Progressive Web Apps & Polymer

Wendy Ginsberg
Product Manager, Google
@wmginsberg
Who am I?

Wendy Ginsberg
Chrome Web Platform

Google
Who are you?

??????
????????????
?????????
1. Polymer
2. PWAs
3. Polymer + PWAs
1. Polymer
Polymer ==

“An opinionated library that sugars Web Components APIs, to make it easy to build encapsulated, reusable custom elements.”
Polymer library

App Toolbox

webcomponents.org
What problems are we solving?
<h1></h1>
<ul>
</ul>
<p></p>
Building UI tabs should be easy!
<div id="tabs">
  <ul>
    <li><a href="#fragment-1">One</a></li>
    <li><a href="#fragment-2">Two</a></li>
    <li><a href="#fragment-3">Three</a></li>
  </ul>
  <div id="fragment-1">
    <p>First tab is active by default:</p>
    <pre><code>$( "#tabs" ).tabs(); </code></pre>
  </div>
  <div id="fragment-2">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  </div>
  <div id="fragment-3">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  </div>
</div>

<script>
$( "#tabs" ).tabs();
</script>
<div id="tabstrip">
  <ul>
    <li>Tab 1</li>
    <li>Tab 2</li>
  </ul>
  <div id="tabs">
    <script>
      $("#tabstrip").kendoTabStrip({
        animation: {
          // fade-out current tab over 1000 milliseconds
          close: {
            duration: 1000,
            effects: "fadeOut"
          },
          // fade-in new tab over 500 milliseconds
          open: {
            duration: 500,
            effects: "fadeIn"
          }
        }
      });
    </script>
  </div>
</div>
var tabview = new Y.TabView({
  children: [{
    label: 'foo',
    content: '<p>foo content</p>'
  }, {
    label: 'bar',
    content: '<p>bar content</p>'
  }, {
    label: 'baz',
    content: '<p>baz content</p>'
  }]
});
angular.module('tabs', []).
directive('angularTabs', function() {
  return {
    restrict: 'E',
    transclude: true,
    scope: { heading: '@' },
    controller: function($scope, $element) {
      var panels = $scope.panels = [];

      $scope.select = function(panel) {
        panels.forEach.call(panels, function(panel) {
          panel.selected = false;
        });
        panel.selected = true;
      }

      this.addPanel = function(panel) {
        if (panels.length === 0) {
          $scope.select(panel);
        }
        panels.push(panel);
      }

      template:
        '<div id="container">
          ' +
          '<aside>{{heading}}</aside>' +
          '<div class="tab-wrapper">
            ' +
            '<h2 ng-repeat="panel in panels" ng-click="select(panel)" ng-class="{active: panel.selected}">
              ' +
            '</h2>' +
            '<div class="contents" ng-transclude></div>
          ' +
        '</div>' +
        '<script>
          $( "#tabs" )
        </script>
      ' +
      '</div>
    }
  }
});
var tabview = ngModule.directive('angularTabs', function() {
  return {
    restrict: 'E',
    transclude: true,
    scope: { heading: '=' },
    controller: function($scope, $element) {
      var panels = $scope.panels = [];

      $scope.select = function(panel) {
        [].forEach.call(panels, function(panel) {
          panel.selected = false;
        });
        panel.selected = true;
      }

      this.addPanel = function(panel) {
        if (panels.length == 0) {
          $scope.select(panel);
        }
        panels.push(panel);
      }

      template: '
<template>
  <div id="tabs" ng-repeat="panel in panels" ng-click="select(panel)" ng-class="(act ? "active" : "")">
    <div class="tab-wrap"></div>
  </div>
</template>
</div>',
}
});
NOT INTEROPERABLE
How can we redesign HTML for the (modern mobile) web?
Build a better framework.
Build a better framework web.
That sounds hard.

