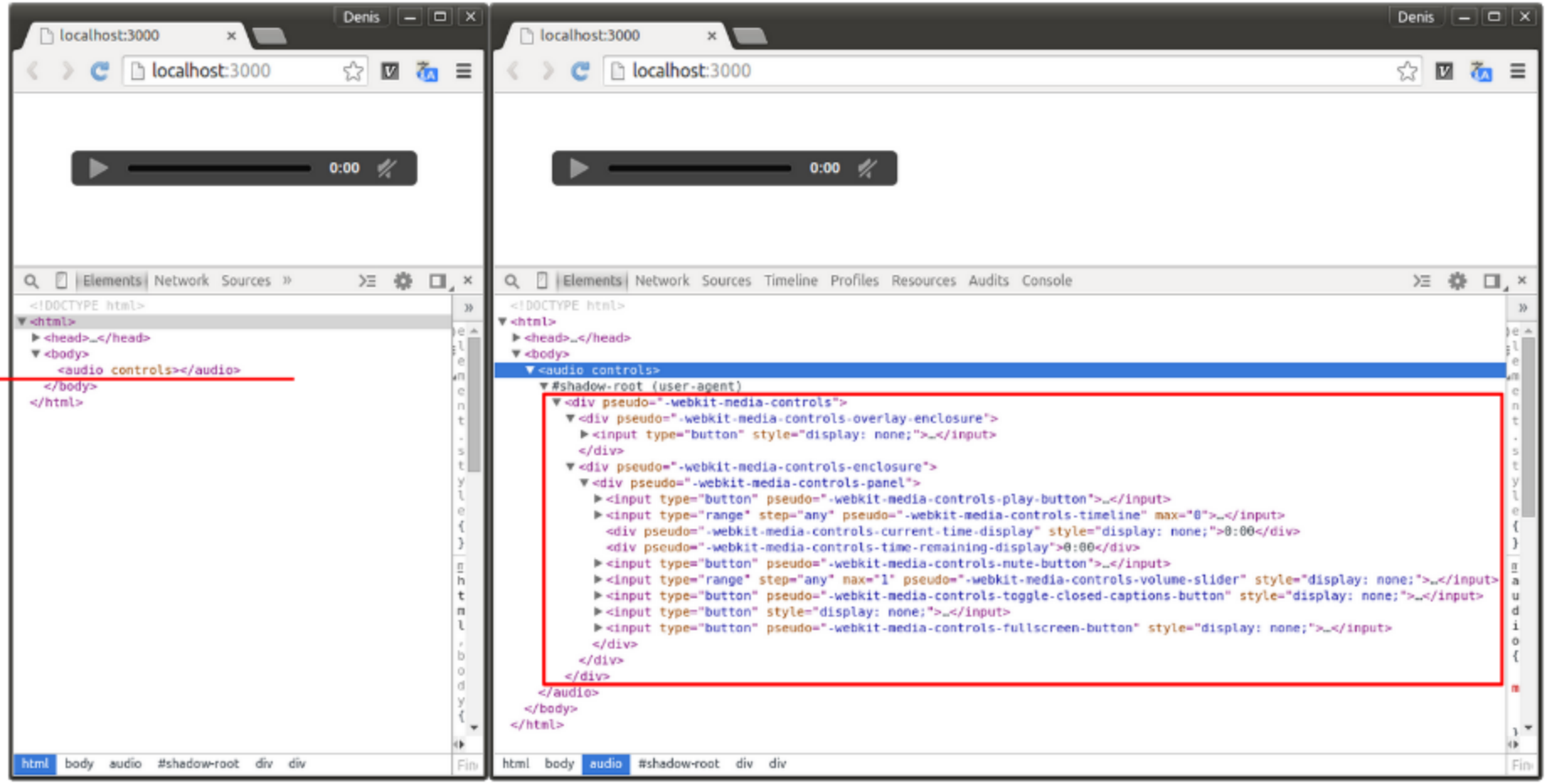
Веб-компоненты - будущее или настоящее?

