Building Progressive Web Apps with Polymer

Rob Dodson

@rob_dodson  +RobDodson
So I was on vacation.
So I was on vacation. In the Netherlands.
Progressive Web Apps are...

Responsive
Connectivity independent
App-like-interactions
Etc...

*Progressive Apps: Escaping Tabs Without Losing Our Soul* -
Progressive Web Apps are...

“just websites that took all the right vitamins”
A Progressive Web App built with Polymer
Responsive
Load Fast
Work Offline
Installable
Engaging

our focus for today
Make It Responsive
With app-layout Elements
We want to provide elements that can support any app's layout, not just for Material Design.
<!-- top toolbar -->
<app-toolbar>
  <!-- menu button -->
  <paper-icon-button icon="menu">
  </paper-icon-button>
</app-toolbar>
<!-- top toolbar -->
<app-toolbar>
  <!-- menu button -->
  <paper-icon-button icon="menu">
  </paper-icon-button>
</app-toolbar>

<!-- bottom toolbar -->
<app-toolbar>
  <div>ZUPERKÜLBLOG</div>
</app-toolbar>
<app-header>
  <!-- top toolbar -->
  <app-toolbar>
    <!-- menu button -->
    <paper-icon-button icon="menu">
    </paper-icon-button>
  </app-toolbar>

  <!-- bottom toolbar -->
  <app-toolbar>
    <div>ZUPERKÜLBLOG</div>
  </app-toolbar>
</app-header>
add attributes for **effects**

```xml
<app-header-layout>
  <app-header fixed waterfall>
    <!-- top toolbar -->
    <app-toolbar>…</app-toolbar>
    <!-- bottom toolbar -->
    <app-toolbar>…</app-toolbar>
  </app-header>

  <main>
    <!-- Site content -->
  </main>
</app-header-layout>
```
App Layout · polymerlabs.github.io/app-layout

A collection of elements, along with guidelines and templates that can be used to structure your app’s layout.
Responsive
Load Fast
Work Offline
Installable
Engaging
Make It Load Fast
With async patterns
<!doctype html>
<html lang="en">
  
  <head>
    <script src="webcomponents-lite.min.js"></script>
    <link rel="import" href="elements/elements.html"/>
  </head>

  <body unresolved>
    <!-- site content -->
  </body>

</html>
Make sure you’re not blocking the renderer waiting on polyfills or elements
<!doctype html>
<html lang="en">

<head>
  
  <script src="webcomponents-lite.min.js"></script>
  
  <link rel="import" href="elements/elements.html">

</head>

<body unresolved>

  <!-- site content -->

</body>

</html>
<!doctype html>
<html lang="en">

<head>
    <script src="webcomponents-lite.min.js" async></script>
    <link rel="import" href="elements/elements.html" async>
</head>

<body unresolved>
    <blog-app></blog-app>
</body>

</html>
Should you load the polyfills in every browser?
Should you load the polyfills in every browser? No.
<!doctype html>
<html lang="en">
<head>
    <script src="webcomponents-lite.min.js" async></script>
    <link rel="import" href="elements/elements.html" async>
</head>

<body unresolved>
    <!-- site content -->
</body>

</html>
<!doctype html>
<html lang="en">

<head>
  <link rel="import" href="elements/elements.html" async>
</head>

<body unresolved>
  <!-- site content -->
</body>

</html>
<!doctype html>
<html lang="en">

<head>
    <link rel="import" href="elements/elements.html" async>
</head>

<body unresolved>
    <!-- site content -->
    <script src="app.js" async></script>
</body>

</html>
```javascript
var webComponentsSupported = ('registerElement' in document && 'import' in document.createElement('link') && 'content' in document.createElement('template'));
```

do we **need** polyfills?
var webComponentsSupported = ('registerElement' in document && 'import' in document.createElement('link') && 'content' in document.createElement('template'));

if (!webComponentsSupported) {
    var script = document.createElement('script');
    script.async = true;
    script.src = 'webcomponents-lite.min.js';
    script.onload = finishLazyLoading;
    document.head.appendChild(script);
} else {
    finishLazyLoading();
}
<!doctype html>
<html lang="en">
  
  <head>
    <link rel="import" href="elements/elements.html" async>
  </head>

  <body unresolved>
    <!-- site content -->
    <script src="app.js" async></script>
  </body>

</html>
<!doctype html>
<html lang="en">

<head>
  <link rel="import" href="elements/elements.html" async>
</head>

<body unresolved>
  <!-- site content -->
  <script src="app.js" async></script>
</body>

</html>
unresolved is a bottleneck
<!doctype html>
<html lang="en">
  
  <head>
    <link rel="import" href="elements/elements.html" async>
  </head>

  <body>
    <!-- site content -->
    <script src="app.js" async></script>
  
  </body>

</html>
Use the :unresolved pseudo-class to style unupgraded elements*

*browser must support Custom Elements
<!doctype html>
<html lang="en">
  
  <head>
    <link rel="import" href="elements/elements.html" async>
    <style>
      app-header-layout[unresolved] { height: 192px; background: #FFF; }
    </style>
  </head>

