

web 2.0 EXPO

Presented by

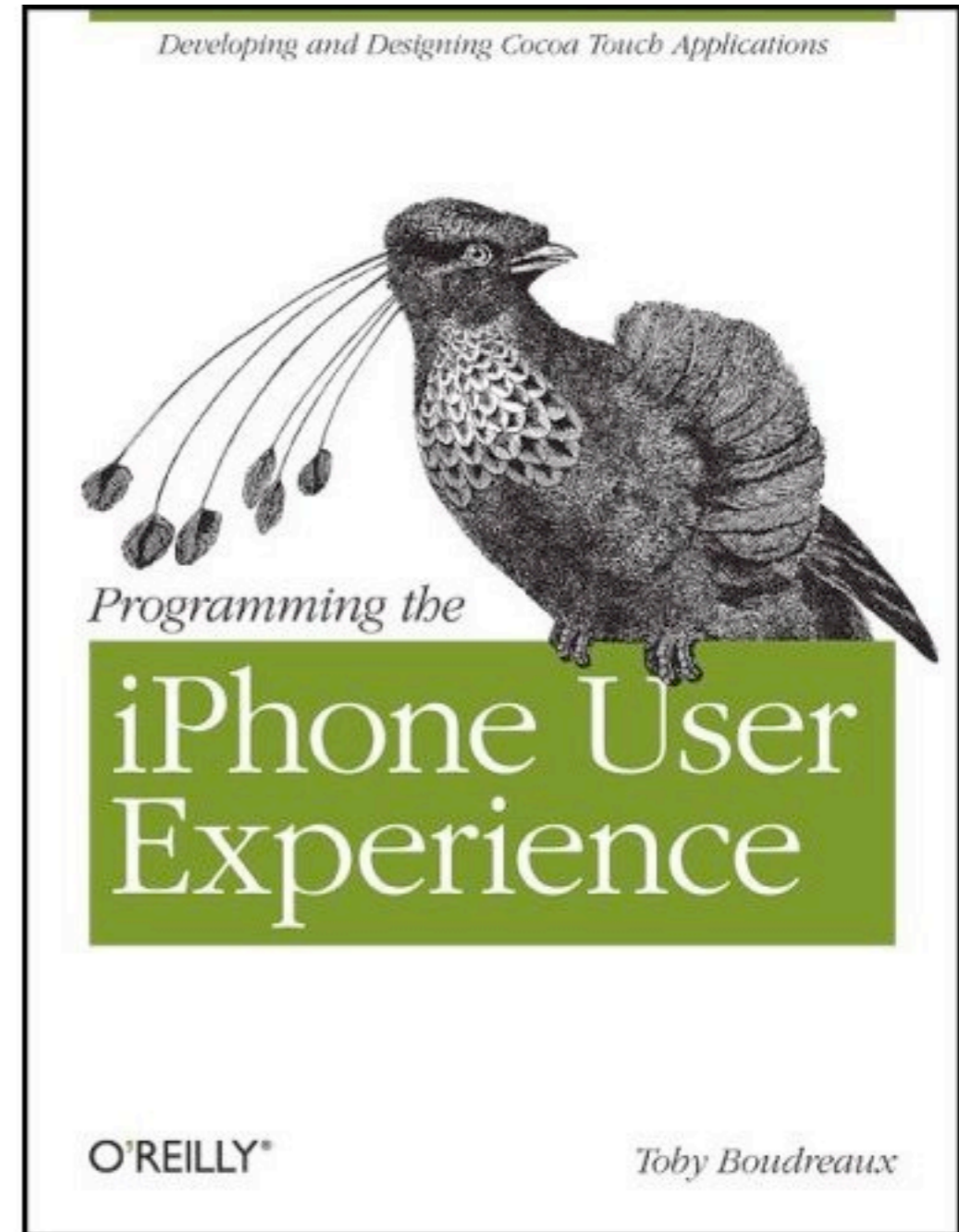


10 iPhone UX Anti-Patterns

Toby Joe Boudreaux
CTO, The Barbarian Group

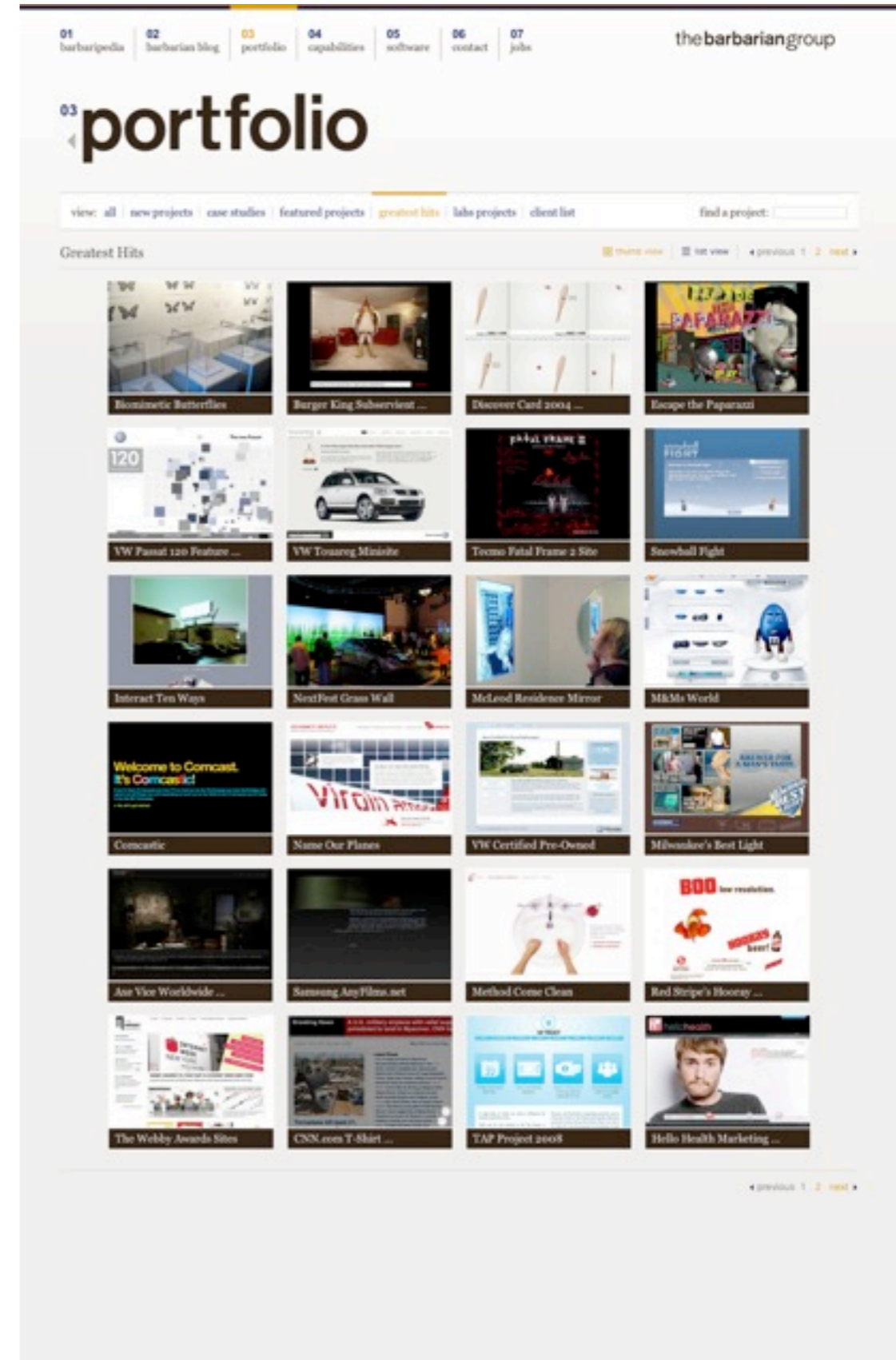
Hi. I'm Toby.

- ▶ Author of Programming the iPhone User Experience, for O'Reilly
- ▶ CTO at The Barbarian Group
- ▶ twitter: tobyjoe
- ▶ tobyjoe.com



The Barbarian Group

- ▶ Full-service digital/interactive company and consultancy
- ▶ The Subservient Chicken to the new iTunes 8 music visualizer to CNN Shirts to Plainview



Today's Talk

- ▶ Terminology
- ▶ iPhone UX Assumptions
- ▶ 10 Anti-Patterns



What is a Design Pattern?

- ▶ Reusable solution to a common problem
- ▶ Extractions of proven solutions
- ▶ Collectively emergent

What is an Anti-Pattern?

- ▶ A design pattern that causes at least as many problems as it solves, despite better options existing

Anti-Patterns Are Not:

- ▶ Bugs
- ▶ Dumb features
- ▶ Crappy code
- ▶ Bad habits

What are UX Anti-Patterns?

- ▶ A user experience anti-pattern is measured by the impact on the experience
- ▶ More subjectively judged than technical or operational anti-patterns

Do Anti-Patterns Mean You Suck?

- ▶ No. They mean you created software on par with your peers, that you solved problems.

So, How Does It Happen?

- ▶ The simplest thing that can possibly work isn't always perfect
- ▶ Collectively, users and use cases mature
- ▶ Better solutions to a problem evolve

Some iPhone UX Assumptions

- ▶ With the iPhone, the platform is the experience
- ▶ Apps should feel Apple-like (the HIG is back!)
- ▶ Apps should perform a small, focused set of tasks very well
- ▶ Apps should be optimized for mobile users
- ▶ Differentiation should come in how the app solves its main problem
- ▶ Immersive 3D games are exempt

Apple-Like?

- ▶ The HIG sets the stage
- ▶ Apple leads by example, and developers follow suit
- ▶ Don't fear the HIG

Users Want the Apple Experience

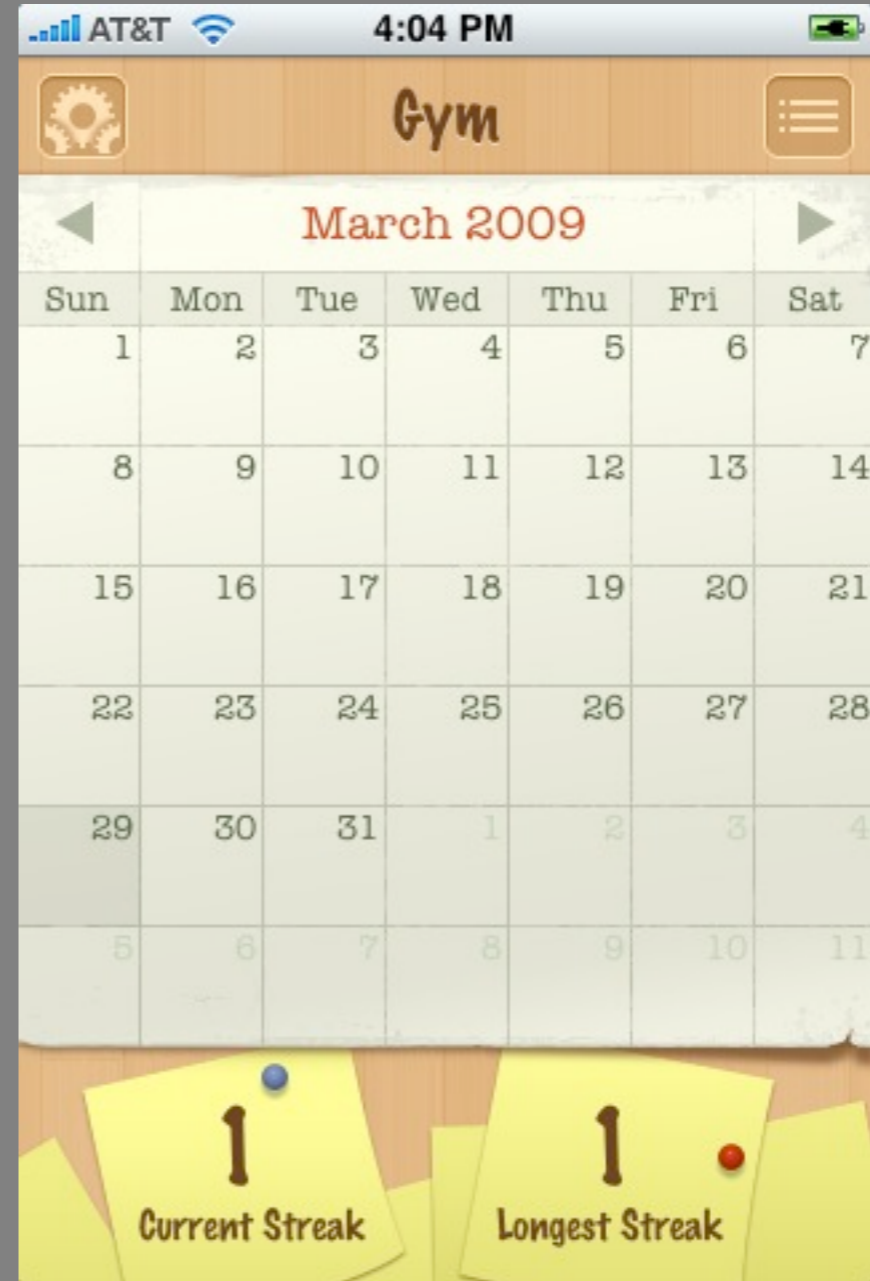
- ▶ You don't have to drink the Kool-Aid
- ▶ You have to serve the Kool-Aid

The Patterns

- ▶ Billboards
- ▶ Sleight of Hand
- ▶ App as OS
- ▶ Bullhorns
- ▶ The Bouncer
- ▶ The High Bar
- ▶ Memory Lapse
- ▶ Gesture Hijacking
- ▶ Spin Zone
- ▶ Sound-Off

Billboards

- ▶ Splash screens are evil, even when they're pretty



wurdle



loading...

