Chrome Dev Summit: Polymer declarative, encapsulated, reusable components

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

Polymer is one gateway into the amazing future of Web Components. We want to make it easy to consume and build custom elements. For the past year, the team has been working hard on a set of polyfills for the evolving specifications. On top of that, we've created a convenient sugaring library to make building web components easier. Lastly, we're crafting a set of UI and utility elements to reuse in your apps. At the 2013 Chrome Dev Summit, I dove into the different parts of Polymer and the philosophy behind our "Everything is an element" mantra.

Slides: http://html5-demos.appspot.com/static/cds2013/index.html

"Everything is an element" (from `<select>` to custom elements)

Slides: http://html5-demos.appspot.com/static/cds2013/index.html#6

Building web pages in the 90s was limiting, but powerful. We only had a few elements at our disposal. The powerful part?...**everything was declarative**. It was remarkably simple to create a page, add form controls, and create an "app" without writing gobs of JavaScript.

Take the humble `<select>` element. There is a ton of functionality built into the element, simply by declaring it:

- Customizable through HTML attributes
- Renders children (e.g. `<option>`) with a default UI, but configurable via attributes.

- Useful in other contexts like `<form>`
- Has a DOM API: properties and methods
- Fires events when interesting things happen

Web Components provide the tools to get back to this heyday of web development. One where we can create new elements, reminiscent of `<select>`, but designed for the use cases of 2014. For example, if AJAX was invented today it would probably be an HTML tag (example):

```
<polymer-ajax url="http://gdata.youtube.com/feeds/api/videos/"
              params='{"alt":"json"}'></polymer-ajax>
```

Or responsive elements that data-bind to a `queryMatches` attribute:

```
<polymer-media-query query="max-width:640px" queryMatches="{{isPhone}}">
```

This is exactly the approach we're taking in Polymer. Instead of building monolithic JavaScript-based web apps, let's create reusable elements. Over time, an entire app grows out of composing smaller elements together. Heck, and entire app could be an element:

```
<my-app></my-app>
```

Building web components with Polymer's special sauce

Slides: http://html5-demos.appspot.com/static/cds2013/index.html#37

Polymer contains a number of conveniences for building web component based applications:

- Declarative element registration: `<polymer-element>`
- Declarative inheritance: `<polymer-element extends="...">`
- Declarative two-way data-binding: `<input id="input" value="{{foo}}">`
- Declarative event handlers: `<button on-click="{{handleClick}}">`
- Published properties: `xFoo.bar = 5` <-> `<x-foo bar="5">`
- Property observeration: `barChanged: function() {...}`
- PointerEvents / PointerGestures by default

Moral of the story is that writing Polymer elements is all about being declarative. The less code you have to write, the better ;)

Web Components: the future of web development

Slides: http://html5-demos.appspot.com/static/cds2013/index.html#26

I would be remissed if I didn't give a shout out to the standards behind Web Components. After all, Polymer is based on these evolving foundational APIs.

We're on the cusp of a very exciting time in web development. Unlike other new features being added to the web platform, the APIs that make up Web Components are not shiny or user-facing. They're purely for **developer productivity**. Each of the four main APIs is useful by itself, but together magical things happen!

1. Shadow DOM - style and DOM encapsulation
2. Custom Elements - define new HTML elements. Give them an API with properties and methods.
3. HTML Imports is the distribution model for a package of CSS, JS, and HTML.
4. Templates - proper DOM templating for defining inert chunks of markup to be stamped out later

If you want to learn more about the fundamentals of the APIs, check out webcomponents.org.

Last updated July 2, 2018.