Денис Иогансен, MoscowJS 20


Denis
localhost:3000
localhost:3000
0:00
Elements
Network
Sources
<!DOCTYPE html>
<html>
<head>…</head>
<body>
<audio controls></audio>
</body>
</html>
html
body
audio
#shadow-root
div
div
Denis
localhost:3000
localhost:3000
0:00
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<!DOCTYPE html>
<html>
<head>…</head>
<body>
<audio controls>
#shadow-root (user-agent)

<input type="button" style="display: none;">…</input>

<input type="button" pseudo="-webkit-media-controls-play-button">…</input>
<input type="range" step="any" pseudo="-webkit-media-controls-timeline" max="0">…</input>
0:00
0:00
<input type="button" pseudo="-webkit-media-controls-mute-button">…</input>
<input type="range" step="any" max="1" pseudo="-webkit-media-controls-volume-slider" style="display: none;">…</input>
<input type="button" pseudo="-webkit-media-controls-toggle-closed-captions-button" style="display: none;">…</input>
<input type="button" style="display: none;">…</input>
<input type="button" pseudo="-webkit-media-controls-fullscreen-button" style="display: none;">…</input>

</audio>
</body>
</html>
html
body
audio
#shadow-root
div
div

Спецификации

- HTML Import
- <template>
- Custom elements
- Shadow DOM

HTML Import

HTML Import

- Загрузка внешнего html документа
- Кэшируется
- Может содержать другие линки (html, css, js)
- Выполнение приостанавливается до полной загрузки

HTML Import

```
01. <link rel="import" href="document.html">
```

<template>

<template>

- Создание шаблонного DOM фрагмента
- Замена `<div hidden>` и `<script type="template">`
- Содержимое не доступно пока не в документе*
- Содержимое не грузится пока не в документе*

* не вызвано .importNode() или .cloneNode()

<template>

```
<template id="temp">
    <div>Привет из шаблона!</div>
</template>
```

```
var temp = document.querySelector('#temp');
var clone = temp.content.cloneNode(true);
document.body.appendChild(clone);
```

Custom elements

Custom elements

- Создание собственного DOM элемента (тега)
- Реализация кастомной логики
- Обработка нативных событий жизненого цикла
- Должен содержать дефис в своём названии

Custom elements

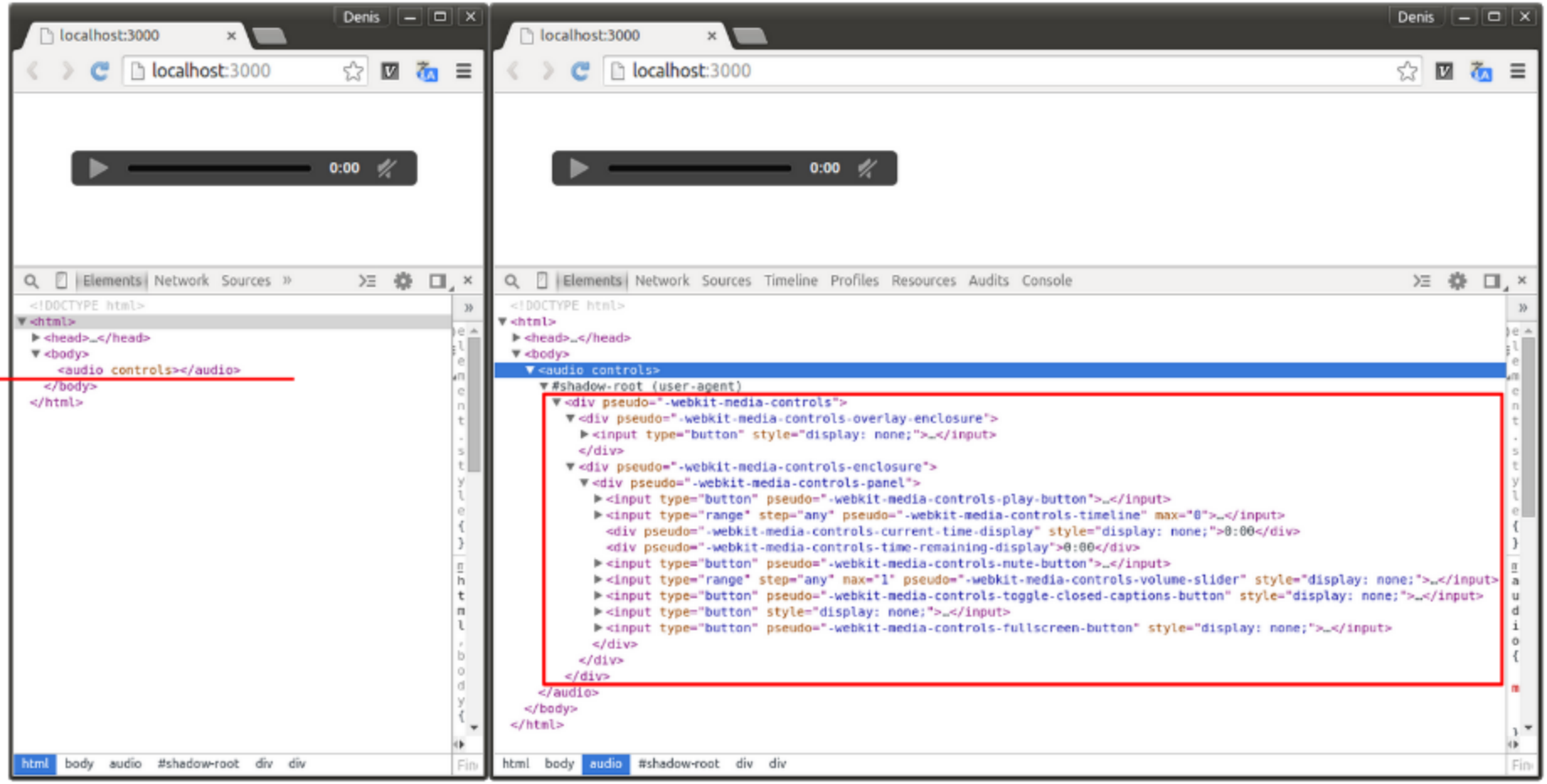
```
01. var proto = Object.create(HTMLElement.prototype);
02. proto.createdCallback = function() {
03.     this.innerHTML = 'Привет из нового элемента!';
04. };
05. document.registerElement('my-elem', {prototype: proto});
```

```
01. <my-elem></my-elem>
```