1:59

0

new game



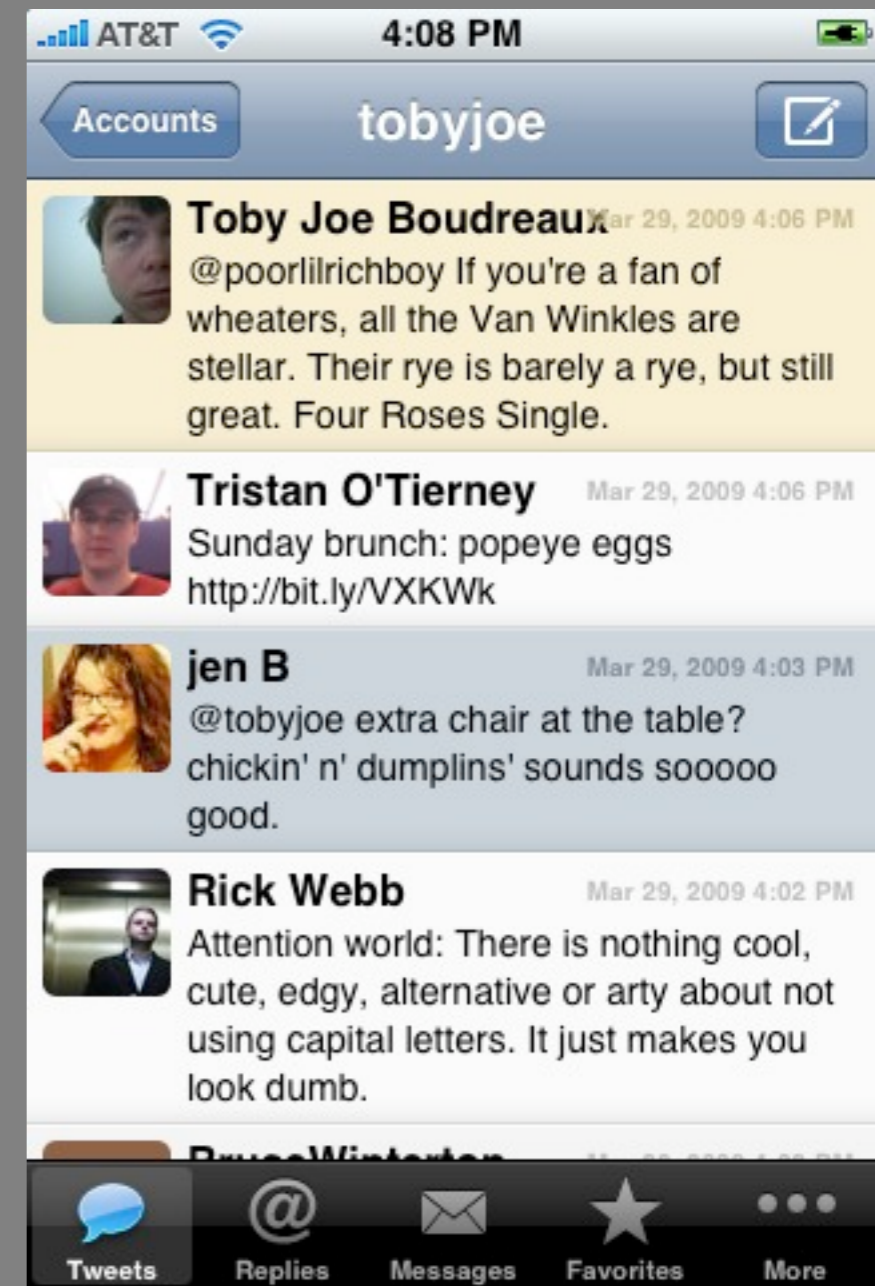
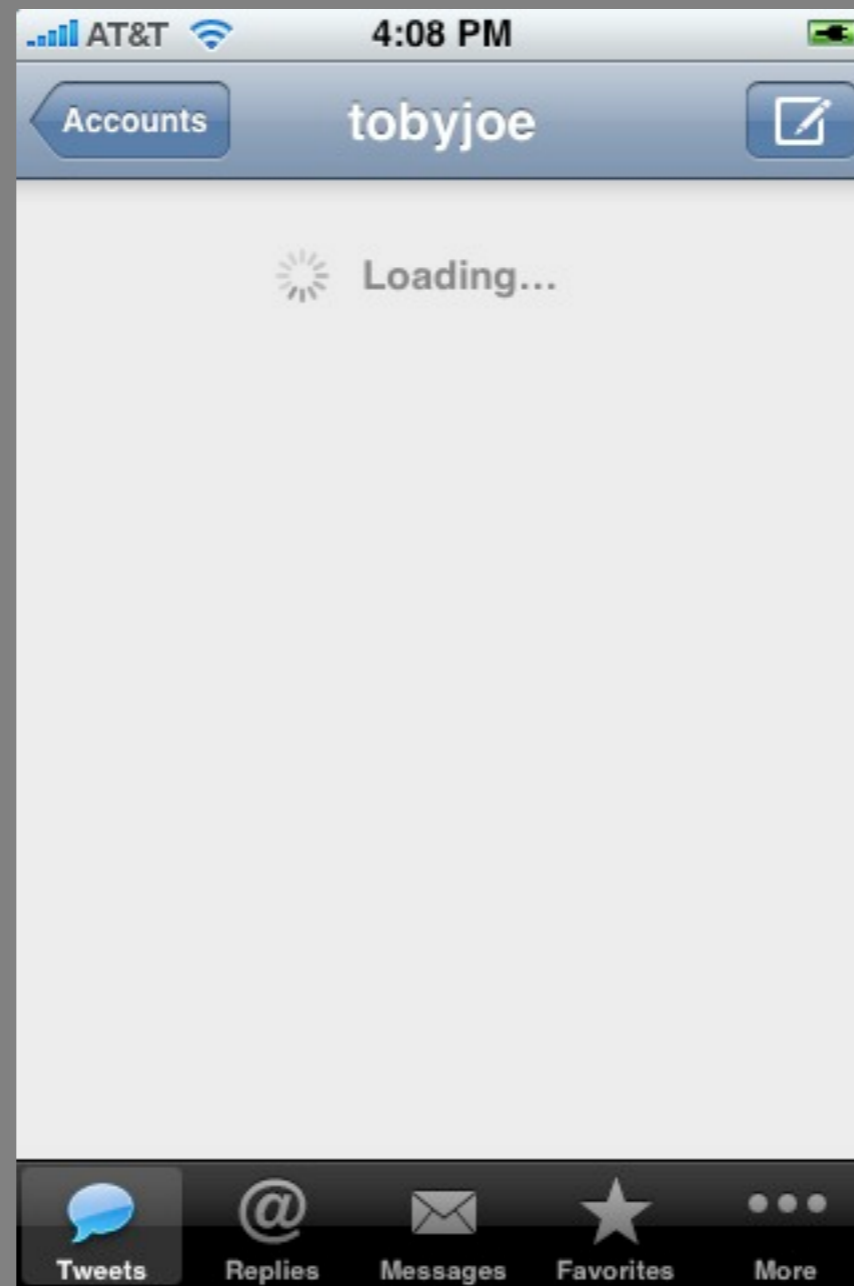
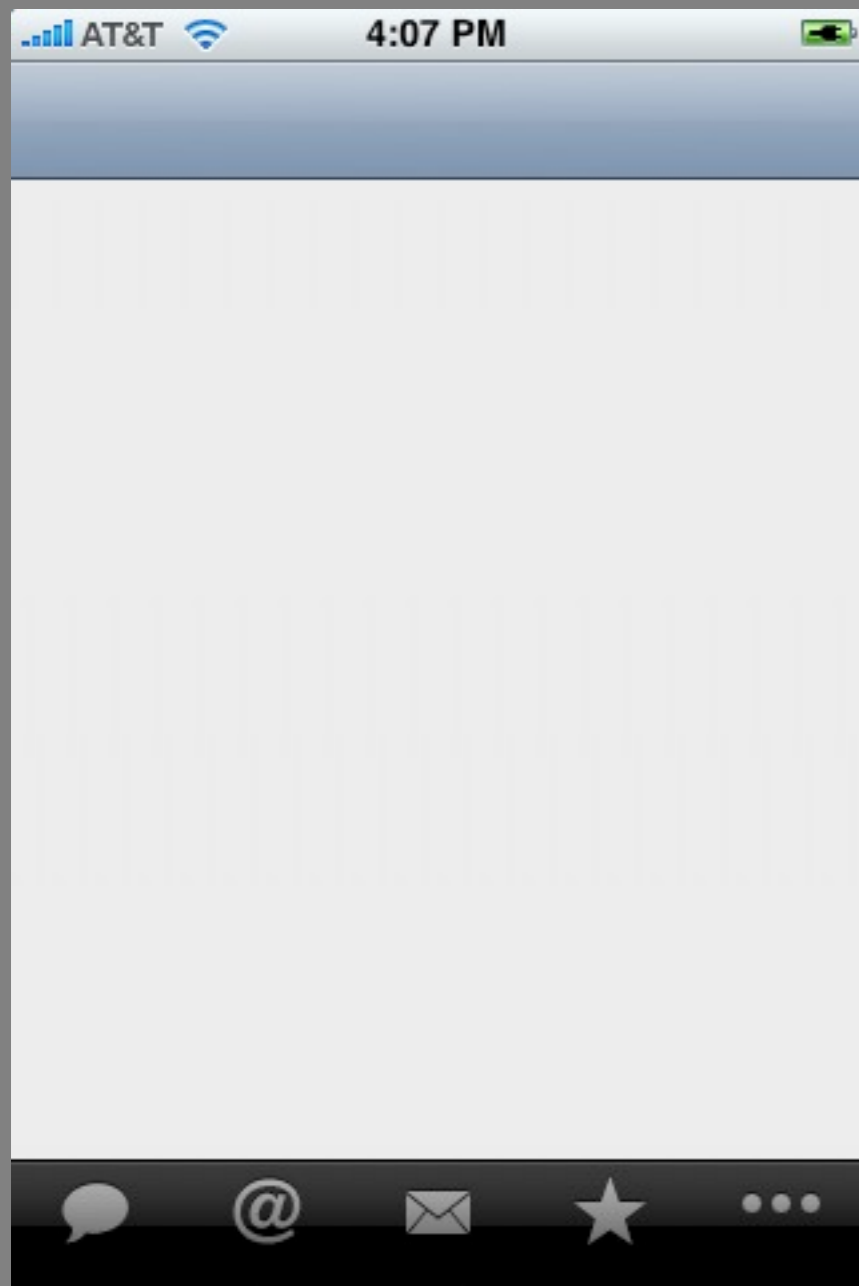
Billboards

“Avoid displaying an About window, a splash screen, or providing any other type of startup experience that prevents people from using your application immediately.” – **Mobile HIG**

Billboards

- ▶ Forget “quit” and “launch”
- ▶ Replace with “pause” and “unpause”
- ▶ Think about fast app cycling
- ▶ Don’t put branding ahead of users

The Progressive Reveal



Billboards

- ▶ Show a structured screen, minus user data
- ▶ Give the impression that your app unpauses instantly
- ▶ Make app cycling addictive

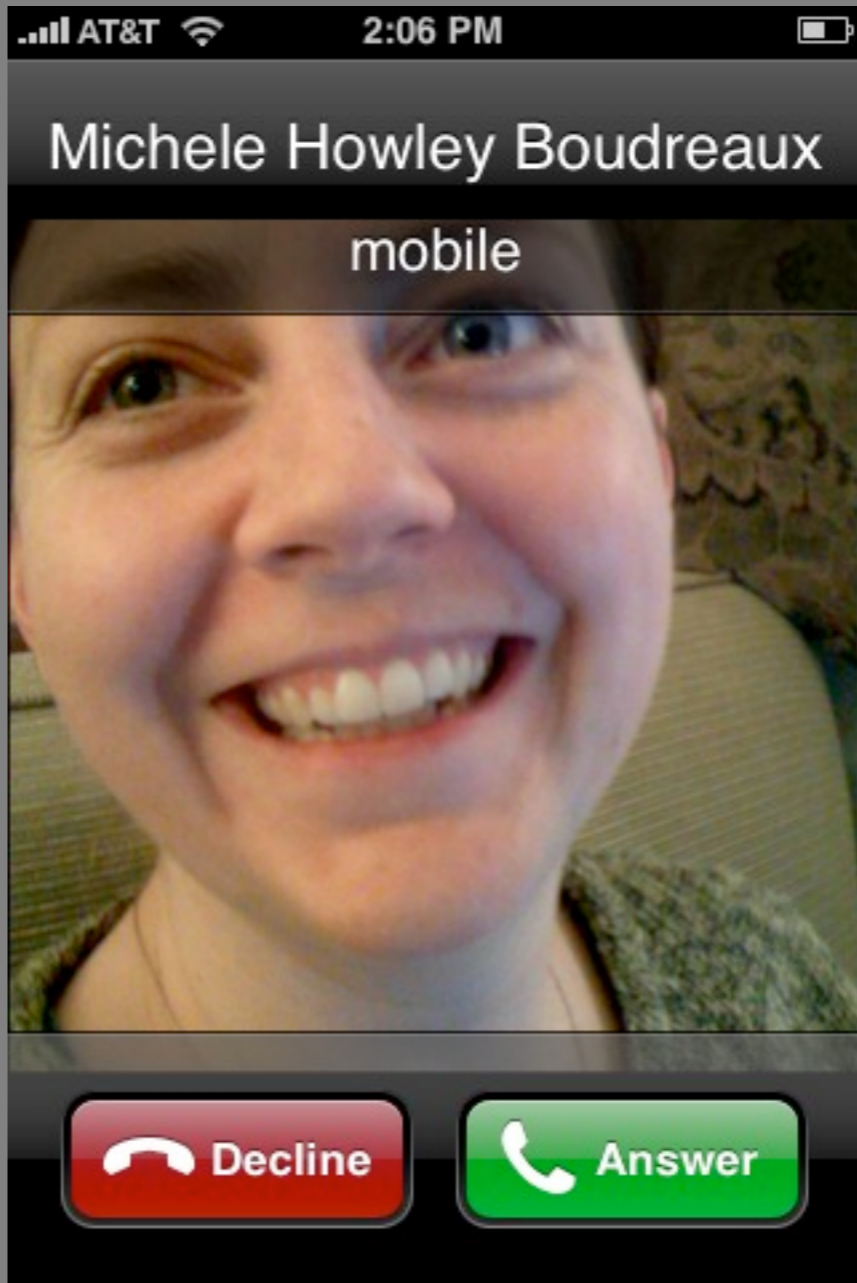
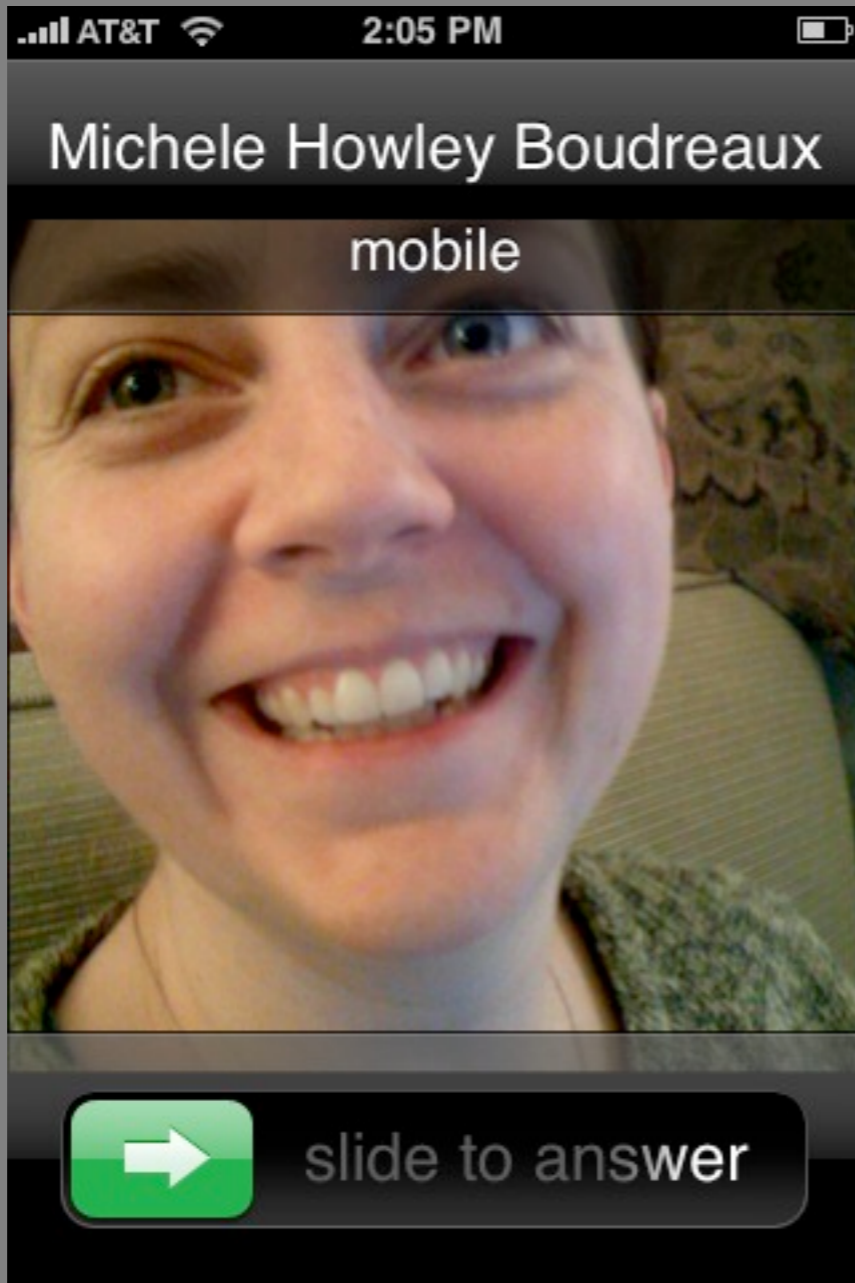
Sleight of Hand

