



# A mobile web app technology stack

---

+Boris Smus

@borismus

# Outline

---



State of mobile web

Design philosophy

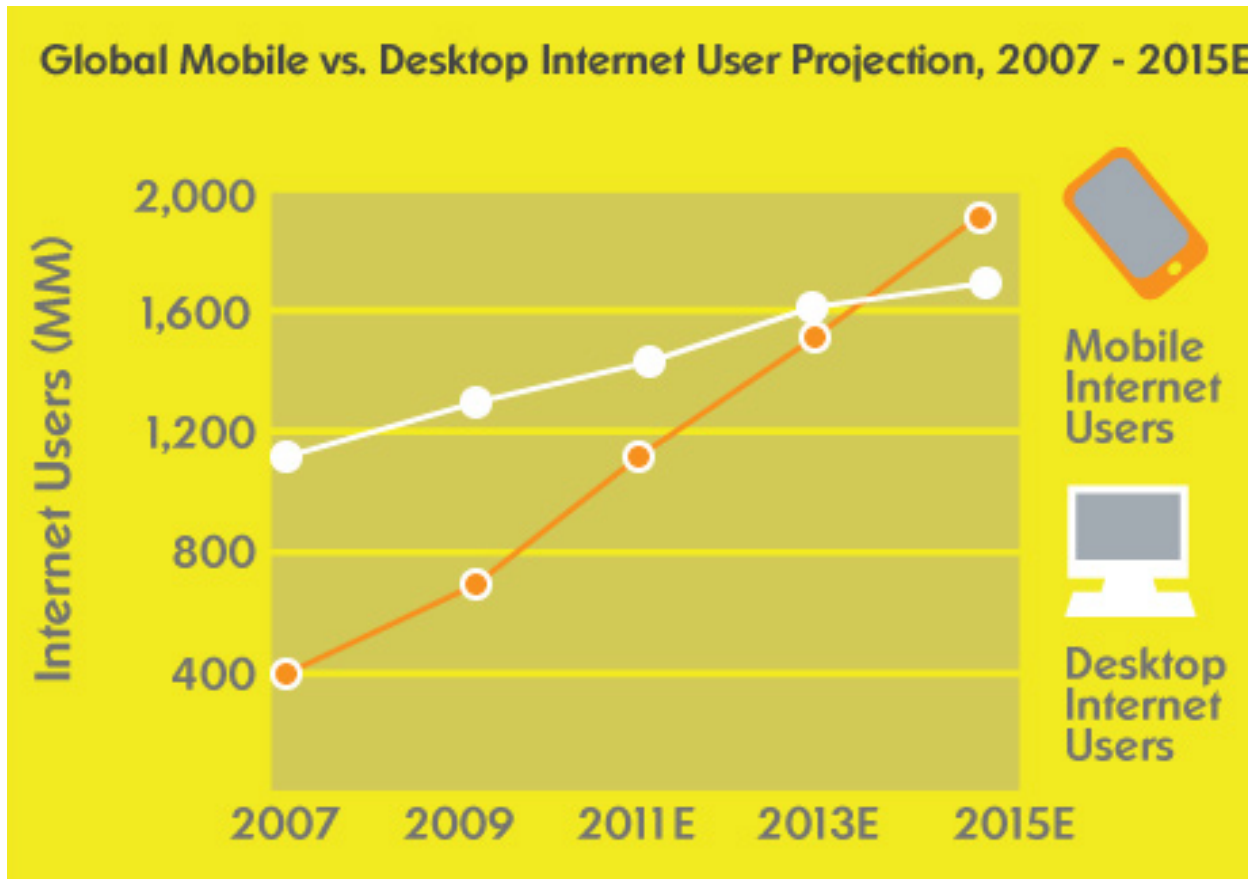
Building a mobile web application

Application demo

# Mobile trends



Mobile internet usage to overtake desktop by 2014



# Mobile browsers



Mobile is 95% WebKit

Missing features:

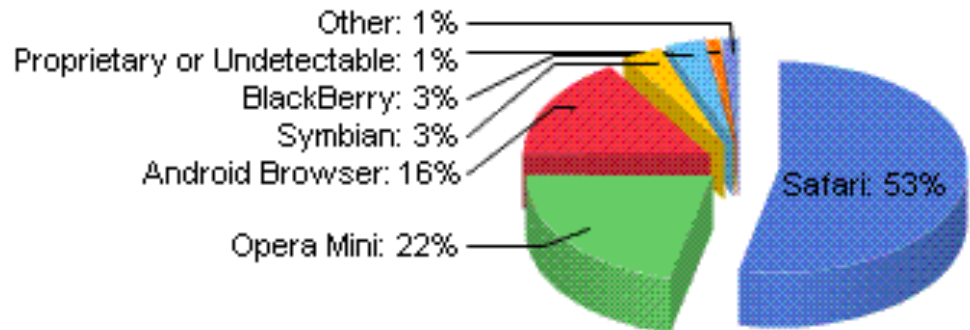
WebGL

Camera

Indexed DB

Web Audio

others



iOS 5 MobileSafari is great

Android 4 Browser improving

Opera Mini not a true browser

# Web vs native

---



Web as a unified platform - build for single target

Native is a moving target: iOS, Android, WP7, etc

Native pushes the boundary (closer to the metal)

Standardized web moves more slowly, but catching up!

# The case for enterprise

---



Largely data-driven applications!

Examples:

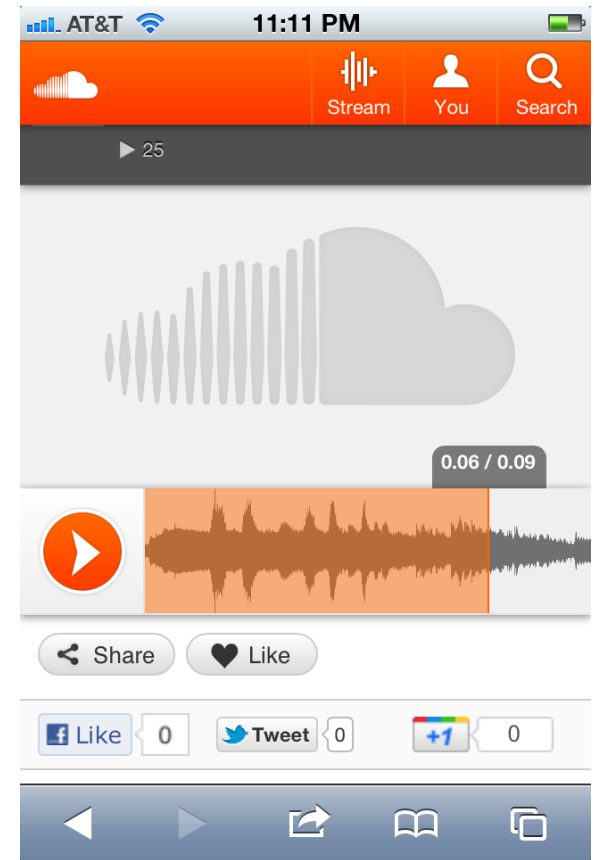
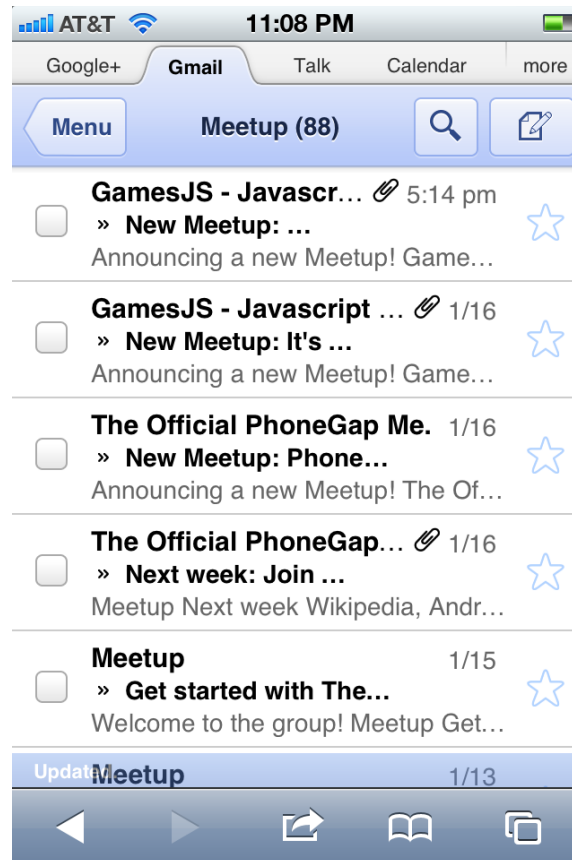
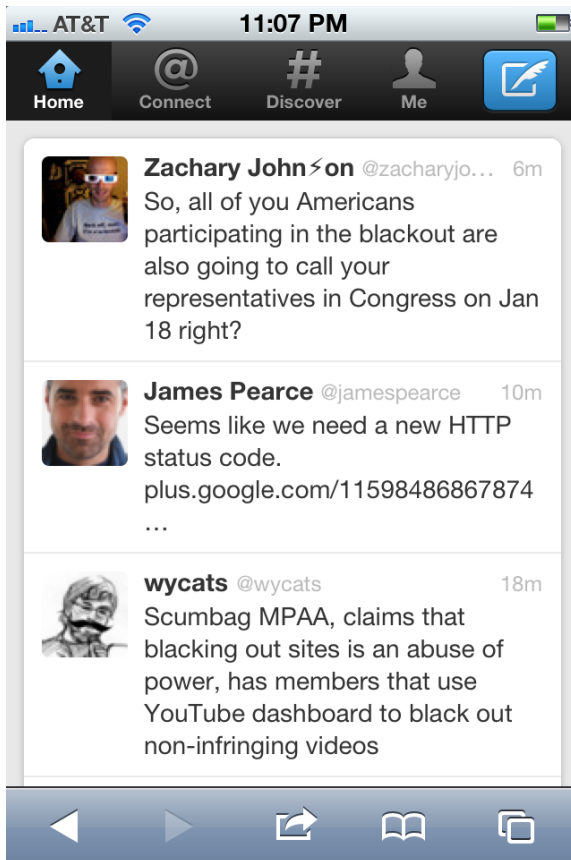
[app.ft.com](http://app.ft.com)