Where do we start?
The web platform is already incredibly powerful.
A study in <select>
Declarative

```html
<select>
  <option disabled>Nylon</option>
  <option selected>Teflon</option>
  <option disabled>Styrofoam</option>
  <option disabled>Lycra</option>
</select>
```
Flexible

Clothing
Nylon
Lycra
Food

<select multiple>
  <optgroup label="Clothing">
    <option>Nylon</option>
    <option>Lycra</option>
  </optgroup>
  ...
</select>
Forgiving

- Nylon
- Teflon
- Styrofoam
- Lycra

Propylene?
<label for="polymer-picker">Pick a Polymer</label>
<select id="polymer-picker">
  <option>Nylon</option>
  <option>Teflon</option>
  <option>Styrofoam</option>
  <option>Lycra</option>
</select>
Programmable

```javascript
> this.selectedIndex;
> 0
> this.addEventListener('change', function() {
>     console.log("I changed!");
> });
```
<select>
an amazing little element

✔ **declarative**, readable

✔ **encapsulated** behavior

✔ **reusable** across contexts

```html
<select>
  <option>Nylon</option>
  <option>Teflon</option>
  <option>Styrofoam</option>
  <option>Lycra</option>
</select>
```
I want more elements like this.
I want to **build** elements like this.
Introduction

You can... with Web Components!

Web components are a set of web platform APIs that allow you to define reusable, encapsulated HTML building blocks for modern web applications. Web components and widgets build on existing web standards, work across modern browsers, and can be used with any JavaScript library or framework that works with HTML.

Web components are based on existing web standards. Features to support web components are currently being added to the HTML and DOM specs, letting web developers easily extend HTML with new elements with encapsulated styling and custom behavior.

Specifications

Web components are based on four main specifications:

Custom Elements

The Custom Elements specification lays the foundation for designing and using new types of DOM elements.
Polymer is an opinionated library that sugars Web Components APIs, to make it easy to build encapsulated, reusable custom elements.”
Web Components + Polymer
Less markup. Less JS. **Less confusion.**

```html
<paper-tabs>
  <paper-tab>KNOWLEDGE</paper-tab>
  <paper-tab>HISTORY</paper-tab>
  <paper-tab>FOOD</paper-tab>
</paper-tabs>
```
2. PWAs
Progressive Web Apps
Progressive Web Apps: Escaping Tabs Without Losing Our Soul

It happens on the web from time to time that powerful technologies come to exist without the benefit of marketing departments or slick packaging. They linger and grow at the peripheries, becoming old-hat to a tiny group while remaining nearly invisible to everyone else. Until someone names them.

This may be the inevitable consequence of a standards-based process and unsynchronized browser releases. We couldn’t keep a new feature secret if we wanted to, but that doesn’t mean anyone will hear about it. XMLHttpRequest was available broadly since IE 5 and in Gecko-based browsers from as early as 2000. “AJAX” happened 5 years later.

This eventual adding-up of new technologies changes how we build and deliver experiences. They succeed when bringing new capabilities while maintaining shared principles:
an app store

- **Linkable**: meaning they’re zero-friction, zero-install, and easy to share. The social power of URLs matters.

These apps aren’t packaged and deployed through stores, they’re just websites that took all the right vitamins. They keep the web’s ask-when-you-need-it permission model and add in new capabilities like being top-level in your task switcher, on your home screen, and in your notification tray. Users don’t have to make a heavyweight choice up-front and don’t implicitly sign browsers from as early as 2006. A/V happened 5 years later.

This eventual adding-up of new technologies changes how we build and deliver experiences. They succeed when bringing new capabilities while maintaining shared principles:
AliExpress Case Study

2X increase in page visits per session

104% increase in conversion rate for new users

https://developers.google.com/web/showcase/
Konga & Jumia
Case Study

Loads 2x faster than their previous web site

Uses 6x less data than their native app counterparts

https://developers.google.com/web/showcase/
How did they do it?
How did they do it? PWAs
Reliable
Fast
Engaging
Reliable

What?
Load instantly, regardless of network connection

Why?
Send less data for initial page loads, save your users money

How?
Service Workers
I implemented service workers to allow for saving blog posts offline, and then wrote about it 😊

una.im/save-offline/
The Service Worker

If you're wondering what a service worker is, it's like a little alien that lives on your page and relays messages for you. It can detect when you have an Internet connection and

una.im/save-offline/
Fast

What?
No janky scrolling or slow to respond interfaces.

Why?
53% of users will abandon a site if it takes longer than 3 seconds to load!