- ▶ Swapping meaning for hot areas

Sleight of Hand

- ▶ A great example by Apple: the incoming call screen

Locked and unlocked



AT&T 2:05 PM

Michele Howley Boudreaux

mobile



Slide to answer

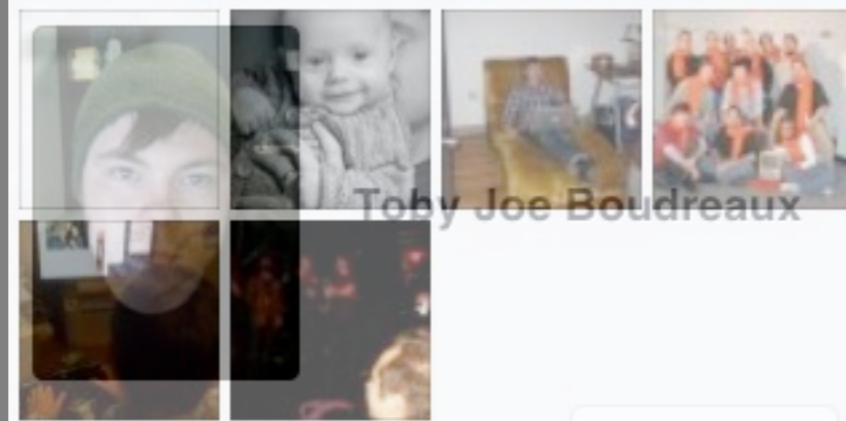
Sleight of Hand

- ▶ Navigation stacks make this issue tricky

Back buttons lead to extra taps



Toby Photos of Toby



Toby Joe Boudreaux

Wall Info Photos

Load More Photos...

Showing 6 of 7 Photos

Photos of Toby (7) >

Wall Photos (1)
3 months ago >

Mobile Uploads (23)
7 months ago >

Sleight of Hand

- ▶ Consider muscle memory and habit
- ▶ Overlay screens and consider proximity
- ▶ Account for one extra, accidental touch
- ▶ Always confirm potential accidents

App as OS

- ▶ Creating tarpits is a sticky habit

App as OS

- ▶ The device is effectively all screen
- ▶ One app at a time, and that app is full screen
- ▶ This creates the illusion of app as OS

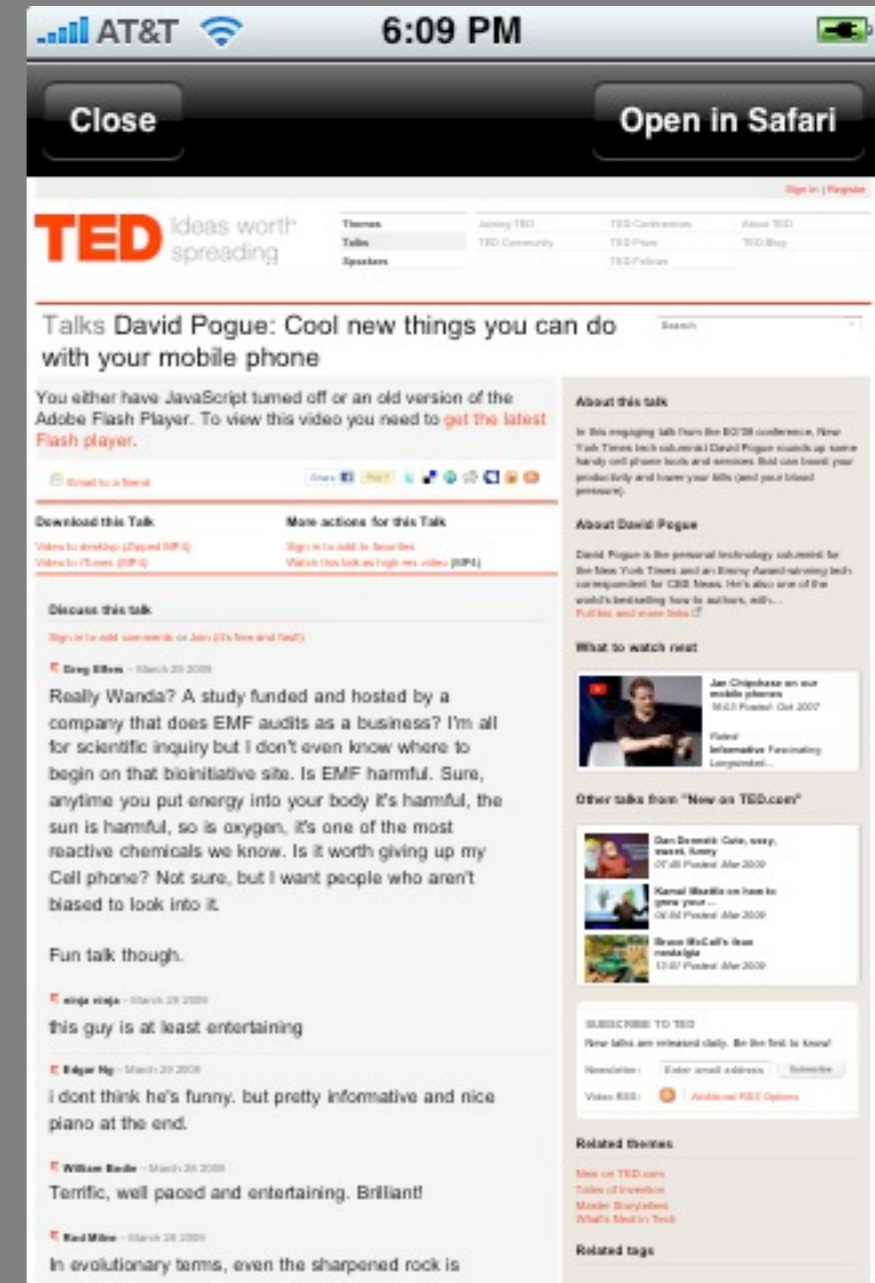
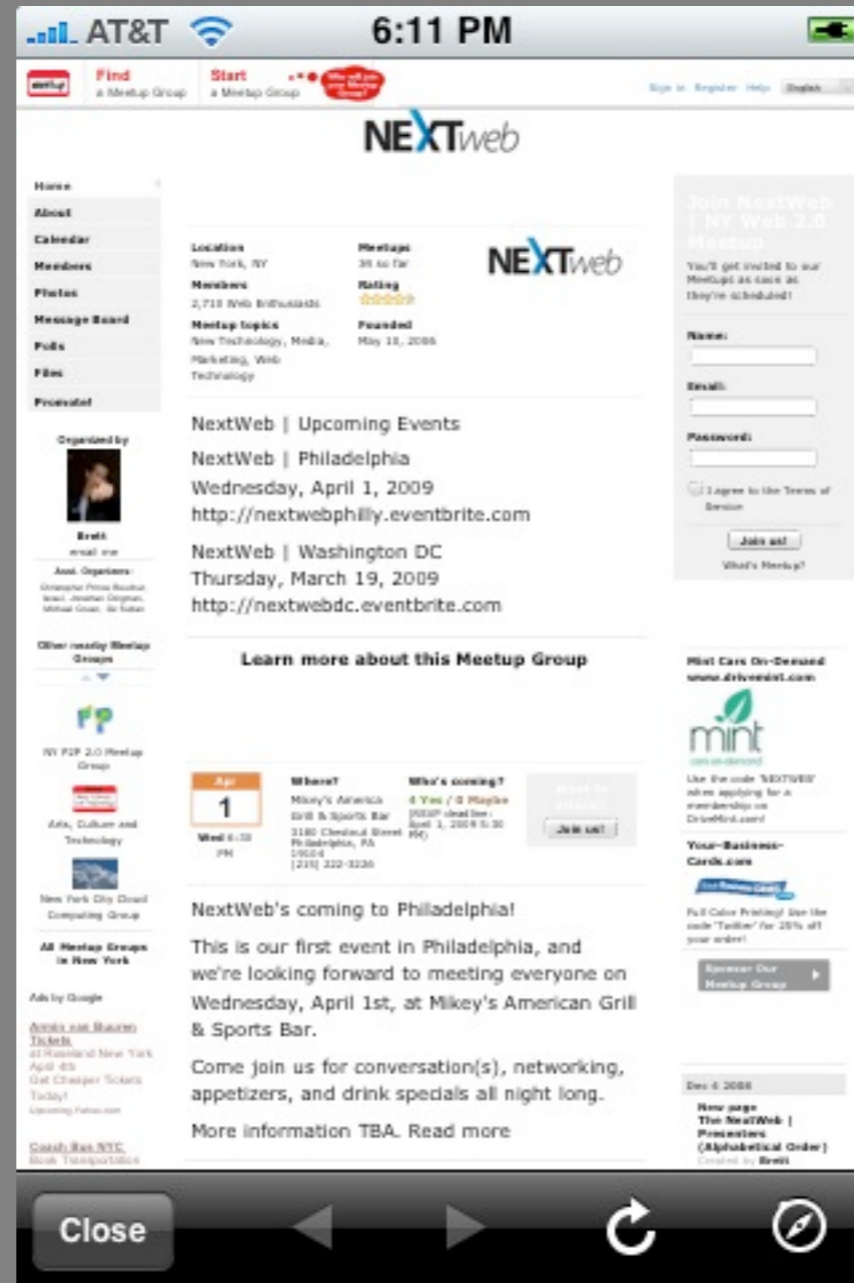
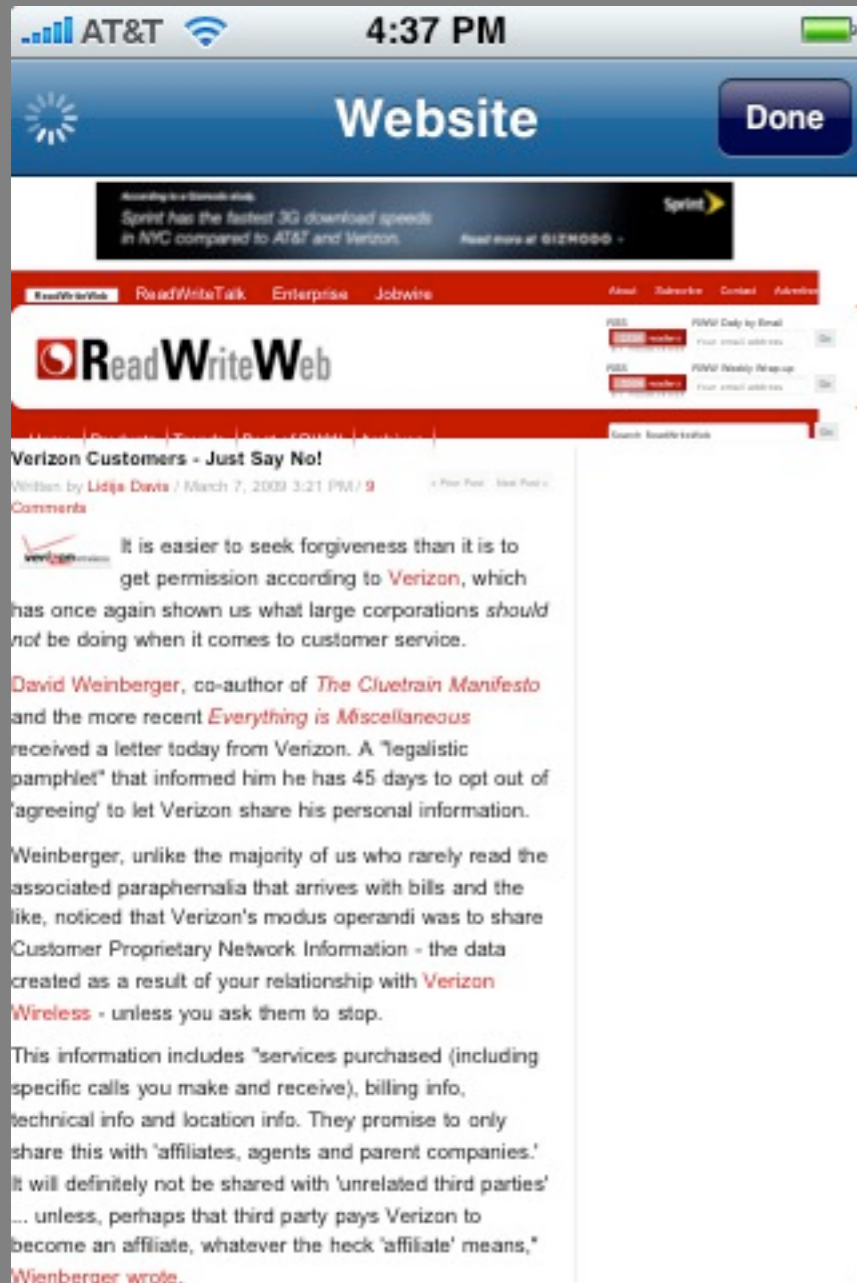
App as OS

