Creating semantic sites with Web Components and JSON-LD

By Ewa Gasperowicz
Ewa is a contributor to Web**Fundamentals**

With the rising popularity of web components and supporting libraries like Polymer, custom elements become an attractive way to build UI features. The default encapsulation of custom elements makes them especially useful for creating independent widgets.

While some of the widgets are self-contained, many of them rely on external data to present the content to the user - e.g., the current forecast for a weather widget or the address of a company for a map widget.

In Polymer, custom elements are declarative, which means once they are imported into a project, it is very easy to include and configure them in HTML, e.g. by passing the data to populate the widget through an attribute.

It would be great if we could avoid repetition and ensure data consistency, by reusing the same data snippets to populate different widgets as well as inform search engines and other consumers about the content of our page. We can achieve this by using the schema.org standard and the JSON-LD format for our data.

Populating the components with structured data

Typically, JSON is a convenient way to inject data into a particular widget. With the rising support for JSON-LD, we can reuse the same data structures to inform the UI as well as the search engines and other consumers of structured data about the exact meaning of the page's content.

By combining web components with JSON-LD, we create a well-defined architecture for an application:

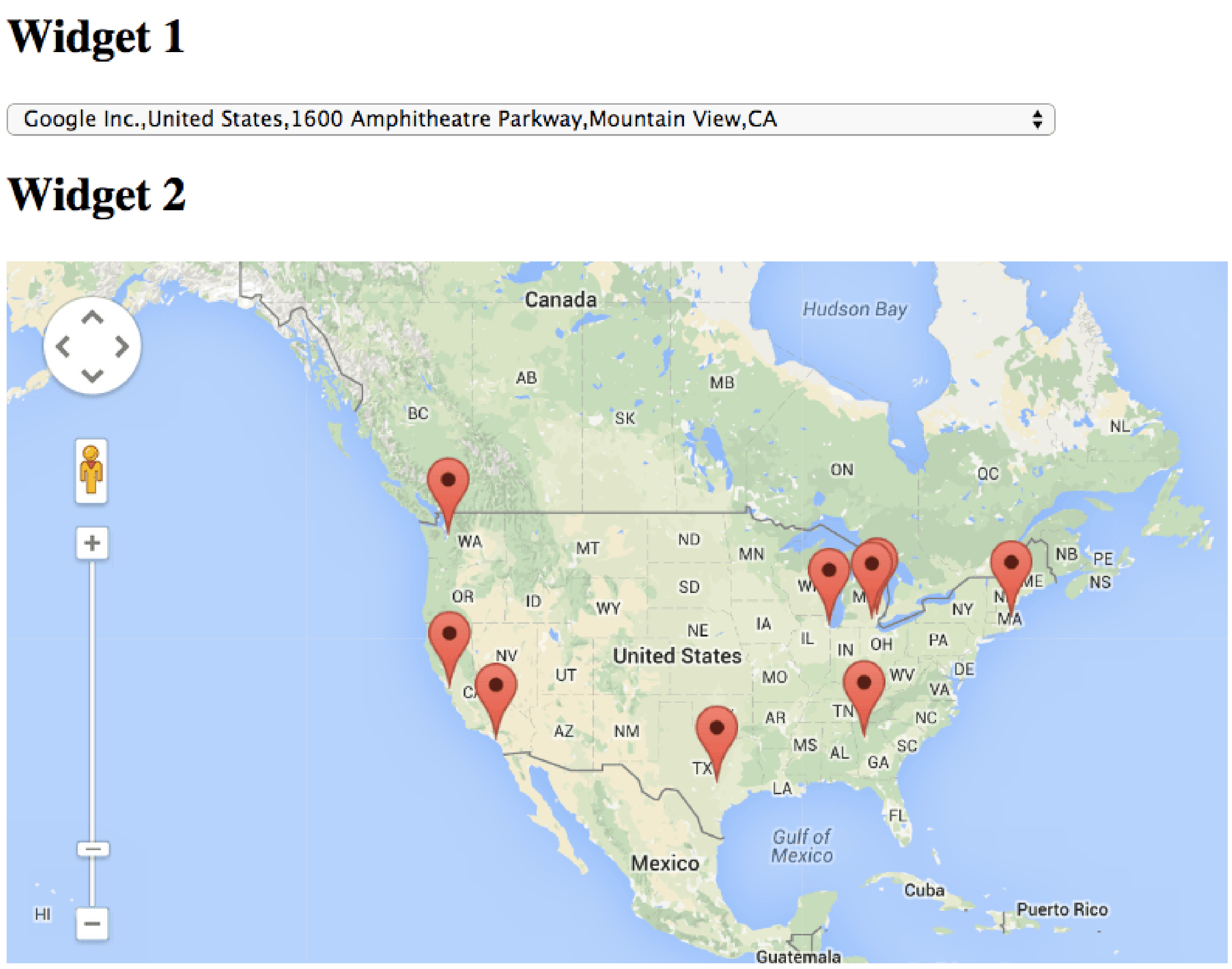
- schema.org and JSON-LD represent the data layer, with schema.org providing the vocabulary for the data and JSON-LD constituting the format and transport for the data;