How?
PRPL pattern
(Push, Render, Pre-cache, Lazy load)
Push using HTTP2/push
Render only the critical route or bundle
Pre-cache the necessary files with a Service Worker
Lazy load your content
Engaging

What?
Live anywhere an app can, send notifications to the user

Why?
Increase transactions and interactions with your app

How?
Web Push Notifications API, Add to Home screen
Google Chrome To Make Web Apps As Powerful As Native Ones

5 February 2017, 6:00 pm EST  By Maricet Francisco  Tech Times

GOOGLE’S PROGRESSIVE WEB APPS TURN MOBILE SITES INTO ANDROID APPS

By Adam Ismail  —  January 23, 2017  9:30 AM

Google is finally making web apps first-class citizens on Android

by Chaim Gartenberg  |  @ogartenberg  |  Feb 3, 2017, 11:41 am EST

Chromium Blog

News and developments from the open source browser project

Integrating Progressive Web Apps deeply into Android

Thursday, February 2, 2017

In 2015, we added a new feature to Chrome for Android that allows developers to prompt users to add their site to the Home screen for fast and convenient access. That feature uses an Android shortcut, which means that web apps don’t show up throughout Android in the same way as installed native apps. For example, many developers have asked for their web app to show up in the app drawer section of the launcher. These differences can be confusing for users and prevent native experience from being delivered outside the web. To address the web app frustration, we’re introducing a new feature in Chrome 58: Progressive Web Apps. These apps get an upgrade on the Home screen, and they function more like traditional Android apps.
Video: Web Push Notifications (I/O 2016)

By Pete LePage

Pete is a Developer Advocate

Push notifications are an incredibly effective way to build deeper user engagement with your application, and are now available on the web. In this video, we'll take a look at how they work and deep-dive into how to implement push notifications in web applications, from beginning to end.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated February 6, 2017.
1. Overview

What you'll learn

This codelab will walk you through adding items to a web app that Chrome requires before it will prompt users to add the app to their home screens. Specifically:

- Web app manifest
- Service worker

HTTPS

Add to home screen has one additional requirement: the content must be server over HTTPS. Setting that up requires server work that's beyond the scope of this codelab. Fortunately, this isn't a requirement when serving off localhost, which is what we'll use.

This codelab won't show you every detail about how to build these. What we'll focus on is how to make these work together to craft a user experience.

Is this a progressive web app?

Well, yes and no. Everything I'm going to show you is a requirement for a progressive web app. It will make your site more app-like, but it won't actually be a progressive web app.

What you'll need

- Chrome 47 or above
- Web Server for Chrome, or use your own web server of choice.
What do they look like?
We’re here to help!
developers.google.com/web
Site's content is indexed by Google

To Test:
- Use the Fetch & Google tool to preview how Google will see your site when it is crawled.

To Fix:
- Google's indexing system does not crawl JavaScript but some issues may need to be fixed to make content accessible. For example, if you are using new browser features like the Fetch API, ensure that they are fulfilled in browsers without support.

Schema.org metadata is provided where appropriate

To Test:
- Use the testing tool to ensure title, image, description etc. are available.

To Fix:
- Markup the content. For example:
  - A recipe app should have the Recipe type markup for Rich Cards.
  - A news app should have the NewsArticle type markup for Rich Cards and for AMP support.
  - An ecommerce app should have the Product type markup for Rich Cards.

Social metadata is provided where appropriate

To Test:
- Open a representative page in Facebook's carousel and ensure it looks reasonable.
- Check that Twitter Cards metadata is present (for example `<meta name="twitter:card" content="summary" />`) if you feel it would be appropriate.

To Fix:
- Mark up content with OpenGraph tags and as advised by Twitter.

Canonical URLs are provided when necessary

This is only necessary if your content is available at multiple URLs.

To Test:
- Determine whether any piece of content is available at two different URLs.
- Open both of these pages and ensure they use `<link rel="canonical">` tags in the head to indicate the canonical version.
Here’s a head start

https://www.polymer-project.org/1.0/toolbox/
3. Polymer + PWAs
with the Polymer App Toolbox
Polymer App Toolbox is a collection of components, tools and templates for building Progressive Web Apps with Polymer. App Toolbox features:

- Component-based architecture using Polymer and web components.
- Responsive design using the `app-layout components`.
- Modular routing using the `<app-route>` elements.
- Localization with `<app-localize-behavior>`.
- Turnkey support for local storage with `app storage elements`.
- Offline caching as a progressive enhancement, using service workers.
- Build tooling to support serving your app multiple ways: unbundled for delivery over HTTP/2 with server push, and bundled for delivery over HTTP/1.