- ▶ iPhone apps should be specialized, optimized, and should interoperate
- ▶ I call this **Cooperative Single-Tasking**

App as OS

- ▶ The App as OS anti-pattern leads to competing applications, not cooperative applications

Classic example: built-in browsers



App as OS

- ▶ Use custom URL schemes to interoperate
- ▶ Safari catches `http://`
- ▶ Mail catches `mailto://`
- ▶ YouTube catches `http://www.youtube.com/`
- ▶ Text catches `sms://`
- ▶ Twitterrific catches `twitterrific://`

App as OS

- ▶ Use shared views as they are added
- ▶ With 2.x use Photos and Contacts
- ▶ With 3.0 add Maps and Mail

App as OS

- ▶ Let Apple set the stage for interleaved functionality
- ▶ Anything not provided by Apple should be cooperative
- ▶ If you must compete, make it an option, with cooperation as the default

Bullhorns

- ▶ Notification mechanisms that are disproportional to the messages being communicated

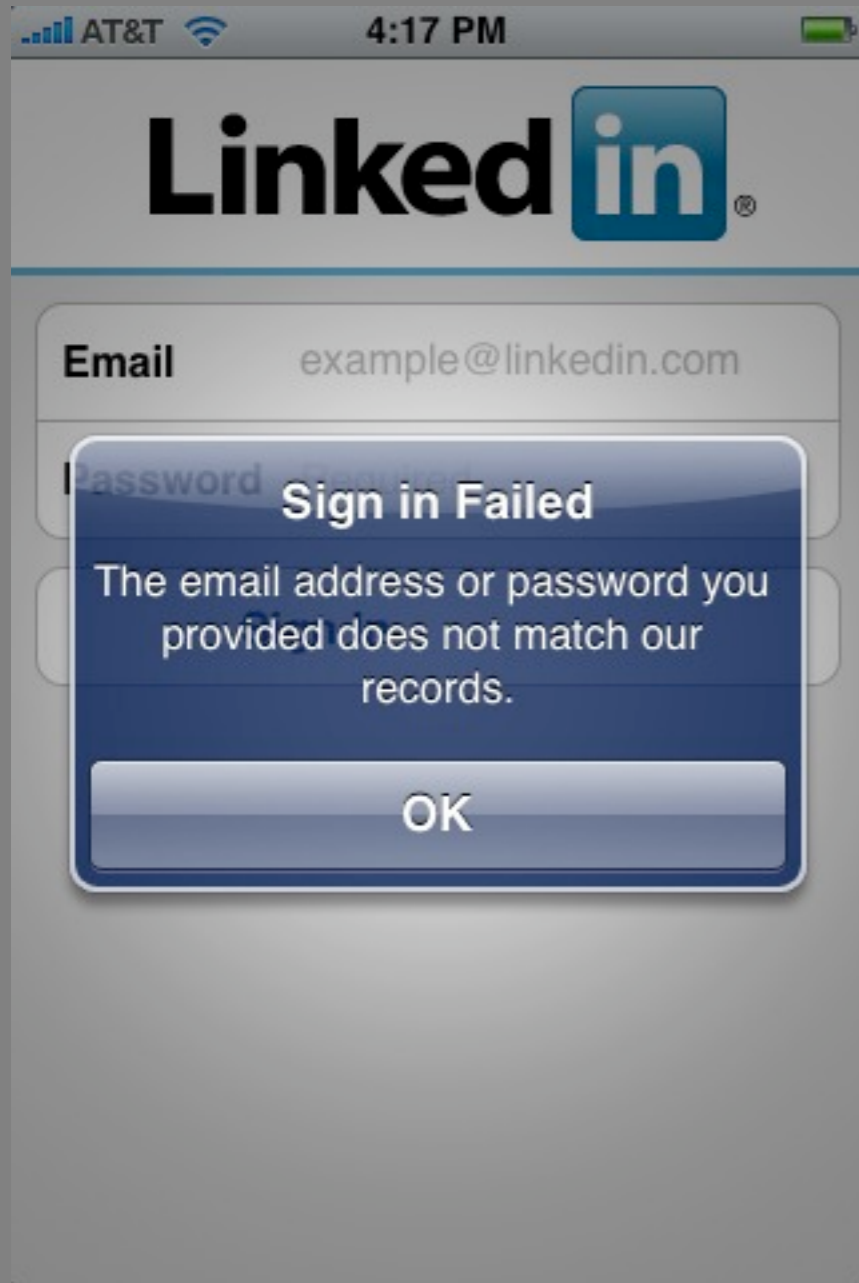
Bullhorns

- ▶ Example: Apple provides a very simple alert mechanism, called *UIAlert*

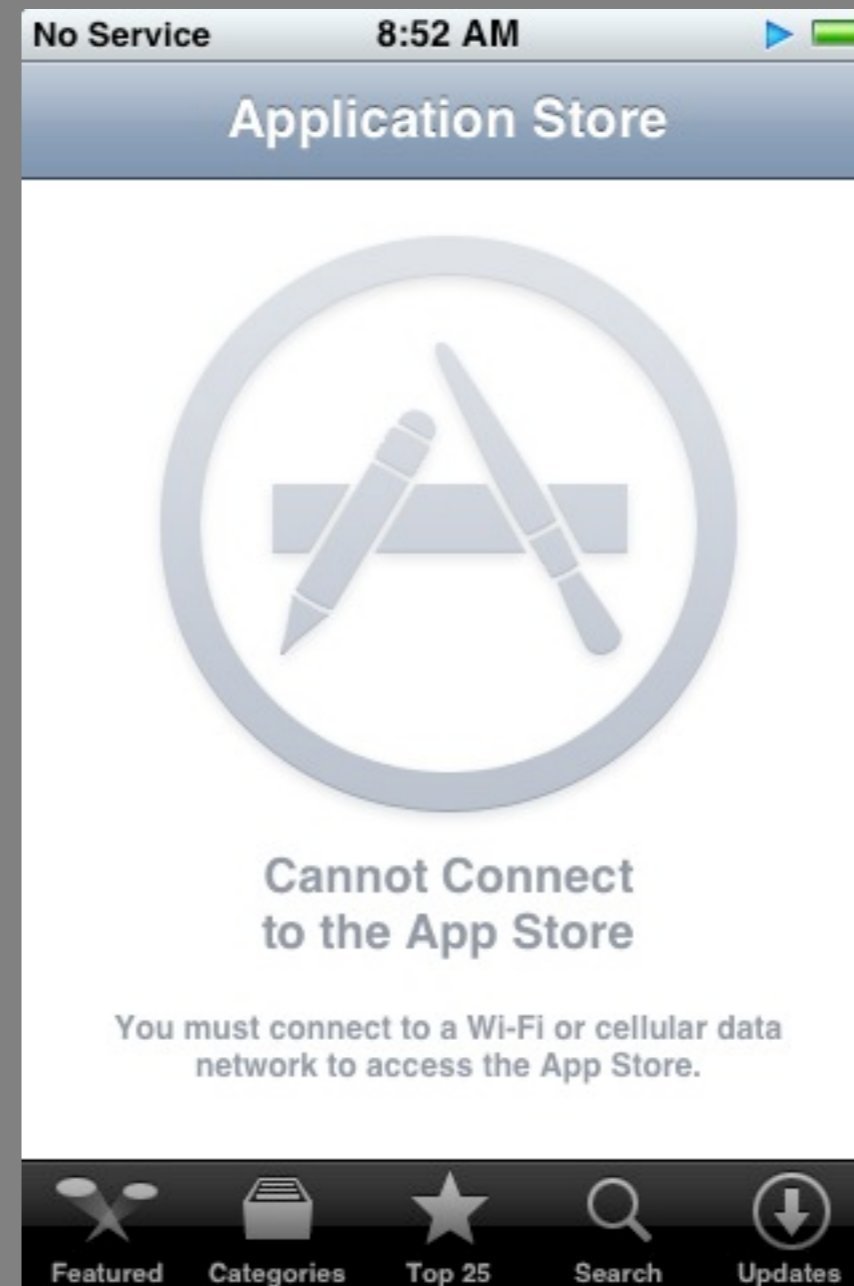
Bullhorns

- ▶ The simplest thing that could possibly work...
works

But it's wicked harsh



More appropriate



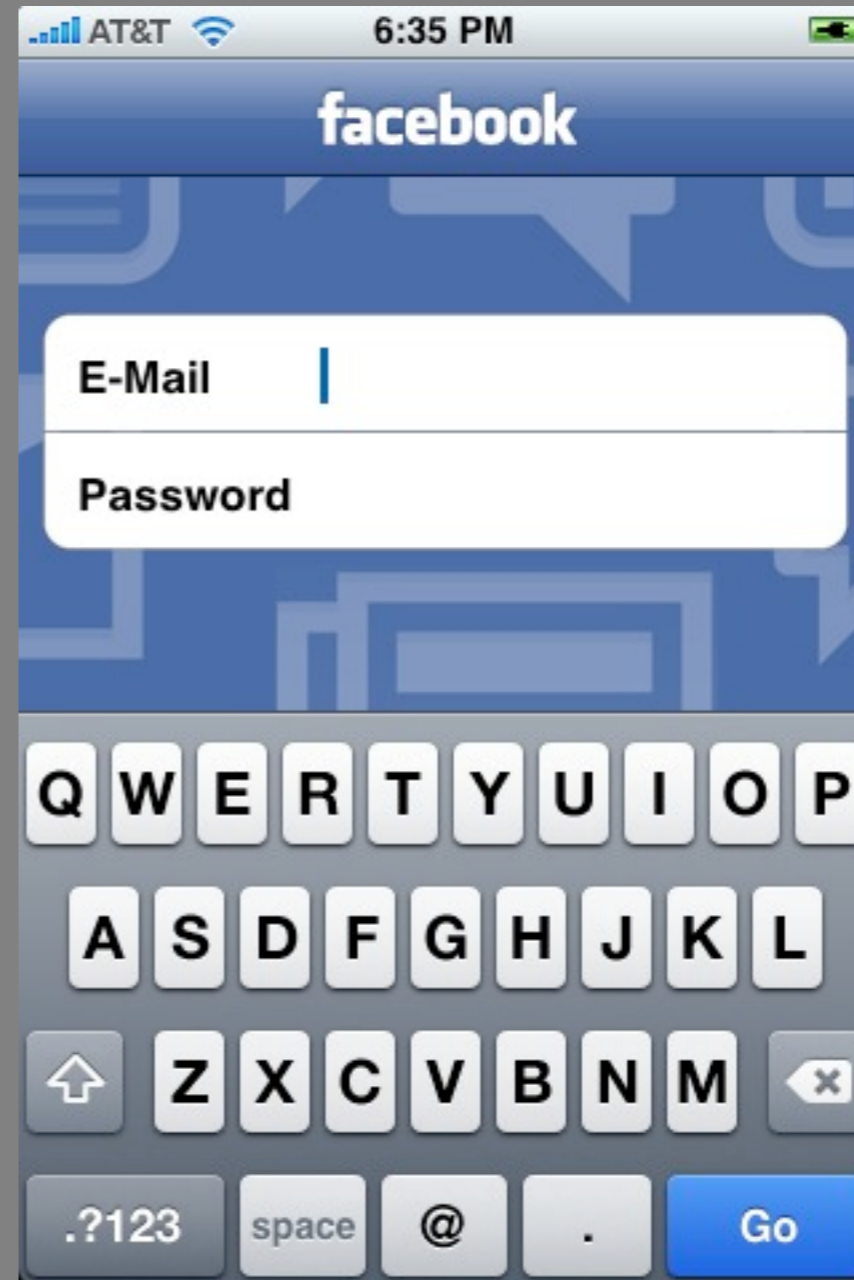
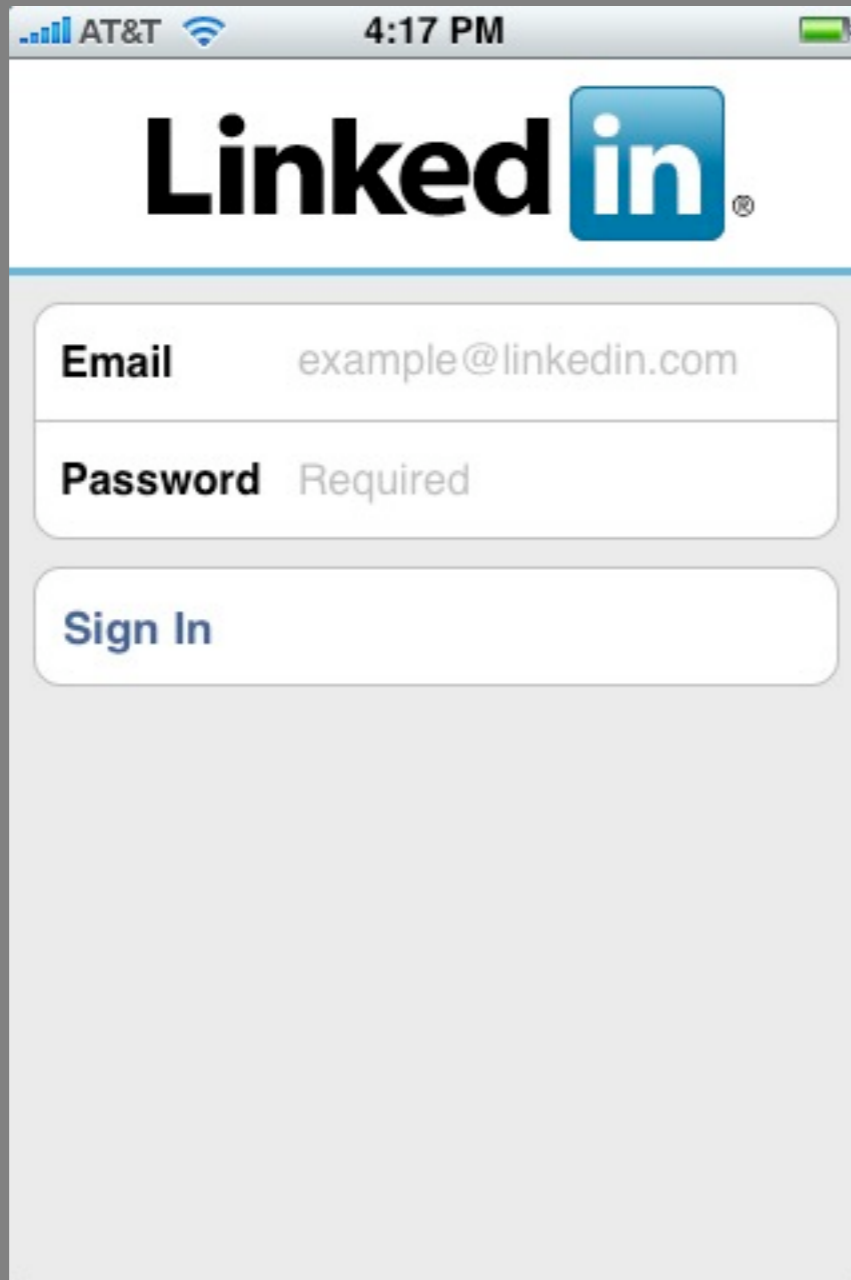
Bullhorns