- custom elements represent the presentation layer, configurable and separated from the data itself.

Example

Let's consider a following example - a page that lists a couple of Google Office locations: http://polymerlabs.github.io/structured-data-web-components/demo/combined-demo.html

It contains two widgets: a map with a pin for every office and a dropdown with the list of locations. It is important that both widgets present the same data to the user and that the page is readable to search engines.



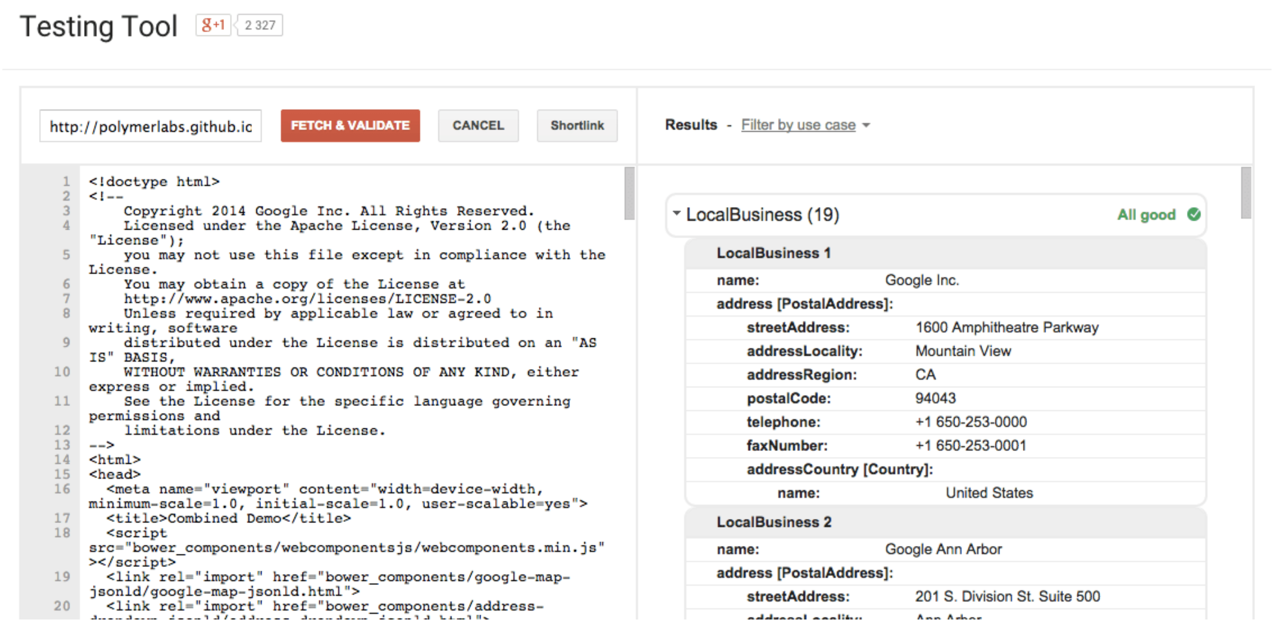
Widget 1

Google Inc.,United States,1600 Amphitheatre Parkway,Mountain View,CA

Widget 2

In this demo we are using LocalBusiness entities to express the meaning of our data, which is the geographical location of some of the Google Offices.