You can use any one of these components separately, or use them together to build a full-featured Progressive web app. Most importantly, each component is additive. For a simple app you may only need app-layout. As it gets more complicated, you can add routing, offline caching, and a high-performance server as required.
Build tooling to support serving your app multiple ways: unbundled for delivery over HTTP/2 with server push, and bundled for delivery over HTTP/1.

You can use any one of these components separately, or use them together to build a full-featured Progressive web app. Most importantly, each component is additive. For a simple app you may only need app-layout. As it gets more complicated, you can add routing, offline caching, and a high-performance server as required.

To get a feel for these components in action, you can try out the Shop demo. Shop is a full-featured e-commerce Progressive web app demo built using the Toolbox. Read about how it’s built in Case study: the Shop app.

To get started with the App Toolbox, visit Build an app with App Toolbox.

Or read on to find out about Responsive app layout.
Build an app with App Toolbox

Step 1. Get set up

The Polymer App Toolbox is a collection of components, tools and templates for building Progressive Web Apps with Polymer.

Follow the instructions below to install, build, and deploy a project using an App Toolbox template in less than 15 minutes.

Install the Polymer CLI

1. Install the LTS version (4.x) of Node.js. The current version (6.x) should work, but is not officially supported. Versions below LTS are not supported.

2. If you don't have bower installed, install it
3. Install the Polymer CLI

```
npm install -g polymer-cli
```

**Initialize your project from a template**

1. Create a new project folder to start from

```
mkdir my-app
cd my-app
```

2. Initialize your project with an app template

```
polymer init starter-kit
```

**Serve your project**

The App Toolbox templates do not require any build steps to get started developing. You can serve the application using the Polymer CLI, and file changes you make will be immediately visible by refreshing your browser.
How can I evaluate my PWA?

https://developers.google.com/web/tools/lighthouse/
CSS Flexbox: New intrinsic size algorithm
The CSS Flexbox specification changed the algorithm for calculating a flexbox's intrinsic size (see below for th...

CSS Generated Content for Paged Media Module
CSS properties helpful for printed publication. We only implement paged-x / paged-y from the whole spec.

CSS Logical Properties: Inline/Block size
The CSS Logical Properties introduces (inline-block-size) and (min, max)(inline-block-size), which provide the...

CSS Logical Properties: margin{-block,inline}{-start,-end}
The CSS Logical Properties introduces margin{-block,inline}{-start,-end}, which provide the author with the abil...

CSS filter() image function
The function allows filtering an CSS input image with a set of filter functions. The used filter functions are the ...

Disallow spaces and other bad characters in hostnames
Chrome currently allows some technically invalid characters in hostnames, which it percent-escapes. These h...

Geometry interfaces
This specification describes several geometry interfaces for the representation of points, rectangles, quadrilate...

PaymentComplete result "unknown"
Replace the PaymentComplete enum value "" with the more descriptive "unknown".

UIEvents Keyboard Events
<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS Flexbox: New intrinsic size algorithm</td>
<td></td>
</tr>
<tr>
<td>CSS Generated Content for Paged Media Module</td>
<td></td>
</tr>
<tr>
<td>CSS Logical Properties: Inline/Block size</td>
<td></td>
</tr>
<tr>
<td>CSS Logical Properties: margin( block,-inline)(-start,-end)</td>
<td></td>
</tr>
<tr>
<td>CSS filter() image function</td>
<td></td>
</tr>
<tr>
<td>Disallow spaces and other bad characters in hostnames</td>
<td></td>
</tr>
<tr>
<td>Geometry interfaces</td>
<td></td>
</tr>
<tr>
<td>PaymentComplete result &quot;unknown&quot;</td>
<td></td>
</tr>
<tr>
<td>UIEvents Keyboard Events</td>
<td></td>
</tr>
</tbody>
</table>

For more information, visit [Chrome Platform Status](https://www.chromestatus.com/).
1. Polymer 😎
2. PWAs 😎
3. Polymer + PWAs 😎
Thank you!

Wendy Ginsberg
@wmginsberg