Shadow DOM

Shadow DOM

- Создание закрытого теневого дерева
- Скрывает внутреннюю реализацию (разметку)
- Изолирует внутренние оформление (стили)
- Позволяет обрабатывать содержимое элемента


Denis
localhost:3000
localhost:3000
0:00
Elements
Network
Sources
<!DOCTYPE html>
<html>
<head>…</head>
<body>
<audio controls></audio>
</body>
</html>
html
body
audio
#shadow-root
div
div
Denis
localhost:3000
localhost:3000
0:00
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<!DOCTYPE html>
<html>
<head>…</head>
<body>
<audio controls>
#shadow-root (user-agent)

<input type="button" style="display: none;">…</input>

<input type="button" pseudo="-webkit-media-controls-play-button">…</input>
<input type="range" step="any" pseudo="-webkit-media-controls-timeline" max="0">…</input>
0:00
0:00
<input type="button" pseudo="-webkit-media-controls-mute-button">…</input>
<input type="range" step="any" max="1" pseudo="-webkit-media-controls-volume-slider" style="display: none;">…</input>
<input type="button" pseudo="-webkit-media-controls-toggle-closed-captions-button" style="display: none;">…</input>
<input type="button" style="display: none;">…</input>
<input type="button" pseudo="-webkit-media-controls-fullscreen-button" style="display: none;">…</input>

</audio>
</body>
</html>
html
body
audio
#shadow-root
div
div

Shadow DOM

```
01. var proto = Object.create(HTMLElement.prototype);
02. proto.createdCallback = function() {
03.     var shadow = this.createShadowRoot();
04.     shadow.innerHTML = 'Привет из закрытого дерева!';
05. };
06. document.registerElement('my-elem', {prototype: proto});
```

```
<template id="temp">
    <div>'Привет изнутри! <content></content></div>
</template>
```

```
var temp = document.querySelector('#temp');
var proto = Object.create(HTMLElement.prototype);
proto.createdCallback = function() {
    var shadow = this.createShadowRoot();
    shadow.appendChild(temp.content.cloneNode(true));
};
document.registerElement('my-elem', {prototype: proto});
```

Использование

```
01. <link rel="import" href="my-elem.html">
02. <my-elem>Привет из содержимого элемента!</my-elem>
03. <my-elem><b>Жирный привет!</b></my-elem>
```

Привет изнутри! Привет из содержимого элемента!

Привет изнутри! **Привет снаружи!**

Приемущества

- Сложное разделяется на простые части
- Повышается семантика
- Легко поддерживать / дорабатывать
- Легко тестировать
- Легко переиспользовать
- `component.kitchen` и `customelements.io`

Текущая поддержка

Браузеры *	IE	Chrome	Firefox	Safari	Android	IOS
HTML Imports	-	+	f	-	+	-
<template>	-	+	+	+	+	+
Custom elements	-	+	f	-	+	-
Shadow DOM	-	+	f	-	+	-

* последние версии

Набор полифилов

Браузеры *	IE	Chrome	Firefox	Safari	Android	IOS
HTML Imports	+	+	+	+	+	+
<template>	+	+	+	+	+	+
Custom elements	+	+	+	+	+	+
Shadow DOM	+	+	+	+	+	+

* последние версии

webcomponent.js

- Отделили от `polymer`
- Содержит полифилы WeakMap(es2015) и MutationObservers
- Полная версия 105 kb
- Лайт версия (без Shadow DOM) 28 kb

Ложка дегтя - Shadow DOM

- Достаточно медленный
- Проблемы с событиями
- Два режима инкапсуляции (открытый и закрытый) *
- Закрытый еще плохо описан *