  <body>
    <app-header-layout unresolved>
      ...
    </app-header-layout>
    <script src="app.js" async></script>
  </body>

</html>
Create a **skeleton** that mimics the look of your app shell.
<html lang="en">
<head>
  <link rel="import" href="elements/elements.html" async>
  <style>
    #skeleton .header { height: 192px; background: #FFF; }
  </style>
</head>
<body>
<div id="skeleton">
  <div class="header">…</div>
</div>
<blog-app></blog-app>
<script src="app.js" async></script>
</body>
</html>
Responsive
Load Fast
Work Offline
Installable
Engaging
Make It Work Offline

With platinum-sw Elements
“A service worker is a script that is run by your browser in the background, separate from a web page”
// updated service worker is activated.
var CACHE_VERSION = 1;
var CURRENT_CACHES = {
'read-through': 'read-through-cache-v' + CACHE_VERSION
};

self.addEventListener('activate', function(event) {
  // Delete all caches that aren't named in CURRENT_CACHES.
  // While there is only one cache in this example, the same logic will handle the case where
  // there are multiple versioned caches.
  var expectedCacheNames = Object.keys(CURRENT_CACHES).map(function(key) {
    return CURRENT_CACHES[key];
  });

  event.waitUntil((
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (expectedCacheNames.indexOf(cacheName) == -1) {
            // If this cache name isn't present in the array of "expected" cache names,
            // then delete it.
            console.log('Deleting out of date cache:', cacheName);
            return caches.delete(cacheName);
          }
        })
      ).catch(error => console.error(error))
    )
  ));
});
Libraries like sw-toolbox and sw-precache abstract these low level complexities
Pt
Platinum Elements
Push notifications, offline caching, and more.
Elements to turn your web page into a true web app, with push, offline, bluetooth and more.
Elements to turn your web page into a **progressive** true web app, with push, offline, bluetooth and more.
<!-- In your index.html -->

<platinum-sw-register></platinum-sw-register>
<platinum-sw-register auto-register skip-waiting clients-claim>
</platinum-sw-register>
<platinum-sw-register auto-register skip-waiting clients-claim>

<platinum-sw-cache></platinum-sw-cache>

</platinum-sw-register>

Take advantage of sw-toolbox
Caching Strategies

**fastest:** race the cache and network

**networkFirst:** try the network, then the cache

**networkOnly:** try the network, else fail
Configure your caching strategy.
Prefetch everything for your app shell
Prefetch an array of file paths
<platinum-sw-register auto-register skip-waiting clients-claim>

<platinum-sw-cache default-cache-strategy="fastest" cache-config-file="/cache-config.json">

</platinum-sw-cache>

</platinum-sw-register>

...Or a json file
Example `cache-config.json`

```json
{
  "cacheId": "zuperkulblog",
  "disabled": false,
  "precache": [
    "data\art.json",
    "data\film.json",
    "scripts\app.js",
    ".\",
    "bower_components\webcomponentsjs\webcomponents-lite.min.js"
  ],
  "precacheFingerprint": "847c082ce8e3eb8c54054a7cdc76544e"
}
```
Example `cache-config.json`

```json
{
  "cacheId": "zuperkulblog",
  "disabled": false,
  "precache": [
    "data\art.json",
    "data\film.json",
    "scripts\app.js",
    ".\",
    "bower_components\webcomponentsjs\webcomponents-lite.min.js"
  ],
  "precacheFingerprint": "847c082ce8e3eb8c54054a7cdc76544e"
}
```

**Change this** to tell SW to **precache** files again.
Check out the cache-config gulp task in the Polymer Starter Kit project for an example of generating at build time.
Responsive
Load Fast
Work Offline
Installable
Engaging
Make It Installable
With a Web App install banner
Your App Must...

✓ Have a Web App Manifest
✓ Have a registered Service Worker
✓ Be served over HTTPS
✓ Be visited twice, with at least 5 minutes between visits*

Increasing engagement with Web App install banners –
Your App Must…

✓ Have a Web App Manifest
✓ Have a registered Service Worker
✓ Be served over HTTPS
✓ Be visited twice, with at least 5 minutes between visits*
{
    "name": "Zuperkülblog",
    "short_name": "Zuperkülblog",
    "icons": [{
        "src": "images/touch/icon-128x128.png",
        "sizes": "128x128",
        "type": "image/png"
    },
    ...
    {
        "src": "images/touch/chrome-splashscreen-icon-384x384.png",
        "sizes": "384x384",
        "type": "image/png"
    }],
    "start_url": "/?homescreen=1",
    "background_color": "#3E4EB8",
    "display": "standalone",
    "theme_color": "#FFFFFF"
}
{
  "name": "Zuperkülblog",
  "short_name": "Zuperkülblog",
  "icons": [
    {
      "src": "images/touch/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    },
    ...
    {
      "src": "images/touch/chrome-splashscreen-icon-384x384.png",
      "sizes": "384x384",
      "type": "image/png"
    }
  ],
  "start_url": "/?homescreen=1",
  "background_color": "#3E4EB8",
  "display": "standalone",
  "theme_color": "#FFFFFF"
}
<!-- In your index.html -->