[mobile.twitter.com](http://mobile.twitter.com)

The mobile web is ready for apps.

Users may even [prefer web](#) to native  
[Some reports](#) claim that 87% actually do.

# Great examples



# Designing

---

Google

# Design is important

---



Think about it up-front!

Don't let "web" and "enterprise" be an excuse for poor UX

Be inspired by mobile patterns. Don't reinvent the wheel:

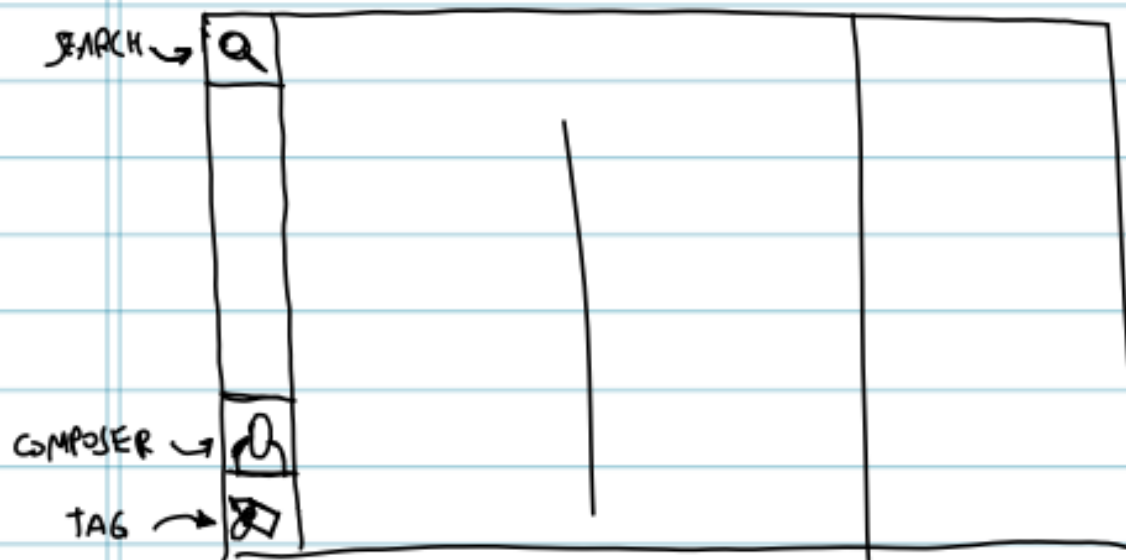
<http://ptrns.com/>

<http://mobile-patterns.com/>

# Low fidelity



(MANAGE MAKE — LANDSCAPE)



FINDER-STYLE UI FOR BROWSING.