* черновик на github

<welcome-to-future></welcome-to-future>

<google-map></google-map>

<habrauser-card></habrauser-card>

<web-component></web-component>

Polymer 0.8

- Новая версия (полностью переписанна)
- Увеличена производительность
- Значительно уменьшен размер
- Возможность выбора уровня функциональности

Размер 8 kb

Polymer-micro

Базовый синтаксический сахар

```
01. Polymer({
02.     is: 'my-elem',
03.     created: function() {
04.         this.innerHTML = 'Привет из веб-компоненты!';
05.     }
06. });
```

```
01. <my-elem></my-elem>
```

Наследование от нативных элементов

```
01. Polymer({
02.     is: 'my-elem',
03.     extends: 'input'
04. });
```

```
01. <my-elem></my-elem>
```

Конвертация атрибутов к свойствам

```
01. Polymer({
02.     is: 'my-elem',
03.     properties: {
04.         userName: String,
05.     }
06. });
```

```
01. <my-elem user-name="Denis"></my-elem>
```

Размер 28 kb

Polymer-mini

Расширеная работа с шаблонами

```
<dom-module id="my-elem">
    <template>Привет из веб-компоненты!</template>
</dom-module>
```

```
Polymer({
    is: 'my-elem'
});
```

Работа с содержимым элемента

```
01. <dom-module id="my-elem"><template>
02.     <div>Привет из веб-компоненты!</div>
03.     <content select=".myclass"></content>
04. </template></dom-module>
```

```
01. <my-elem>
02.     <p>Привет без класса!</p>
03.     <p class="myclass">Привет с классом!</p>
04. </my-elem>
```

Изолирование оформления (scoped css)

```
<dom-module id="my-elem">
    <style>/* Оформление */</style>
    <template>Привет из веб-компоненты!</template>
</dom-module>
```

```
Polymer({
    is: 'my-elem'
});
```

Расширеные внутренние свойства

```
01. Polymer({
02.     is: 'my-elem',
03.     properties: {
04.         myProp: {
05.             type: String,
06.             notify: true,
07.             value: 'text' или function() {}
08. } } });
```

Собственное DOM API

```
01. var toLight = document.createElement('div');
02. Polymer.dom(this).appendChild(toLight);
```

```
01. var toLocal = document.createElement('div');
02. var beforeNode = Polymer.dom(this.root).childNodes[0];
03. Polymer.dom(this.root).insertBefore(toLocal, beforeNode);
```

```
01. var allSpans = Polymer.dom(this).querySelectorAll('span');
```

Размер 78 kb

Polymer (full)

Удобную работу с обработкой событий

```
01. Polymer({
02.     is: 'my-elem',
03.     listeners: {
04.         'click': 'handleClick'
05.     },
06.     handleClick: function(e) {
07.         alert("Спасибо за клик!");
08. } });
```

Установка обработчика через атрибуты

```
<dom-module id="my-elem"><template>
    <button on-click="handleClick">Жми!</button>
</template></dom-module>
```

```
Polymer({
    is: 'my-elem',
    handleClick: function(e) {
        alert("Спасибо за клик!");
} });
```

Отслеживание изменения свойств

```
01. Polymer({
02.     is: 'my-elem',
03.     properties: { preload: Boolean, src: String },
04.     observers: {
05.         'preload src': 'updateImage'
06.     },
07.     updateImage: function( preload, src ){ что-то делаем }
08. });
```

Декларативный дата биндинг

```
01. <dom-module id="my-elem"><template>
02.     <p>Привет! Меня зовут <span>{{user.name}}</span></p>
03. </template></dom-module>
```

```
01. Polymer({
02.     is: 'my-elem',
03.     properties: { user: { type: Object,
04.                           value: { name: 'Денис' }
05. } } });
```

А так же:

- Обновление внешних атрибутов
- Вычисляемые свойства
- Набор хелперов
- Глобальные параметры
- Эксперементальные элементы
- Собственные css свойства (экс.)

Напоследок

- `Polymer` больше не эксперимент
- Ожидаемый выход `production release 1.0` - 2 квартал 2015

- Библиотека базовых элементов
- Библиотека в стиле `Material Design`
- Визуальный редактор

Заключение

- Уже используют Github, Salesforce, NewsCorp ...
- Скоро будут Youtube, QuickOffice, GoogleMusic ...
- Много примеров: PolymerMail, Topeka, SantaTracker ...

- Веб-компоненты - это уже настоящее!

Вопросы?

Денис Иогансен

- github: @pofigizm
- twitter: @pofigizm
- pofigizm@gmail.com

- Презентация
- Главная ссылка