Web Components with google Polymer

Polymer Meetup

24.03.2017

RECORD.evolution

Agenda

- Why Web Components?
- Benefits of Polymer
- Basics
- Data Binding
- Web-Component-Tester
- Comparison to other frameworks
- Coding Challenge

Why Web Components?

Why Web Components?

- create your own HTML tags
- reusable
- encapsulated
- interoperable
- unit testing

→ component-based software engineering

Web Components

native browser technology

- Custom Elements
  - writing own elements
- HTML Templates
  - <template> tag: will not be rendered when the page is loaded, can be rendered later with JS
- Shadow DOM
  - encapsulation of DOM and CSS
- HTML Imports
  - enables import of HTML files

Browser support for web components

Browser support	CHROME	OPERA	FIREFOX	SAFARI	EDGE
TEMPLATES	STABLE	STABLE	STABLE	STABLE	STABLE
IMPORTS	STABLE	STABLE	POLYFILL ON HOLD	POLYFILL ON HOLD	POLYFILL CONSIDERING
CUSTOM ELEMENTS	STABLE	STABLE	POLYFILL DEVELOPING	POLYFILL DEVELOPING	POLYFILL CONSIDERING
SHADOW DOM	STABLE	STABLE	POLYFILL DEVELOPING	STABLE	POLYFILL CONSIDERING

Benefits of Polymer

Google Polymer

open source JS library

- convenient API for creating elements
- lifecycle callbacks
- local DOM
- data binding of properties and attributes
- behaviors: reusable code modules for Polymer elements
- utility functions
- element catalog

Basics

Use a custom element from the Polymer Catalog

```
<html>
  <head>
    <!-- 1. Load webcomponents-lite.min.js for polyfill support. -->
    <script src="bower_components/webcomponentsjs/webcomponents-lite.min.js"></script>
    <!-- 2. Use an HTML Import to bring in some elements. -->
    <link rel="import" href="bower_components/paper-button/paper-button.html">
  </head>
  <body>
    <!-- 3. Declare the element. Configure using its attributes. -->
    <paper-button raised>click</paper-button>
  </body>
</html>
```

Demo

Register a custom element

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

<dom-module id="my-element">
    <template>
        <style>
        div {
            background-color: yellow;
        }
        </style>
        <div>
            <h2>Hello World</h2>
            <content></content>
        </div>
    </template>
```

```
. . .
<script>
   Polymer({
        is: 'my-element',
        properties: {
            myProperty: {
                type: String,
                value: 'Some String'
            }
        }
    });

   </script>
</dom-module>
```

… then use it

```
<body>

     <div>
          Look at my element below
     </div>
    <my-element my-property=”test”>
          The element may have content
    </my-element>

...

</body>
```

Demo

Data Binding

Data Binding inside a custom element

bind a property of your custom element to

- text content of local DOM
- attribute of an element
- property of a custom element

one way binding: [[ ]]

two way binding: {{ }}

```
<template>
 <p>[[myProperty]]</p>
 <div style$="background-color: [[myProperty]]"></div>
 <paper-input value="{{myProperty}}"></paper-input>
</template>

<script>
Polymer({
    is: 'binding-example',
    properties: {
        myProperty: {
            type: String,
            value: 'orange'
        }
    }
});
</script>
```

Demo

there is more ...

- binding to objects and arrays
  - slightly different syntax
  - need to notify path
- computed bindings
- binding outside the local dom, i.e. in index.html
  - dom-bind template

dom-repeat

- automatically stamps and binds an instance of a template to an array

```
<dom-module id="repeat-example">
 <template>

  <template is="dom-repeat" items="[[myArray]]">
   <div> element in [[item]]</div>
   <binding-example my-property="[[item]]"></binding-example>
  </template>

 </template>

  <script>
  Polymer({
    is: 'repeat-example',
    properties: {
      myArray: {
        type: Array
      }

