Testing Web Components

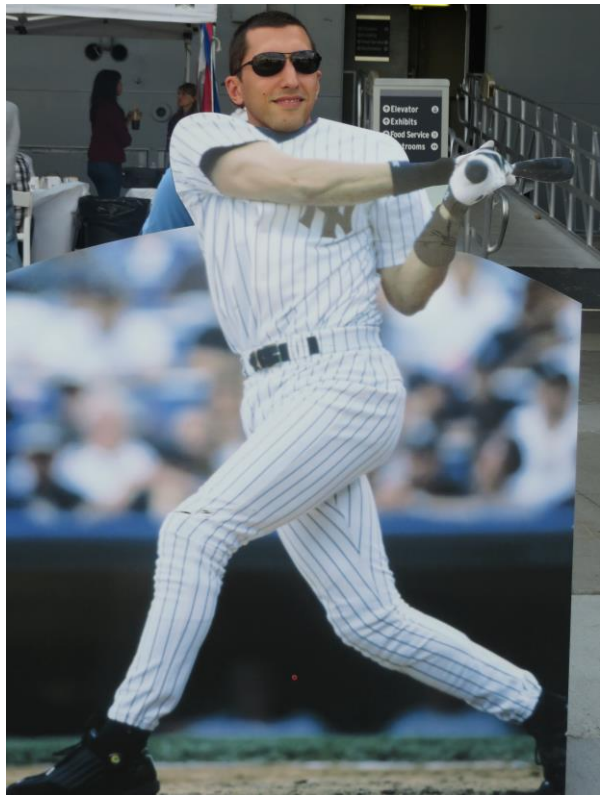
Dario Đurić

Testival Meetup #28

April 4, 2017

About Me

- Over 9 years of experience in IT
- Team lead in Altima
  - Telecom integrator
  - Technologies: Java (Spring), JavaScript (Angular)
- Haven’t been to many meetups :/



Agenda

- Web Components and Polymer
  - What are web components and how Polymer helps
- Creating a Web Component
  - How web components are created in Polymer
- Testing Web Components
  - Using Polymer tools to test web components

Web Components and Polymer

- Web components allow creation of custom HTML tags
- Implementation is hidden under shadow DOM
  - Even standard elements such as <audio> are basically web components composed of other components
- HTML, CSS and JavaScript are all encapsulated within the shadow DOM
  - No JavaScript conflicts
  - No CSS bleed
- Browser support is good, but not ideal

Web Components and Polymer

- Polymer is a lightweight library that helps developing web components across various browsers, and polyfilling features that are not supported
- Developed and actively maintained by Google
- In addition to the core library, Polymer also provides a great deal of custom elements allowing development of nicely designed applications from scratch

Creating a Web Component

- Web component consists of:
  - Imports
  - CSS styles
  - HTML
  - JavaScript API and internal methods and properties

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

<dom-module id="element-name">

  <template>
    <style>
      /* CSS rules for your element */
    </style>

    <!-- local DOM for your element -->

    <div>{{greeting}}</div> <!-- data bindings in local DOM -->
  </template>

  <script>
    // element registration
    Polymer({
      is: "element-name",

      // add properties and methods on the element's prototype

      properties: {
        // declare properties for the element's public API
        greeting: {
          type: String,
          value: "Hello!"
        }
      }
    });
  </script>

</dom-module>
```

Creating a Web Component

- Using web component involves:
  - Importing it
  - Using its HTML tag and any attributes that it provides

```
<html>
  <head>
    <link rel="import" href="element-name.html">
  </head>
  <body>
    <element-name greeting="How are you?"></element-name>
  </body>
</html>
```

- Implementation specifics are completely hidden from the end user
- With reference to the component's instantiated DOM element you may:
  - Override properties
  - Invoke methods

Testing Web Components

- Polymer CLI is used to run tests
  - An all-in-one command interface that covers vast majority of development tasks, including unit testing
- Built on top of popular third-party tools
  - Mocha as a test framework
  - Chai for assertions
  - Sinon for spies, stubs, and mocks
  - Selenium for running tests against multiple browsers

Testing Web Components

- Test fixtures are used to create an instance of the element per each test
  - Prevents shared state between tests

```
<test-fixture id="seed-element-fixture">
  <template>
    <seed-element>
      <h2>seed-element</h2>
    </seed-element>
  </template>
</test-fixture>

<script>
  suite('<seed-element>', function() {
    var myEl;
    setup(function() {
      myEl = fixture('seed-element-fixture');
    });
    test('defines the "author" property', function() {
      assert.equal(myEl.author.name, 'John Smith');
    });
  });
</script>
```

Testing Web Components

- Stub methods enable us to replace default implementations with custom methods

```
setup(function() {
  stub('paper-button', {
    click: function() {
      console.log('paper-button.click called');
    }
  });
});
```

- Stub elements allow us to test elements in isolation
  - “Fake” element can be created within the actual test or externally

```
setup(function() {
  replace('paper-button').with('fake-paper-button');
});
```

Testing Web Components

- We can test asynchronous code by passing the *done* function to test
- Once *done()* is called, the test is complete

```
test('fires lasers', function(done) {
  myEl.addEventListener('seed-element-lasers', function(event) {
    assert.equal(event.detail.sound, 'Pew pew!');
    done();
  });
  myEl.fireLasers();
});
```

Thank You

Any questions?