UI POSITION BAR ON LEFT.

# Mid fidelity



Bach, Johann Sebastian	Creep
Chopin, Frederic	Electioneering
<b>Radiohead</b>	Lurgee
Santana	Karma Police
	Kid A
	No Surprises
	Nothing Touches Me
	<b>Paranoid Android</b>
	Stop Whispering
	Thinking about You



Paranoid Android  
by  
Radiohead

Chords:

```
A#sus5  A7Sus4  Dm7  Dm*  
-0-    -0-    -1-  -0-  
-3-    -3-    -3-  -3-  
-3-    -0-    -2-  -2-  
-2-    -2-    -0-  -3-  
-0-    -0-    -3-  -0-  
-0-    -0-    -0-  -0-
```

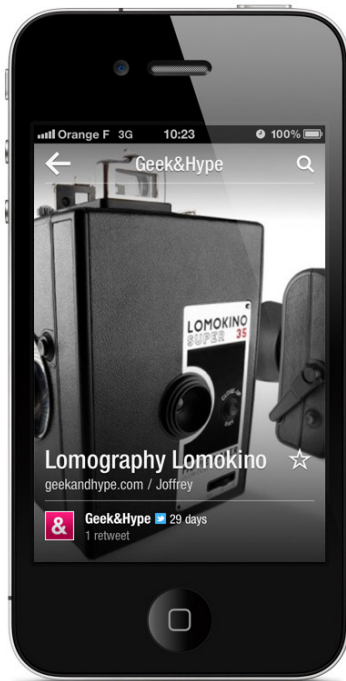
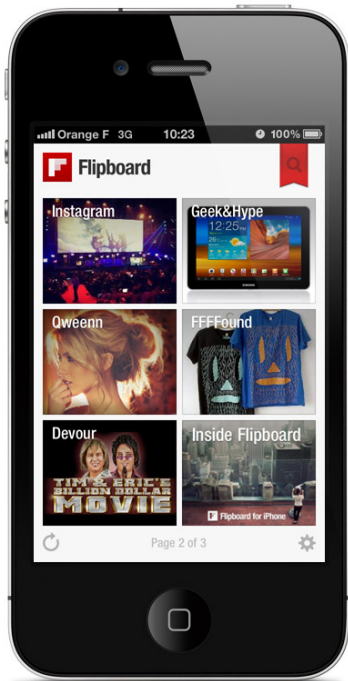
Intro:

```
Cm  Eb  F  Eb  Gm  Bb  
A#sus5  A7Sus4  Gm  Bb  
A#sus5  A7Sus4
```

Verse:

```
Cm      Eb      F  
Eb      (Gm      Bb  
A#sus5  A7Sus4) x2  
Please could you stop  
the noise. I'm trying to
```

# Tablet != Phone != ...



# Philosophy: Adaptive Apps



---

Too hard to use one DOM tree for all form factors

**Adaptive apps – Custom views and templates for each form factor, shared model**

Use responsive design within the form factor (eg. same layout on iPad 4:3 and Galaxy Tablet 16:9)

# Building

---

Google

# Model view controller

---



Fundamental pattern for separating concerns

Model handles data and server persistence layer

View handles user input and rendering

Used to be on the server. Now moving to the thicker client.

# Paradox of choice

---



There are many MVC frameworks

[TodoMVC](#) - one app written in all of them

[Comparison blog post](#) - a high level comparison

I use Backbone.js for relatively simple apps

# Templating engines



Complex apps require complex DOM  
DOM manipulation is relatively slow

Answer: JS templating

1. **Embed** `<script id="my-template" type="text/my-template-language">` into HTML, with text contents of the template.

2. Use template library to populate template with data.

## Mustache is a logic-less templating engine

```
{{#items}}  
  {{#link}}<li><a href="{{url}}">{{name}}</a></li>{{/link}}  
{{/items}}
```

template

+

```
{  
  "items": [{"name": "green", "link": true, "url": "#Green"},  
            {"name": "blue", "link": true, "url": "#Blue"}],  
}
```

data

=

```
<li><a href="#Green">green</a></li>  
<li><a href="#Blue">blue</a></li>
```

Mustache.to\_html(template, data);

# CSS Frameworks



## Augmented CSS-style languages

- \$variables: true
- .nesting { .allowed { font-color: bold; }}
- mixins/inheritance

Many variants of syntax, but basically the same.