- ▶ Keep the notification as passive as the situation merits

The Bouncer

- ▶ Providing value only for registered users

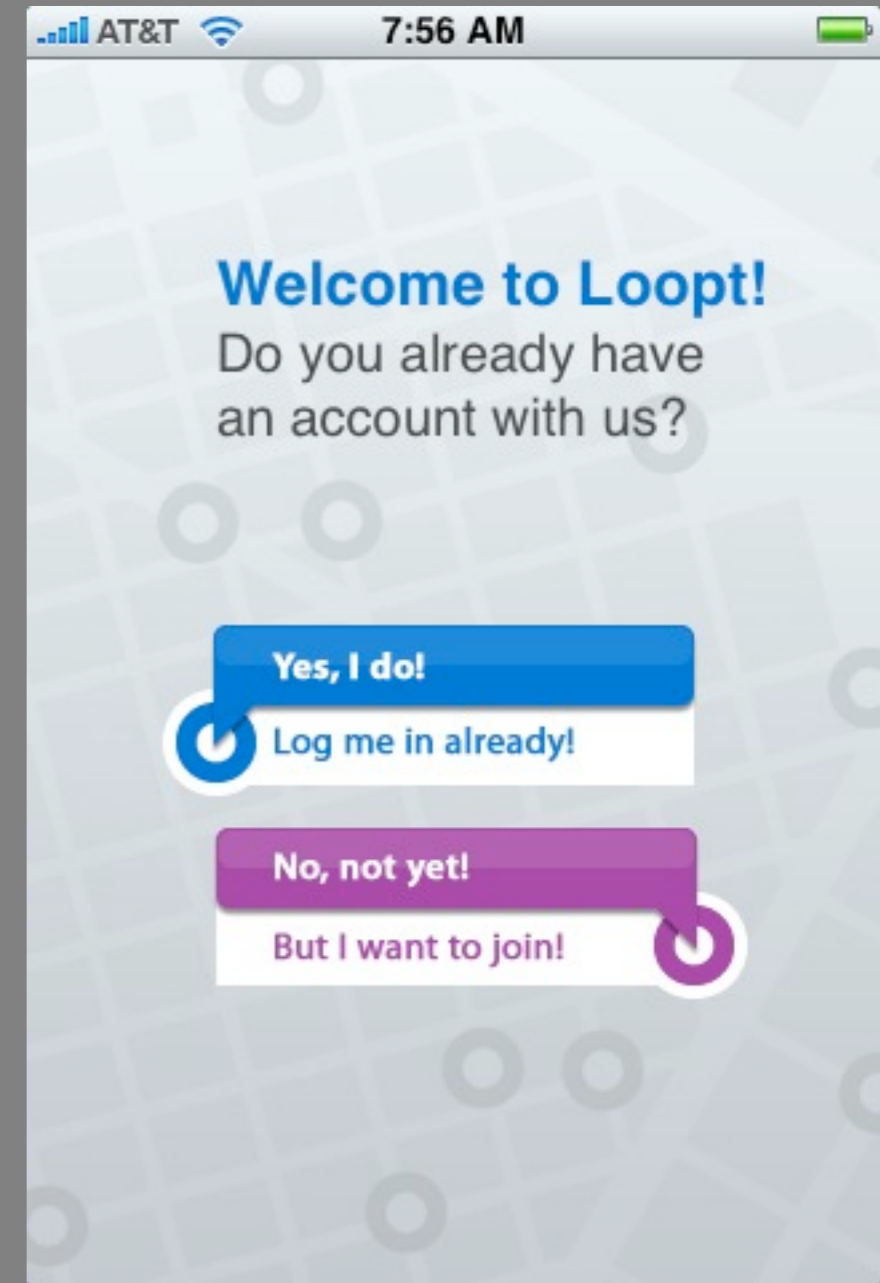
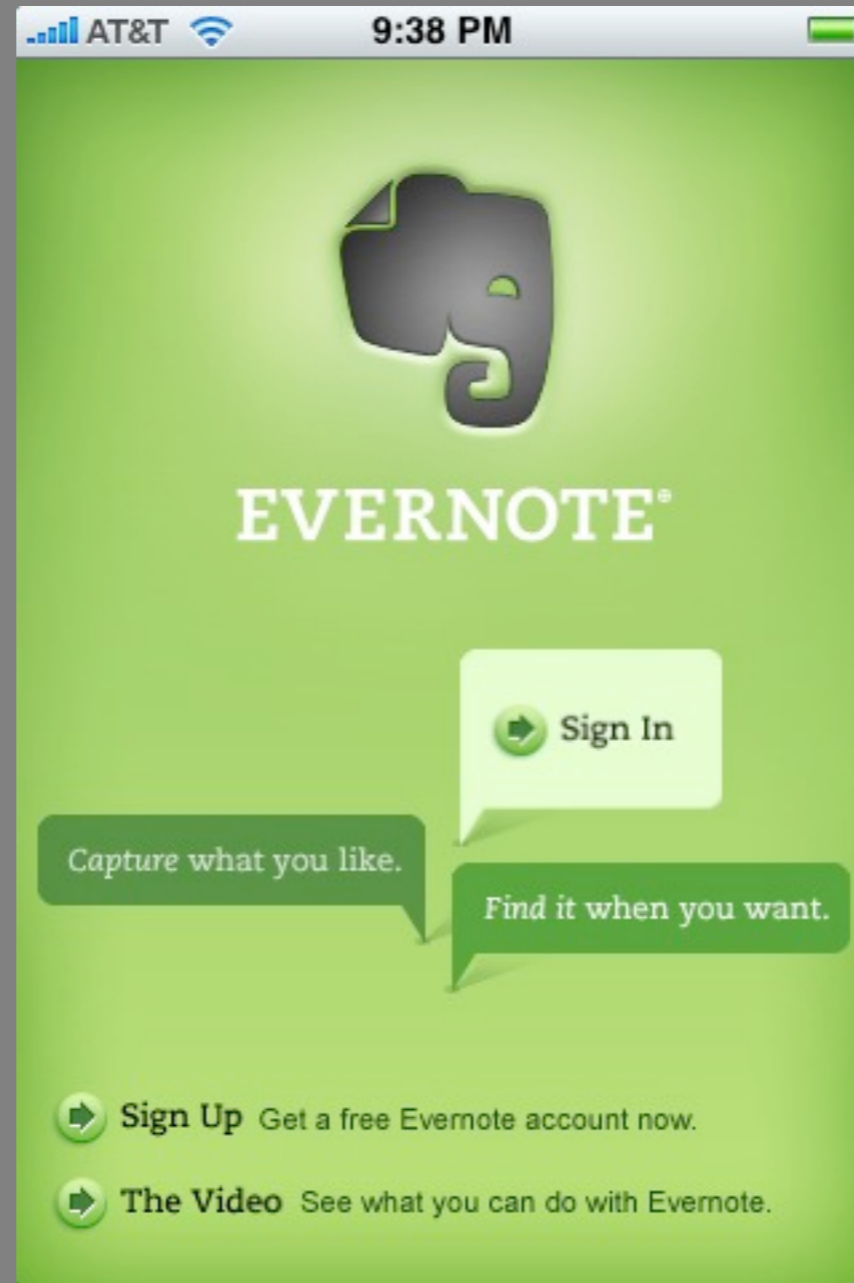
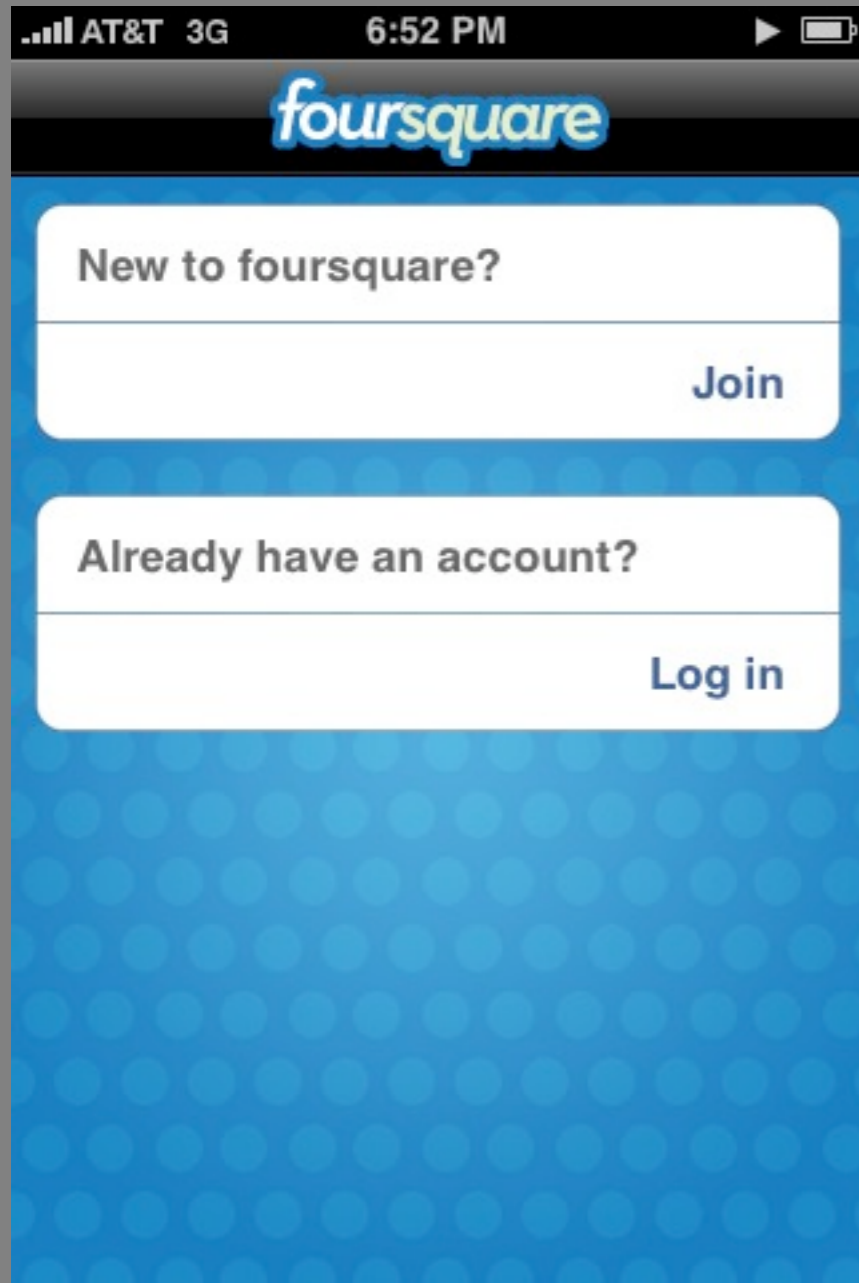
What now?