The best way to check how Google is reading and indexing this page is though the new improved Structured Data Testing Tool. Submit the demo's URL in the *Fetch URL* section and click *Fetch and validate*. The section on the right will show you parsed data retrieved from the page along with any errors that may occur. It is a very convenient way to check if your JSON-LD markup is correct and processable by Google.



```
<!doctype html>
<!--
    Copyright 2014 Google Inc. All Rights Reserved.
    Licensed under the Apache License, Version 2.0 (the
"License");
    you may not use this file except in compliance with the
License.
    You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
    Unless required by applicable law or agreed to in
writing, software
    distributed under the License is distributed on an "AS
IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
    See the License for the specific language governing
permissions and
    limitations under the License.
-->
<html>
<head>
  <meta name="viewport" content="width=device-width,
minimum-scale=1.0, initial-scale=1.0, user-scalable=yes">
  <title>Combined Demo</title>
  <script
src="bower_components/webcomponentsjs/webcomponents.min.js"
></script>
  <link rel="import" href="bower_components/google-map-
jsonld/google-map-jsonld.html">
  <link rel="import" href="bower_components/address-
```

LocalBusiness (19) — All good ✓

LocalBusiness 1	
name:	Google Inc.
address [PostalAddress]:	
streetAddress:	1600 Amphitheatre Parkway
addressLocality:	Mountain View
addressRegion:	CA
postalCode:	94043
telephone:	+1 650-253-0000
faxNumber:	+1 650-253-0001
addressCountry [Country]:	
name:	United States

LocalBusiness 2	
name:	Google Ann Arbor
address [PostalAddress]:	
streetAddress:	201 S. Division St. Suite 500

You can read more about the tool and the improvements it introduced in the Webmaster Central blog post.

Linking components to a structured data source

The code for the demo and for the web components used to build it is on Github. Let's look at the **`combined-demo.html`** page source code.

As a first step, we embed the data in the page using a JSON-LD script:

```
<script type="application/ld+json">
{...}
</script>
```

This way we ensure that the data is easily accessible to other consumers supporting schema.org standard and the JSON-LD format, e.g. search engines.

As a second step, we use two web components to display the data:

- address-dropdown-jsonld - This element creates a dropdown with all the locations passed in a "jsonld" attribute.
- google-map-jsonld - This element creates a google map with a pin for every location passed in a "jsonld" attribute.

In order to do so, we import them to our page using HTML imports.

```
<link rel="import" href="bower_components/google-map-jsonld/google-map-json
<link rel="import" href="bower_components/address-dropdown-jsonld/address-dropdow
```

Once they are imported, we can use them on our page:

```
<address-dropdown-jsonld jsonld=""></address-dropdown-jsonld>
<google-map-jsonld jsonld=""></google-map-jsonld>
```

Finally, we hook the JSON-LD data and the elements together. We do so in a polymer-ready callback (it is an event that triggers when the components are ready to use). Because the elements can be configured via attributes, it is enough to assign our JSON-LD data to the appropriate attribute of the component:

```
document.addEventListener('polymer-ready', function() {
    var jsonld = JSON.parse(
        document.querySelector(
            'script[type="application/ld+json"]').innerText);
    document.querySelector('google-map-jsonld').jsonld = jsonld['@graph'];
    document.querySelector('address-dropdown-jsonld').jsonld = jsonld['@graph'];
  });
```

JSON-LD, the powerful brother of JSON

As you probably noticed, this works very similarly to using plain, old JSON to pass data around. JSON-LD has a few advantages though, that are directly derived from its schema.org compatibility:

- The data is structured in an unambiguous way using the schema.org standard. It is a non-trivial advantage, because it ensures you can provide a meaningful and consistent input to any JSON-LD enabled web component.
- The data can be efficiently consumed by search engines, which improves indexing of the page, and may result in rich snippets to be shown in search results.
- If you're writing web components in this way, there is no need to learn or devise a new structure (and documentation) for the data the components expect - schema.org is already doing all the heavy lifting and consensus-building for you. It also makes it easier to build a whole ecosystem of compatible components.

To sum up, JSON-LD and schema.org combined with the web components technology enable building reusable, encapsulated pieces of UI that are developer and search engine friendly.

Create your own components

You can try out the examples on Github or read the Polymer's guide on creating reusable components to start writing your own. Check the Structured Data documentation on developers.google.com to get inspired about various entities you can mark up with JSON-LD.

Consider submitting your shiny new elements to customelements.io for others to enjoy and give us a shout at @polymer or +Polymer community to show off the awesomeness!