The Road to the Native Mobile Web

Kenneth Rohde Christiansen

TIZEN™
DEVELOPER CONFERENCE
2014
SAN FRANCISCO

Kenneth Rohde Christiansen

- Web Platform Architect at Intel Europe
- Blink core owner and former active WebKit reviewer
- Works on Chromium, Crosswalk and related projects

Topics for today

- Major advantages of the web platform
- Top issues and lessons learned
- Current industry focus
- Introducing Crosswalk
- An outlook to the future

The web that binds us

The web platform

The web has changed out lives

Transformed businesses

Created innovation…

But the Web itself has also been transformed

The web platform

HTML

The inception

The web platform

DHTML

The dark ages

Lots of animations, scripts (incl. VBScript, etc)

The web platform

AJAX

New potential

XMLHttpRequest, Google Maps, Gmail, browser applications

The web platform

WHATWG

New cooperation, new refinement

Standardization outside the W3C, the birth of what became HTML5

The web platform

HTML5

The word on everyone’s lips

Adopted by the W3C, the birth of the web platform (several specs), CSS3, JS6 and The Living Standard

The web platform

Mobile Web

The full desktop experience in your pocket

Started by iOS, developers realized the potential for cross platform apps, hybrid (PhoneGap, Sencha, etc) created new potentials

Advantages of using the Web Platform

What made the web popular

- Connectivity
- Deep linking
- Indexability

Something most native apps lack

What makes it popular today

- Integration with cloud services
- Flexible layout primitives
- Responsive/adaptable design
- The cross-platform promise
- Deployment and distribution model

Many advantages over native apps, and also reason why many native apps make use of a WebView

“The danger zone” and slow standardization

To understand the web and sometimes the slow process in catching up with native, we need to realize that the web in not curated and **danger lurks around every corner**

You don’t stumble upon a native app before installing it

Other reasons: multiple vendors, standardization, future proofing

Failures along the way

1. No consistent offline story

- appcache became a joke (expiration date, know all resources up front)
- Packaged apps are all offline but lack the deployment/update model of the web

Failures along the way

2. Lack of control

- Window management, orientation locking, avoid device going to sleep and much more

Failures along the way

3. Lack of sensitive API

- Fingerprinting, network info, bluetooth, power, among others

Failures along the way

4. New API rolls out very slow

- Slow standardization, have to fit the web and all platforms
- Polyfilling and prollyfilling is becoming increasingly popular – but enablers are missing

Lessons learned

1. It is close to impossible to roll out new APIs as fast as native

- But with right layering, polyfilling can solve much of this
- The extensiblewebmanifesto.org

Lessons learned

2. Permission model is hard, not but really! really really hard

- Up front vs. when you need?
- We need a model which doesn’t only cover packaged apps
- No one got it right yet

Lessons learned

3. Platforms do differ

- Can screen change orientation? One app at the time?
- Keep the API available everywhere but let the method/promise fail

Lessons learned

4. Developers are fine with special platform code

- For special features that is totally acceptable and preferred – integration
- For core features, like CSS differences, it is not!

Lessons learned

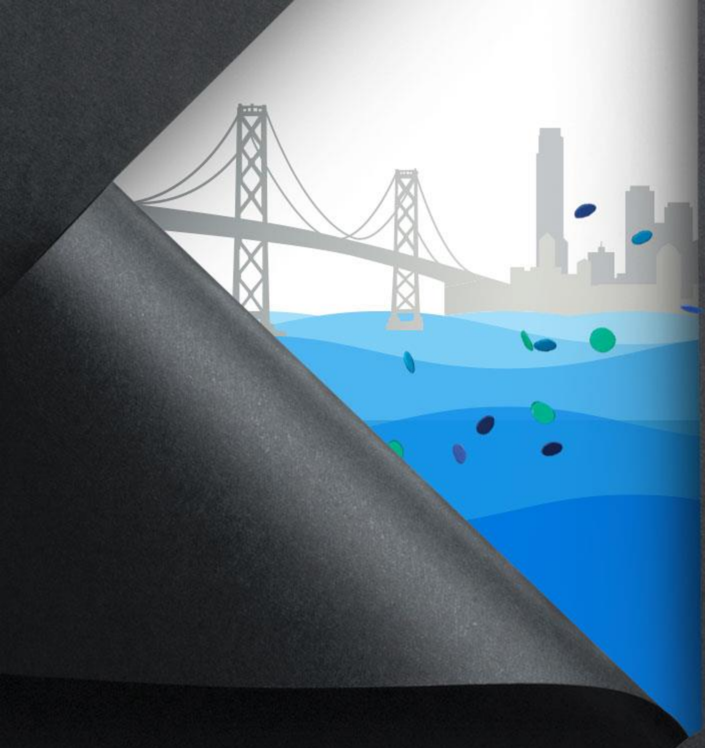
5. The web platform is not layered - and full of magic