The Bouncer

- ▶ If possible, allow users to register from the app

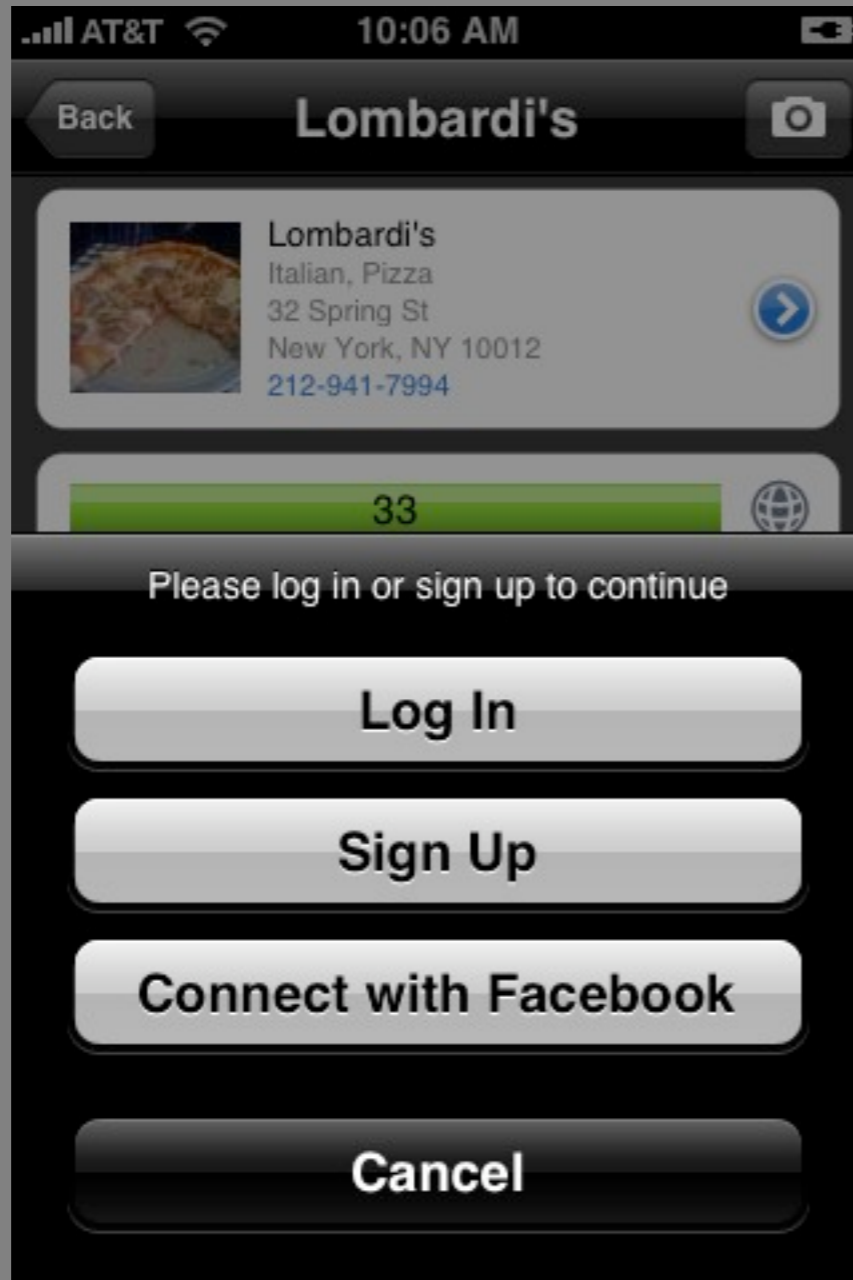
I signed up for all three. From the app.



The Bouncer

- ▶ If it isn't possible to register with the app, provide value and information

GoodFood options, WordPress info



The Bouncer

- ▶ Your mobile app can be an HTTP client – no different than a desktop Web browser
- ▶ Reward installs instead of penalizing or stonewalling

The High Bar

- ▶ Ignoring progressive enhancement

The High Bar

- ▶ Mobile users are unreliable
- ▶ Build for the lowest common (functional) denominator
- ▶ Add optional, intuitive enhancements

The High Bar

- ▶ Example: Don't **make** me shake my phone.
Let me shake my phone.

The High Bar

- ▶ Example: Don't **make** me share my location.
Let me share my location.

The High Bar

- ▶ Load data lazily
- ▶ Let users ask for more
- ▶ Assume spotty networks

The High Bar

- ▶ Accept one-handed use (stop giggling)

The High Bar

- ▶ **Pass the NYC Subway test**

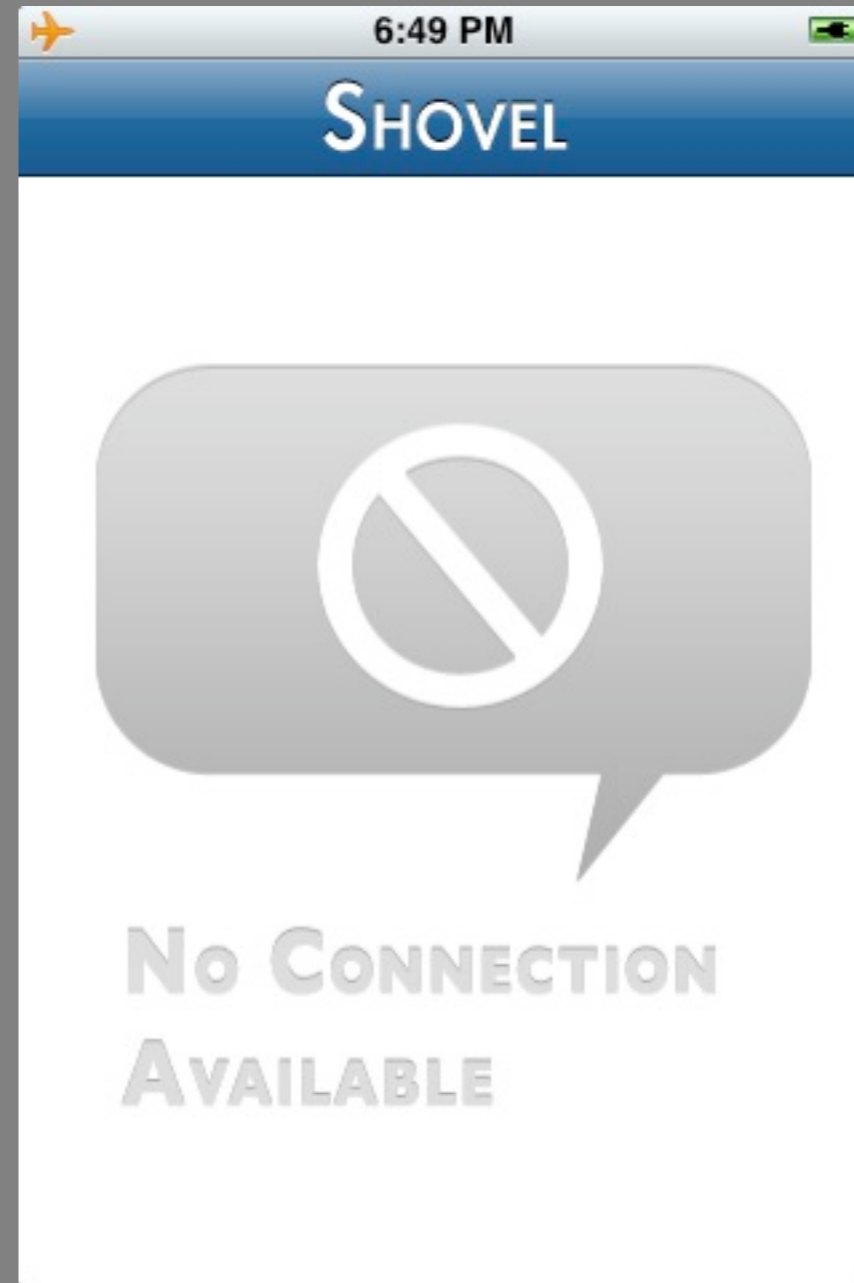
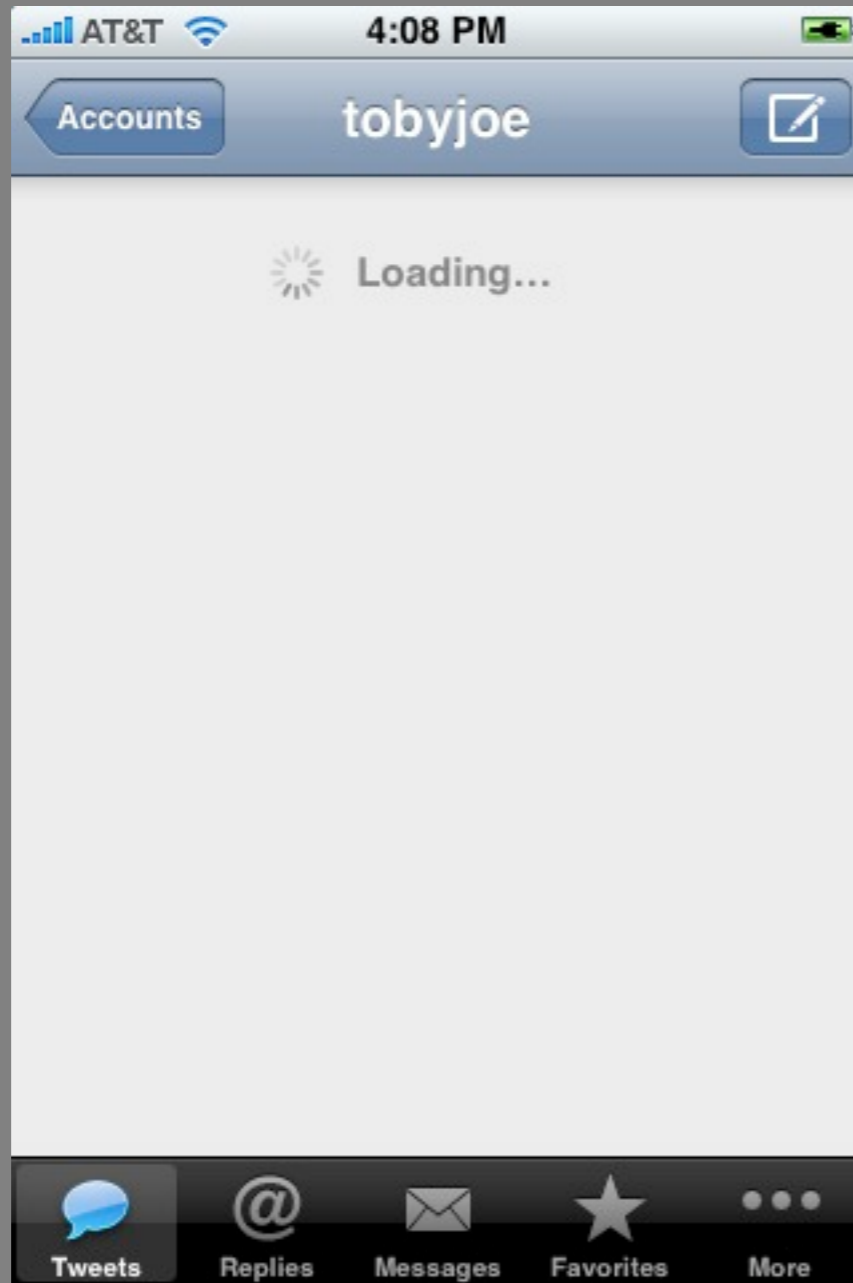
Memory Lapse

- ▶ Failing to persist content across pauses/launches

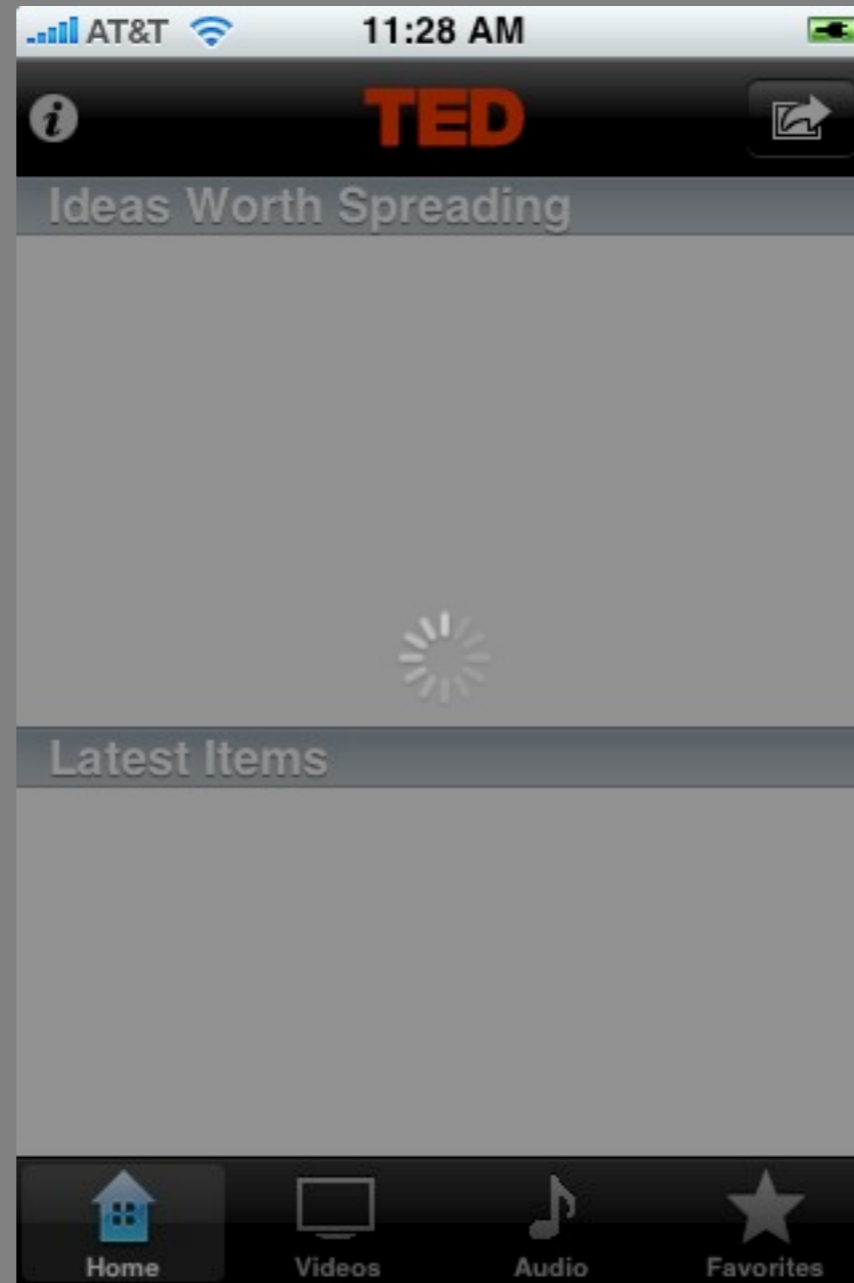
Memory Lapse

- ▶ The illusion of fast task-switching, pausing and unpausing, requires *state persistence*