<!-- Web Application Manifest -->

<link rel="manifest" href="manifest.json">
Be sure to also include fallback meta tags.
<!-- Tile color for Win8 -->
<meta name="msapplication-TileColor" content="#3372DF">

<!-- Add to homescreen for Chrome on Android -->
<meta name="mobile-web-app-capable" content="yes">
<meta name="application-name" content="PSK">
<link rel="icon" sizes="192x192" href="images/touch/chrome-touch-icon-192x192.png">

<!-- Add to homescreen for Safari on iOS -->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">
<meta name="apple-mobile-web-app-title" content="Polymer Starter Kit">
<link rel="apple-touch-icon" href="images/touch/apple-touch-icon.png">

<!-- Tile icon for Win8 (144x144) -->
<meta name="msapplication-TileImage" content="images/touch/ms-touch-icon-144x144-precomposed.png">
<!-- Tile color for Win8 -->
<meta name="msapplication-TileColor" content="#3372DF">

<!-- Add to homescreen for Chrome on Android -->
<meta name="mobile-web-app-capable" content="yes">
<meta name="application-name" content="PSK">
<link rel="icon" sizes="192x192" href="images/touch/chrome-touch-icon-192x192.png">

<!-- Add to homescreen for Safari on iOS -->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">
<meta name="apple-mobile-web-app-title" content="Polymer Starter Kit">
<link rel="apple-touch-icon" href="images/touch/apple-touch-icon.png">

<!-- Tile icon for Win8 (144x144) -->
<meta name="msapplication-TileImage" content="images/touch/ms-touch-icon-144x144-precomposed.png">
“But, I’m all lazy... ’n stuff...”
Polymer Starter Kit gives you a working manifest, meta tags, and device icons.
So... Is This Thing Working?

Debugging your banner
Enable the bypass flag to test your banner

Bypass user engagement checks

Use Chrome Sync sandbox

Enable Out-of-process V8 Proxy Resolver.
Watch Out For This Gotcha

Revenge of the offline dino
{ "name": "Zuperkülblog",
"short_name": "Zuperkülblog",
"icons": [ {
  "src": "images/touch/icon-128x128.png",
  "sizes": "128x128",
  "type": "image/png"
},
...,
{ "src": "images/touch/chrome-splashscreen-icon-384x384.png",
  "sizes": "384x384",
  "type": "image/png"
}],
"start_url": "/?homescreen=1",
"background_color": "#3E4EB8",
"display": "standalone",
"theme_color": "#FFFFFF"}
“By default, the request URL must exactly match the URL used to store the cached response, including any query parameters in the search portion of the URL.”
Responsive
Load Fast
Work Offline
Installable
Engaging
Make It Engaging
With Push Notifications
Don’t spam.
Don’t spam. Seriously.
Let the user decide when they want to opt-in to push messaging.
<paper-toggle-button></paper-toggle-button>

<platinum-push-messaging></platinum-push-messaging>
<paper-toggle-button checked="{{pushEnabled}}"> </paper-toggle-button>

<platinum-push-messaging enabled="{{pushEnabled}}"> </platinum-push-messaging>
<paper-toggle-button checked="{{pushEnabled}}"></paper-toggle-button>

<platinum-push-messaging enabled="{{pushEnabled}}"
message-url="/notification-data-data.json"></platinum-push-messaging>
app.get('/notification-data.json', function (req, res) {
    res.json({
        'title': 'Zuperkülblog just posted…',
        'message': 'Demystifying Density by Sebastien Gabriel',
        'url': 'https://zuperkulblog.appspot.com/…',
        'icon': '/images/article/demystifying-192x192.png',
        'tag': 'zuperkulblog-push-notification'
    });
});
Web Fundamentals is a comprehensive resource for web development best practices, designed to help you add the right features and experiences to your web project. If you’re new to web development or just looking to make your project better, we’ve got you covered.
There are currently 912 registered listeners

Did you register for cat notifications?

If not, you need to enable push notifications here

github.com/notwaldorf/caturday-post
The food in my bowl
Is old, and more to the point
Contains no tuna.
In Closing...
(yes it’s finally over)
Things are getting kind of awesome
Learn Polymer at your local event, or start one yourself • #itshackademic!

Polymer is a new kind of library, built atop Web Components, that enables developers to create their own encapsulated, custom HTML elements. With Polymer and Web Components, we can componentize the web!
thanks!

@rob_dodson
+RobDodson

source
github.com/polymerlabs/zuperkulblog-progressive

credits
Images by Paul F., Jinhwan Kim, Wolff, Pirog Tetyana, Dani Rolli, Julien Deveaux, Matt Brooks, Pablo Rozenberg, Ribbla Team, Jennifer Goodman, Dan Lowenstein, Blaise Sewell, Arturo Alejandro Romo Escartin Project