- **Hard to understand** what happens below and why some things result in unexpected slow-downs
- Frameworks **are even worse**; they often replace the whole platform and users are lost when they need something slightly different than what it being offered (customization)

Lessons learned

6. **The platform needs a standardized way of creating real reusable UI components with proper encapsulation and nice API**

   - Requires a layered platform
   - Requires control over the graphics pipeline; layout, rasterization, paint



Focus this year

The renewed Google focus for mobile and web apps

Blink developers are now 100% focused on mobile performance

- **Proper layering in the platform**
  - explain new features by using existing ones lower in the stack
- **Remove unused legacy features (if possible)**
- **Shipping and promoting Web Components and Polymer**
- **Allow developers to control the painting**
  - Alternative to relying on the platform (CSS Transforms etc.), allowing to create native like UI components
- **Have predictable performance characteristics**

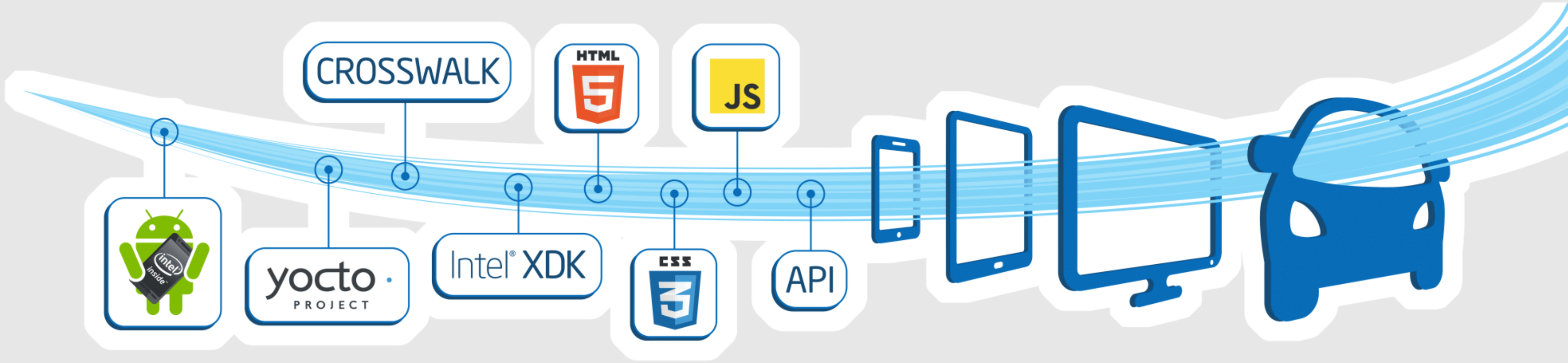
What about the fruit company?

Their focus is not pushing the platform in the same way as Google, but using it where it makes sense for them

- Keep Safari competitive
- Not actively pushing new features that enables apps
- Investigating Web Components (to stay competitive)
- Focus on magazine layouts and CSS which they rely on for products like iBooks (like CSS regions)

The Intel focus

We want to enable web apps everywhere, and the sooner the better



And web apps must run well on IA (devices and cloud)

The Intel focus

How are we getting there

- Work in the W3C working groups like SysApps, promoting native capabilities
- Work on the manifest spec, providing a way to describe an app
- Work on screen orientation, proximity sensor etc. in Chromium
- Create a proper future-proof runtime for Tizen 3.0 (IVI etc.)
- Provide a fast web runtime for Android 4.0+, incl. on IA and ARM