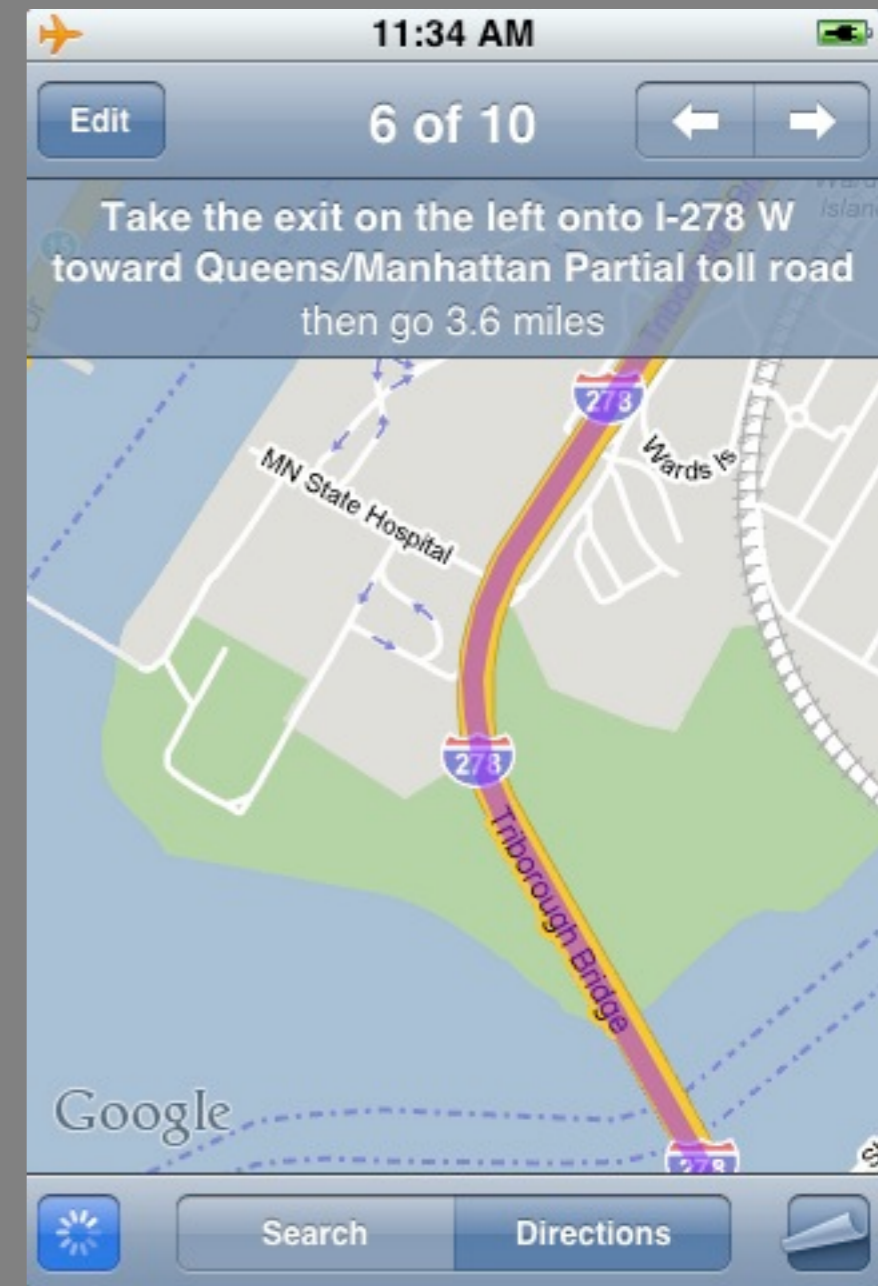
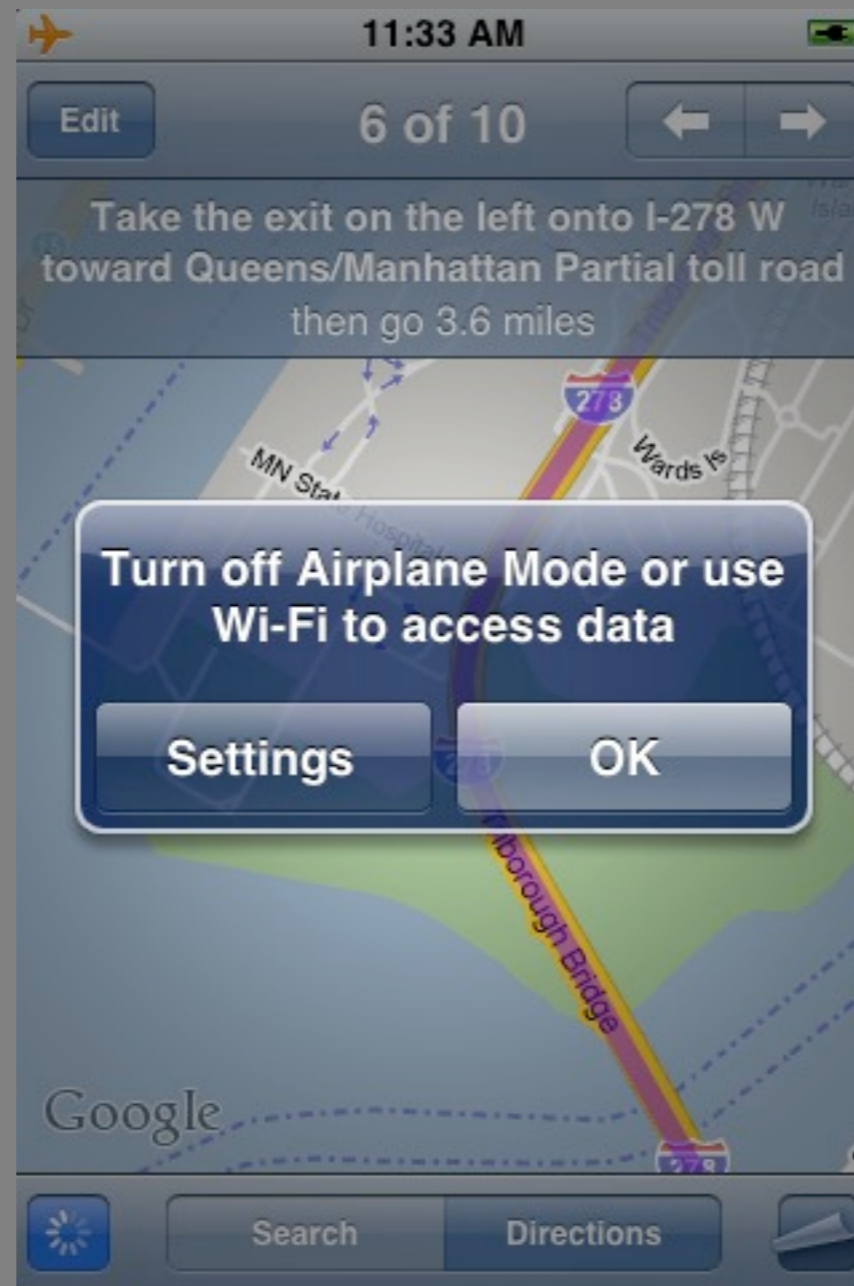
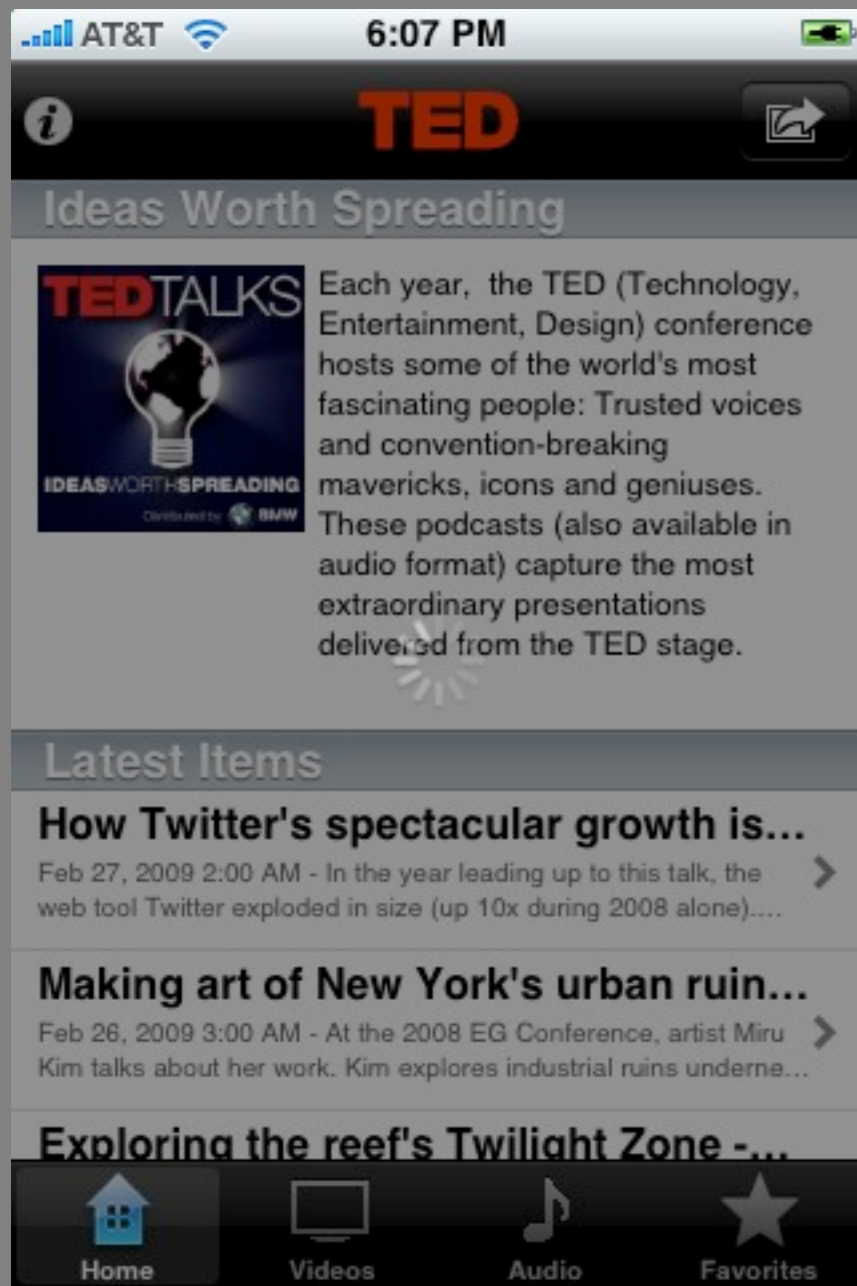
No memory



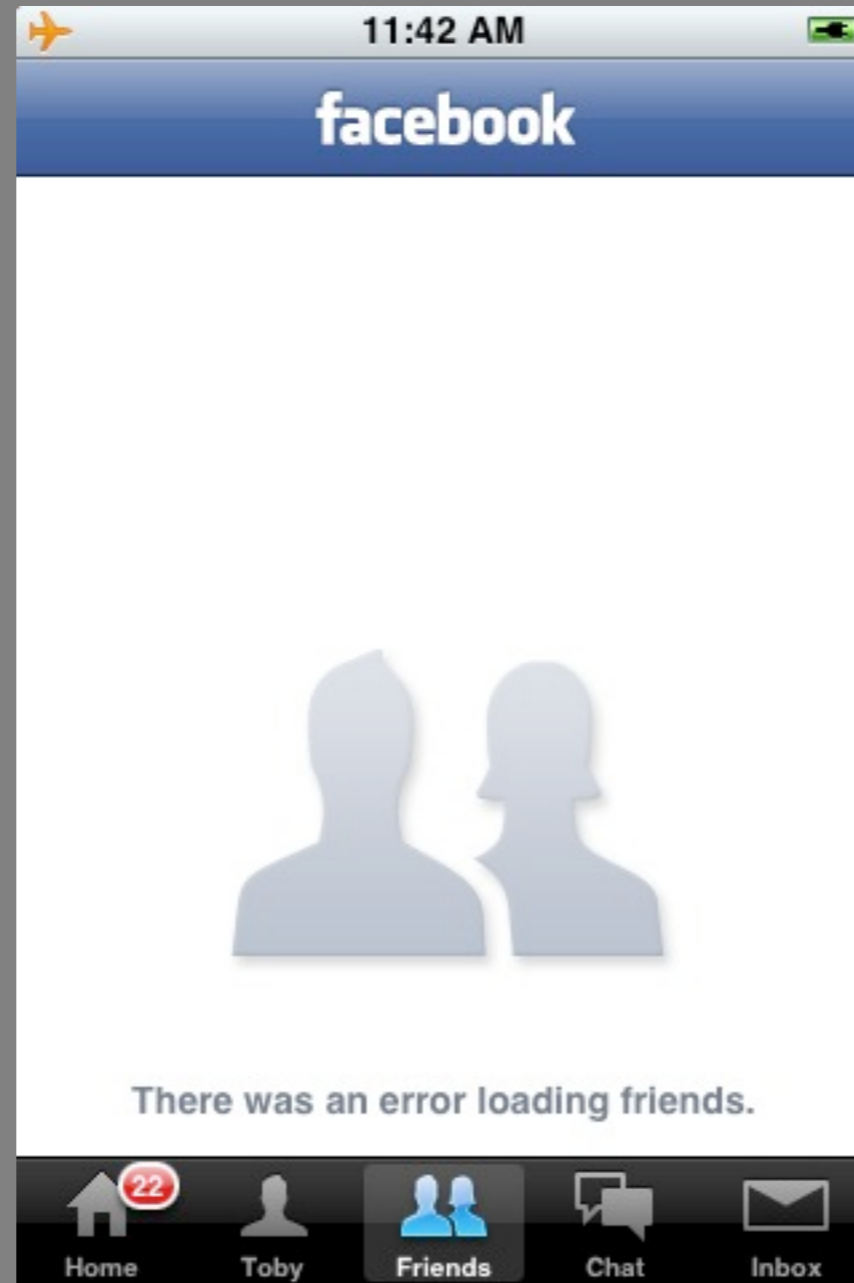
For the first launch, it's ok to be empty