My preference: SCSS

# App view layout



Best practice: avoid tables, relative positioning, absolute positioning, floats

Use [CSS flex-box](#)!

```
#flexbox {  
  display: box;  
  box-orient: horizontal;  
}  
#flexbox > p:nth-child(2),  
#flexbox > p:nth-child(3) {  
  box-flex: 1;  
}
```



Caveat: new API just landed, but relatively few changes

# More layout



What about headers/footers? Use `position: fixed;`

How to scroll inside elements? Use `overflow: scroll;`

Inertial scrolling? (Note: iOS 5 only)

`-webkit-overflow-scrolling: touch;`

# Touch input

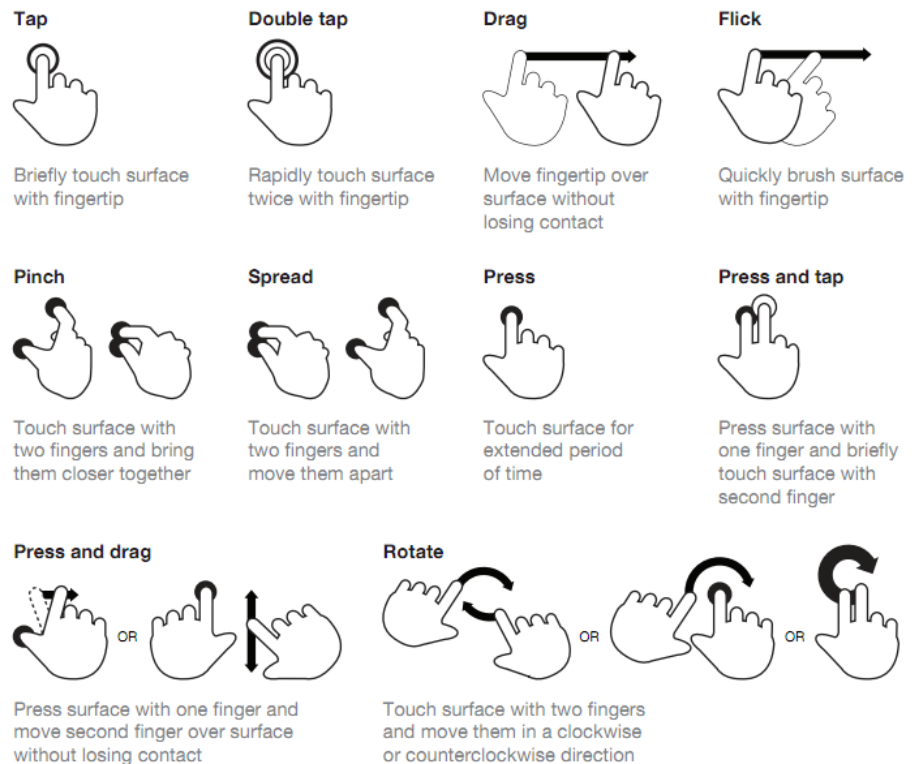


Fingers != mouse. Design for multi-touch!

touchstart  
touchmove  
touchend

Click delayed by 300ms  
on mobile. Use [fast click](#).

More info, [read article](#)



# Make it work offline



Storing assets: `AppCache`, *Filesystem*

Storing data: `localStorage`, `WebSQL`, *IndexedDB*

Incomplete support in mobile.

**Offline first** – pretend that there's no internet connection, implement a sync layer that works only when online.

[Offline/online events:](#)

```
navigator.onLine & window.(ononline|onoffline)
```

# Unit testing

---



MVC provides separation of concerns  
Views are hard to test, but

**Test your models!**

JUnit [start/stop mechanisms](#) for testing async code

# Tips and tricks

---



Enable Safari console for logging on iOS  
(*Settings/Safari/Developer*)

Simulate touch events on desktop with [MagicTouch.js](#)

Remote debugging hack with [jsconsole.com](#)

A similar tech stack, all packed up for you: [thorax.js](#)

# Demo

---

Thanks for your time!

Google