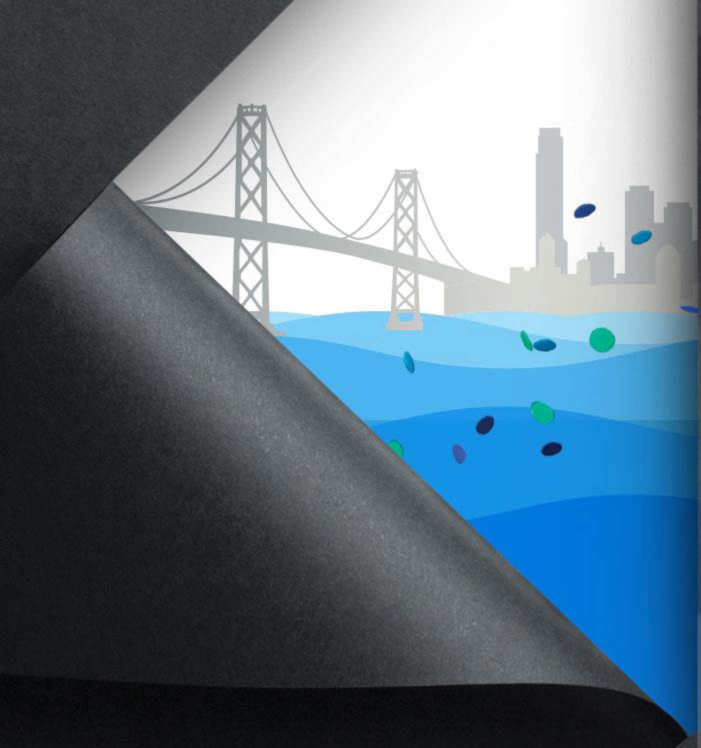
Introducing

CROSSWALK

- **A runtime for web apps, based on Chromium/Blink**
- **A real runtime (system wide) for all Tizen profiles**
- **We work upstream first, which benefits Tizen, Android and Chrome**
- **Packaged apps with permissions (WGT, XPK)**
- **Also works as Embedded runtime for Android**
    - Distribute Crosswalk with the app on 4.0+
    - Get Chrome performance and features, plus Crosswalk value-add

Crosswalk focus areas

- **Emerging standards**
  - Manifest, Bluetooth, Raw Sockets, Orientation Lock
- **Access experimental APIs from day one (via extensions)**
- **Extension system for the paltform and platform developers**
  - We develop most of our experimental APIs like extensions
  - Will support all current Tizen APIs
  - Hybrid app developers can extend the app with native capabilities
- **Intel adds special features, in parallel with standardization**
  - RealSense support (WIP), SIMD, etc



Future outlook

This is how we bridge the gap

Earlier you were introduced to some of the current failures on the web platform

Now let's look at how we expect to solve these issues!

The offline story – Service Worker to the rescue

- **An idea out of Google**
- **Inspired by Chrome Apps Event Pages**
- **Separately installed, upgradeable JS controller**
    - Small, have to act within short amount of time
- **It handles fetch (basically hooks into resource requests)**
- **Building blocks to easily create a cache**
    - Imagine Netflix working offline for pinned movies!

The offline story

Example 1

```
<script>
  // scope defaults to "/*"
  navigator.serviceWorker.register("/assets/v1/worker.js").then(
    function(serviceWorker) {
      console.log("success!");
      serviceWorker.postMessage("Howdy from your installing page.");
    },
    function(why) {
      console.error("Installing the worker failed!:", why);
    });
 </script>
```

The offline story

Example 2 and 3

```
// hosted at: /assets/v1/worker.js
this.version = 1;
var base = "http://videos.example.com";
var inventory = new URL("/services/inventory/data.json", base) + "";

this.addEventListener("install", function(e) { });

this.addEventListener("fetch", function(e) {
 var url = e.request.url;
 console.log(url);
 if (url == inventory) {
   e.respondWith(new Response({
     statusCode: 200,
     body: JSON.stringify({
       videos: { /* ... */ }
     })
   }));
 }
});
```

```
// caching.js
this.version = 1;
var base = "http://videos.example.com";

this.addEventListener("install", function(e) {
 // Create a cache of resources. Begins the process of fetching them.
 var shellResources = new Cache(
   base + "/assets/v1/base.css",
   base + "/assets/v1/logo.png",
   base + "/assets/v1/intro_video.webm",
 );

 // The coast is only clear when all the resources are ready.
 e.waitUntil(shellResources.ready());

 // Add Cache to the global so it can be used later during onfetch
 caches.set("shell-v1", shellResources);
});
```