After the first sync, show the last-known state



Message failures appropriately



Gesture Hijacking

- ▶ Using established gestures for novel behavior in a single app

Gesture Hijacking

- ▶ Gestures are learned, not intuited

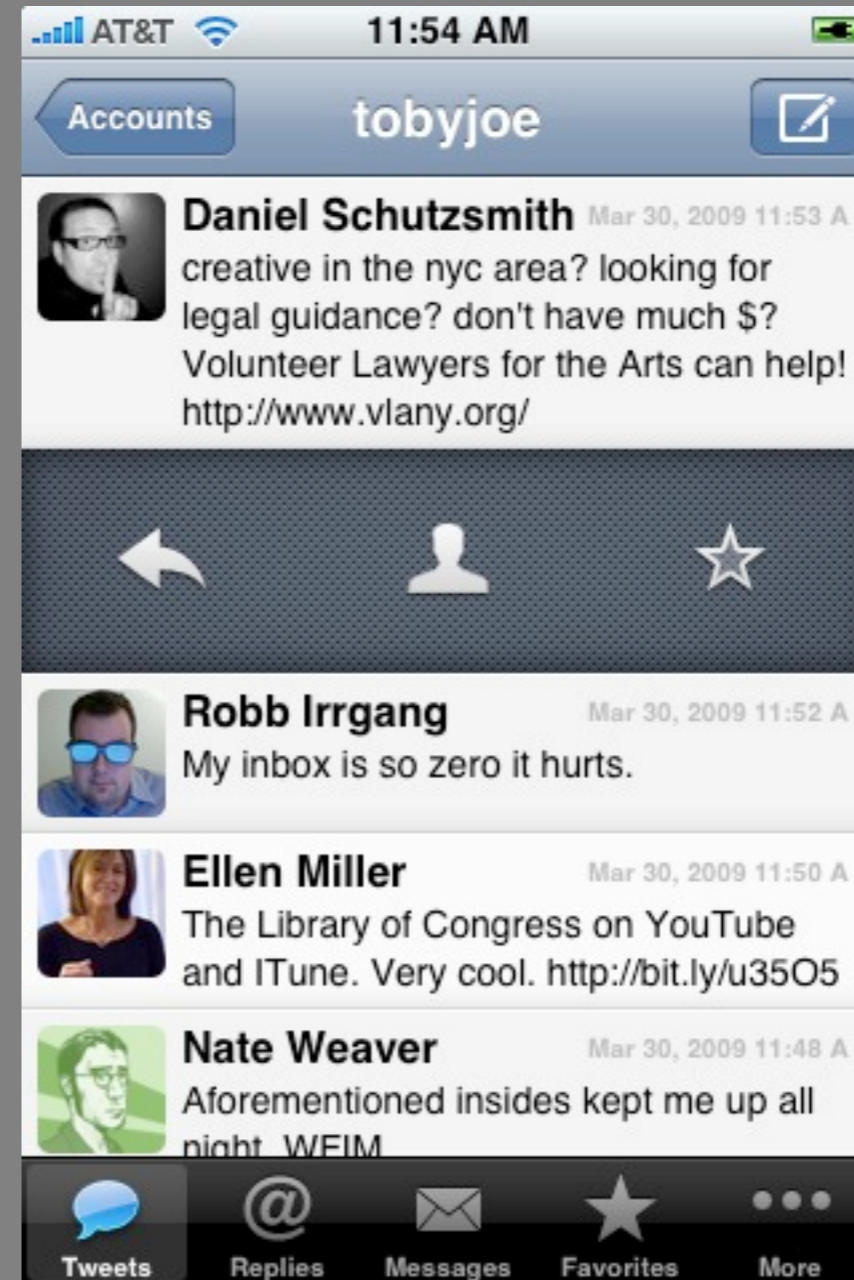
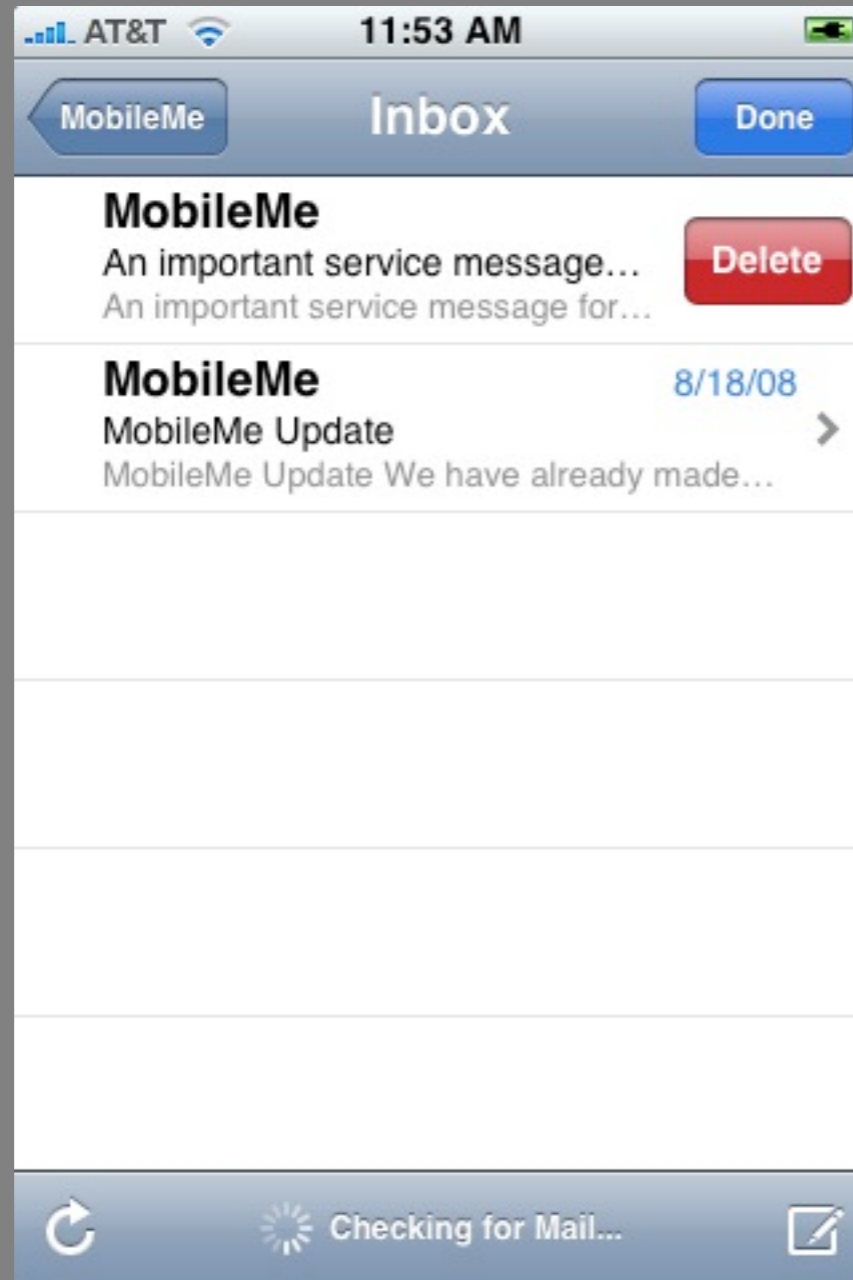
Gesture Hijacking

“...a brand new interface probably won't be intuitive at the beginning.” – **Aza Raskin**

Gesture Hijacking

- ▶ Hijacking learned gestures injects uncertainty, hindering the user experience

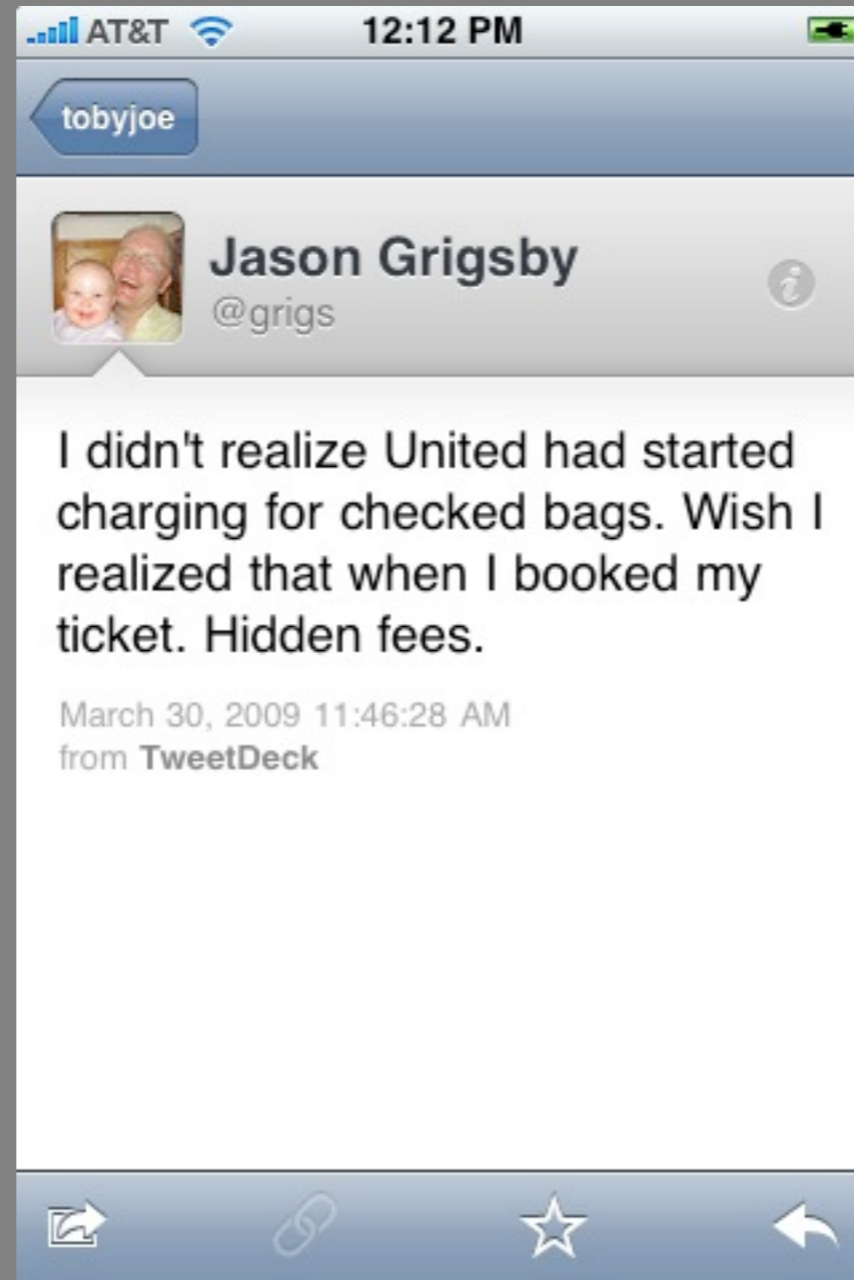
Hijacking the table cell swipe



Gesture Hijacking

- ▶ Crowded interfaces are a real challenge
- ▶ Novel gestures might seem like a solution, but they too must be learned and may not be PE-compliant

A better pattern exists: detail views



Gesture Hijacking

- ▶ Introducing new learning curves is a risk, and the blowback might be larger than your app

Spin Zone

- ▶ Implementing rotation support for views inconsistently, or forcing it arbitrarily

Spin Zone

- ▶ *Why can't I rotate this built-in browser?*
- ▶ *Why can I only play this game rotated to the left?*
- ▶ *Why do I have to rotate to see feature (x)?*

Spin Zone

- ▶ Rotation support is great, but testing for it is work
- ▶ Support it fully, or not at all
- ▶ Match user expectations for the type of app

Sound-Off

- ▶ Hijacking the audio output

Sound-Off

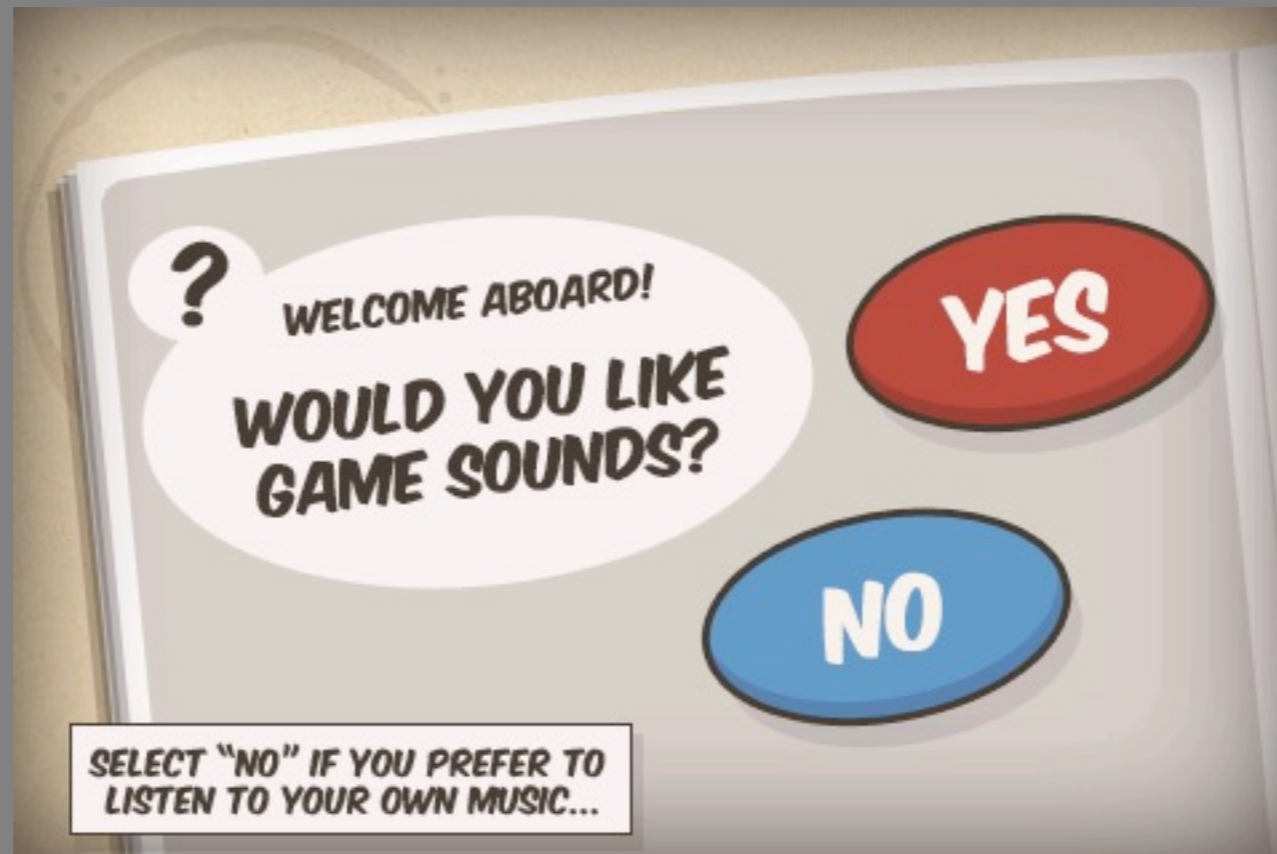
- ▶ Secret: the iPhone is an iPod with a phone built in

Sound-Off

- ▶ The iPhone has rich audio frameworks
- ▶ Blending audio is pretty easy

Sound-Off

- ▶ Give users the option
- ▶ If you can't do that, blend your audio



Summary

- ▶ Some anti-patterns are emerging
- ▶ That means stuff is being built, used
- ▶ Compete in the App Store, cooperate on the device
- ▶ Differentiate, but not at the expense of consistency
- ▶ Keep it up!