      ...
```

Demo

Web Component Tester

Web Component Tester

- Standard Javascript Testframework
  - Mocha
  - libraries like Chai and Sinon for testing
  - asynchronous test, spies, stubs, mocks, …
  - test-fixture (no leaky states)
- Test-Driven-Development
- local testing in all your installed browsers with selenium
- testing in cloud (Sauce Labs), on wide variety of OS & browser configurations
- continuous integration

Comparison with other frameworks

Comparison to other frameworks

	reusable components	scoped css	custom html tags	server-side rendering	element library
React					
Vue					
Polymer					

Conclusion

Pros:

- using the web component standard
- truly sharable interoperable custom-elements
- lots of useful already existent elements and behaviors (-> easy and fast development of responsive web apps)

Cons:

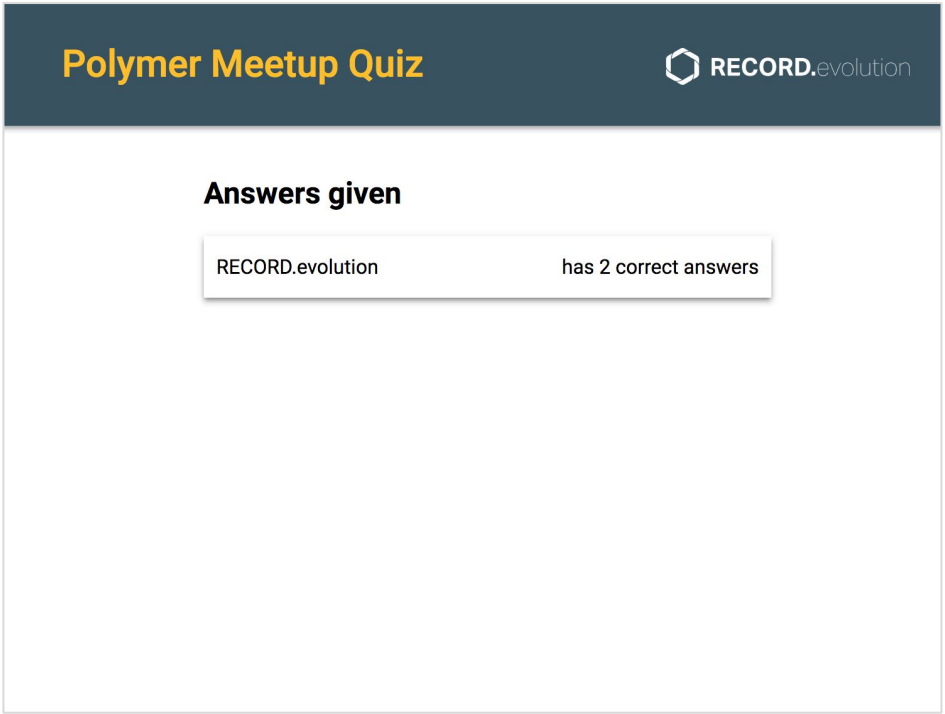
- not full browser support without polyfills
- still in developement, Polymer 2 will not be compatible

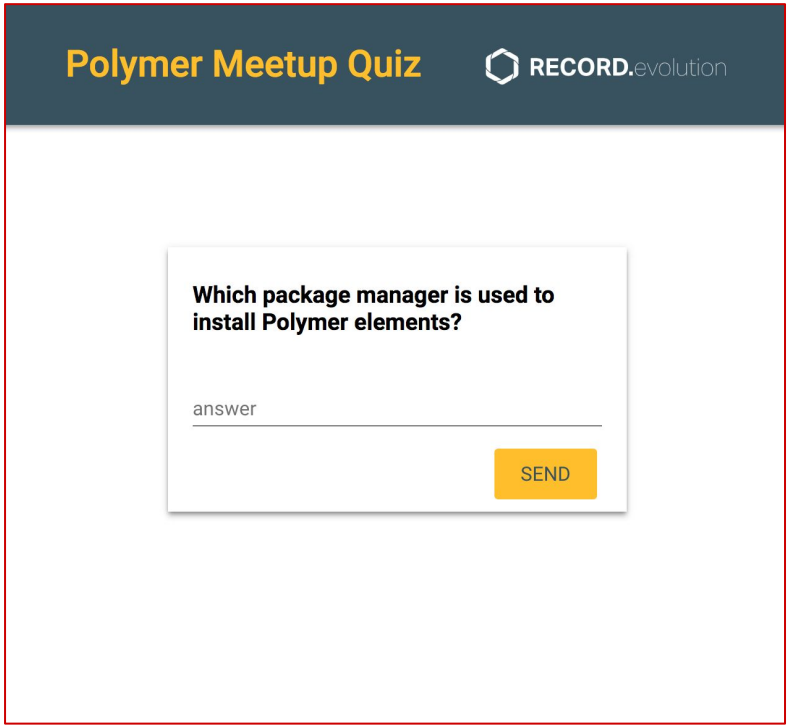