Lack of control

- **Focus of the W3C SysApps + DAP working groups**
  - DAP has released many mini specs: Battery, Proximity, Ambient Light
  - Intel is actively implementing and pushing those specs in Chromium
  - SysApps started out too ambitious; Manifest moved to WebApps
- **At Intel we are pushing hard in W3C and Chromium upstream**
  - Intel and Samsung pushing Screen Orientation Lock
  - Now with support from Google/Android
- **Some APIs are being pushed in Tizen and Crosswalk**

Lack of sensitive APIs

- **Sensitive APIs are hard due to fingerprinting and private data and lack of web wide permission model**
  - File system access, screen info, Bluetooth, etc.
- **Slightly easier for packaged apps – we ask up front**
- **Intel was involved with current Tizen APIs and we are now working on better, more standard aligned APIs in W3C**
  - We push those though Crosswalk, while keeping current APIs
  - We try to stay aligned with new spec work, like Promises, Streams API
  - We know that this is a ***long term*** process

Slow API roll out

We some-what solved this in Crosswalk

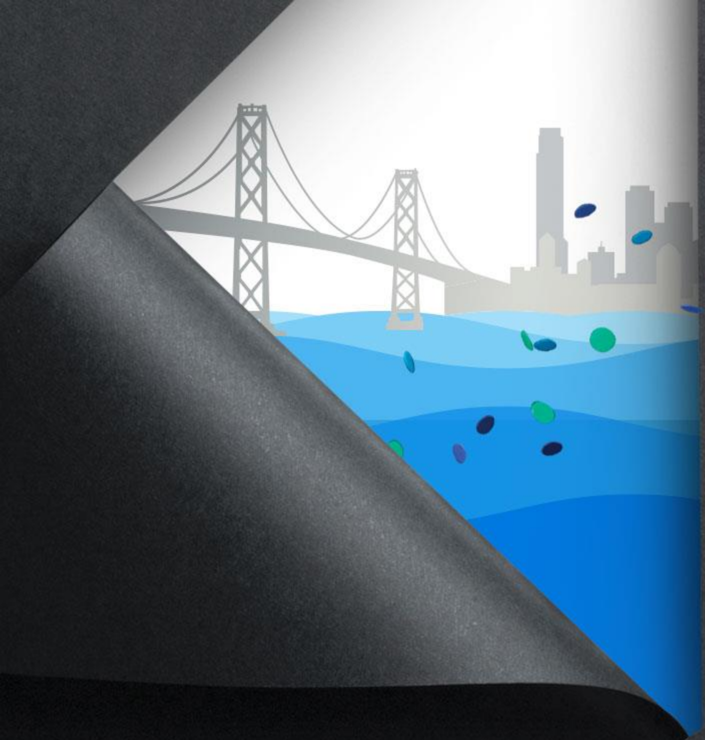
- **Enabling and shipping experimental APIs**
- **Keep most experimental APIs as separate extensions**
    - Ship only what you need (TV APIs on TV only)
    - Ability to not ship legacy APIs on newer platforms/profiles
    - Tizen lacks a deprecation plan

Manifest.json - application meta data

Spec pushed by Intel and Mozilla: w3c.github.io/manifest

- **Initial release almost done**
- **Service Worker integration coming later**
- **Google implementing in Chrome**

Icons, additional security policies (CSP), title/name, start URL, default orientation, preferred display style and much more to come



Summing up

Summary

The web is entering a new era and focus is on web apps and building blocks making it easy and obvious to construct native like and performing apps

New constructs for getting close to native: new API, better offline story, but still lacking a permission model urgently

Polyfilling and a more layered platform will create more innovation on top and close the gap to quicker evolving native platforms

Summary

Tizen is on board

- We attempt to give developers what they want
- We aim at bridging the gap to native and make web native
- We prepare Tizen for the future web innovations
- We support open standards and open innovation

Questions ?

TIZEN™
DEVELOPER CONFERENCE
2014
SAN FRANCISCO