Polymer 2

- ES6 class-style elements
- Implements custom elements v1 specifications
- performance improvements
- breaking changes to Polymer 1

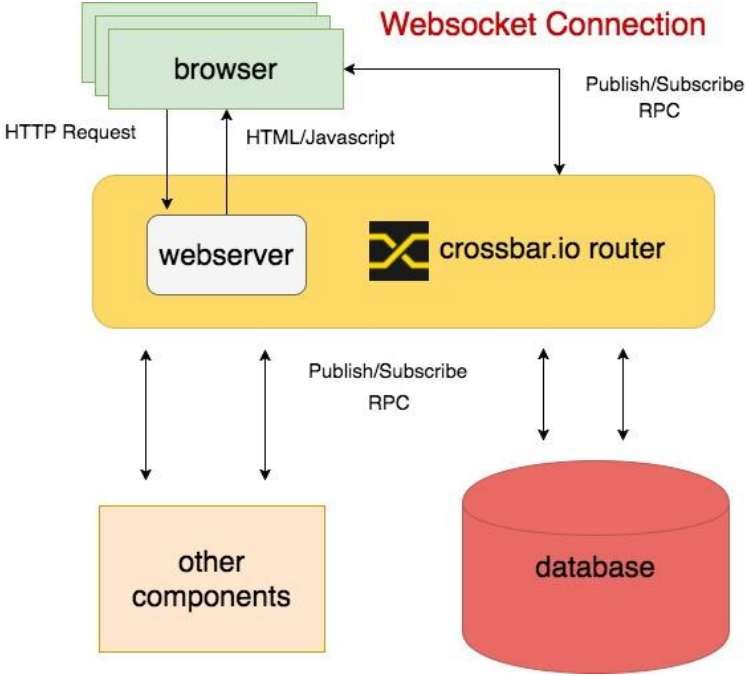
Coding Challenge

Quiz App


Polymer Meetup Quiz
RECORD.evolution
Answers given
RECORD.evolution
has 2 correct answers

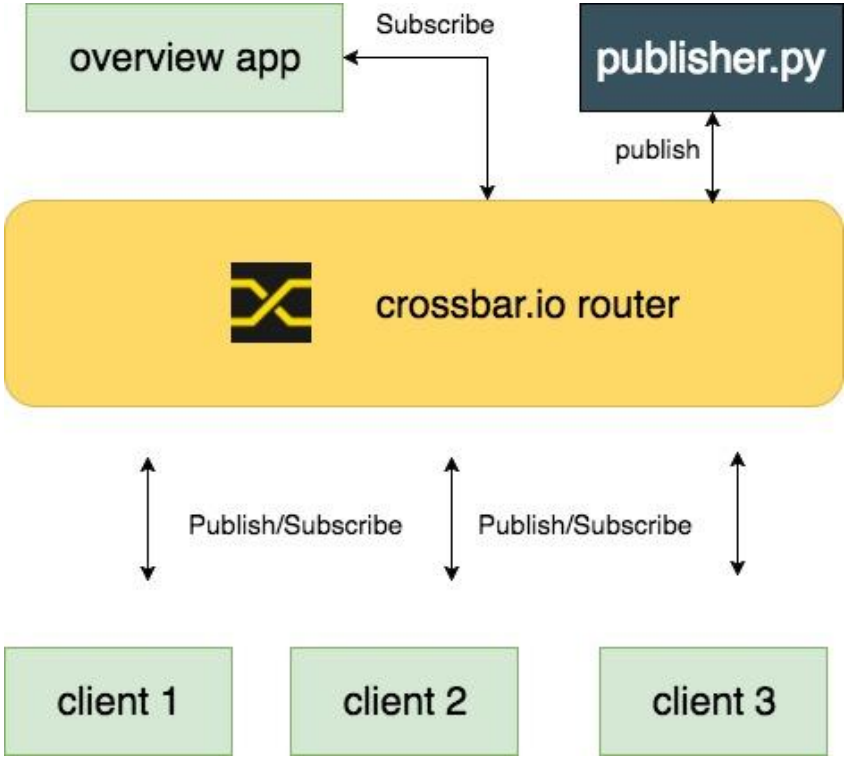

Polymer Meetup Quiz
RECORD.evolution
Which package manager is used to install Polymer elements?
answer
SEND

crossbar.io



- open source application networking platform
- bidirectional connection (websocket)
- publish/subscribe
- remote procedure calls

.. in our case



publisher.py:

1. establish connection to router
2. constantly publish questions to the topic: “re.meetup.question”

client app:

1. establish connection to router (<cb-connect> element)
2. subscribe to topic: “re.meetup.question”
3. publish answer to topic: “re.meetup.answer”

overview app:

1. establish connection to router
2. subscribe to topic: “re.meetup.answer”