
Google™
11
I/O

Designing and Implementing Android UIs for Phones and Tablets

Matias Duarte
Rich Fulcher
Roman Nurik
Adam Powell
Christian Robertson

#io2011 #Android

Ask questions

http://goo.gl/mod/zdyR

Give feedback

http://goo.gl/4dTQp

Note: Both links are also available in the Google I/O Android App

Agenda

1. Introduction to tablets

2. Honeycomb visual design

3. Tablet UI patterns + Honeycomb framework features
   - Interaction design
   - Implementation

4. Case study — Google I/O 2011 App

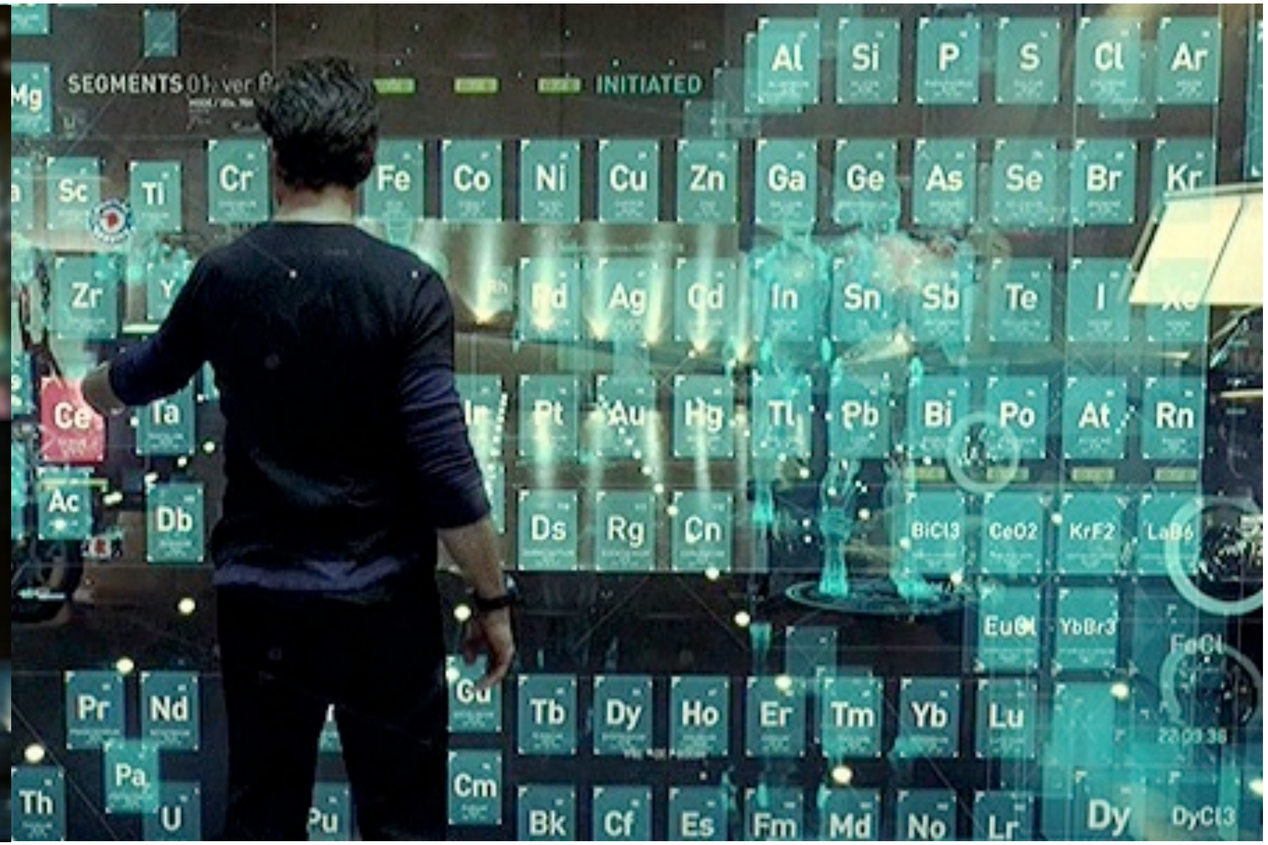
Introduction to tablets

Design Goals for Honeycomb

Design Goals for Honeycomb

Design Goals for Honeycomb

The Beginning of a Journey



Honeycomb visual design

Introducing: Holographic UI

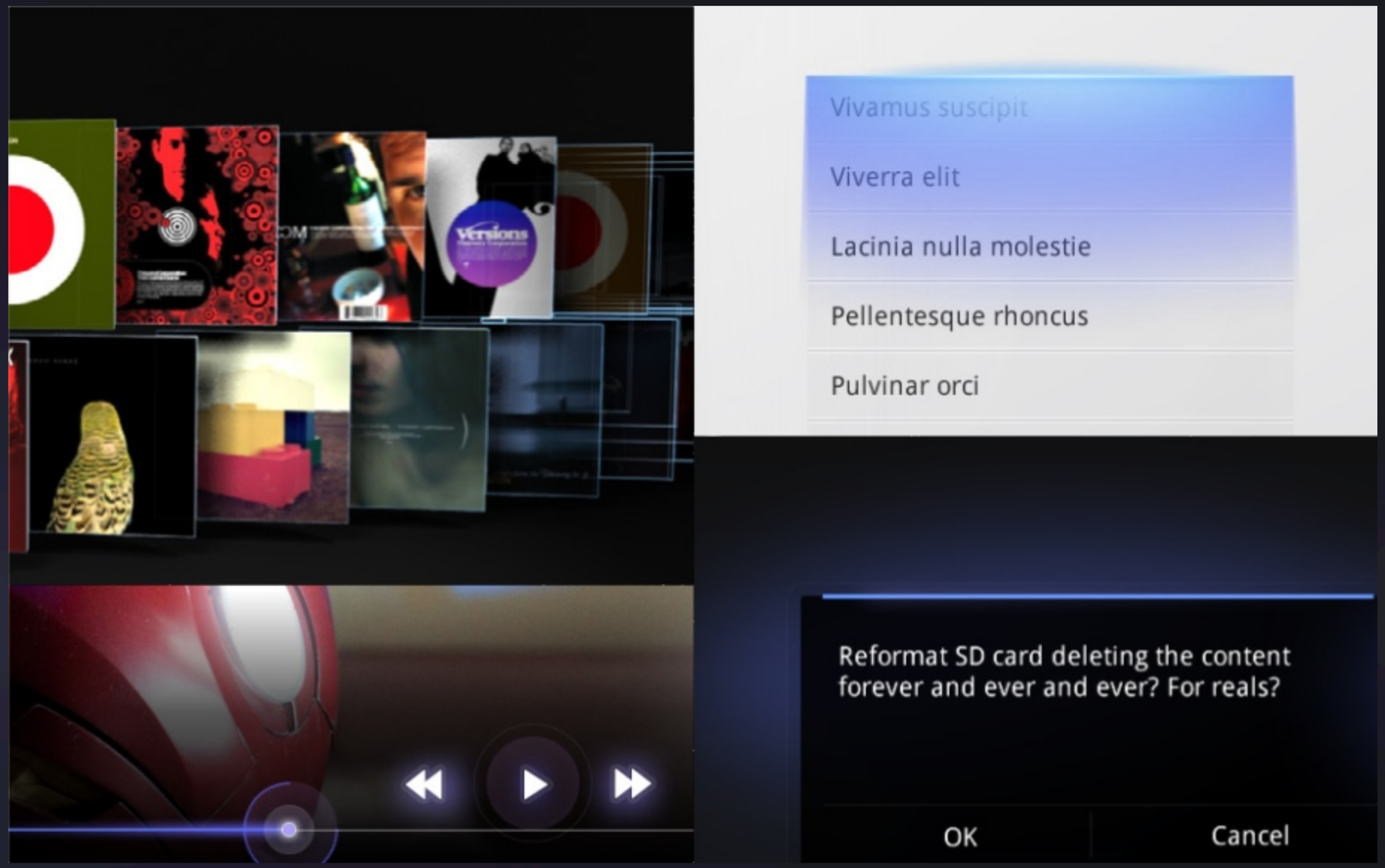
Holo Theme: Design Goals

1 / Enable flexible, dynamic UI

2 / Simplify, open up design

3 / Enable extensibility

1 / Enable dynamic UI: early explorations

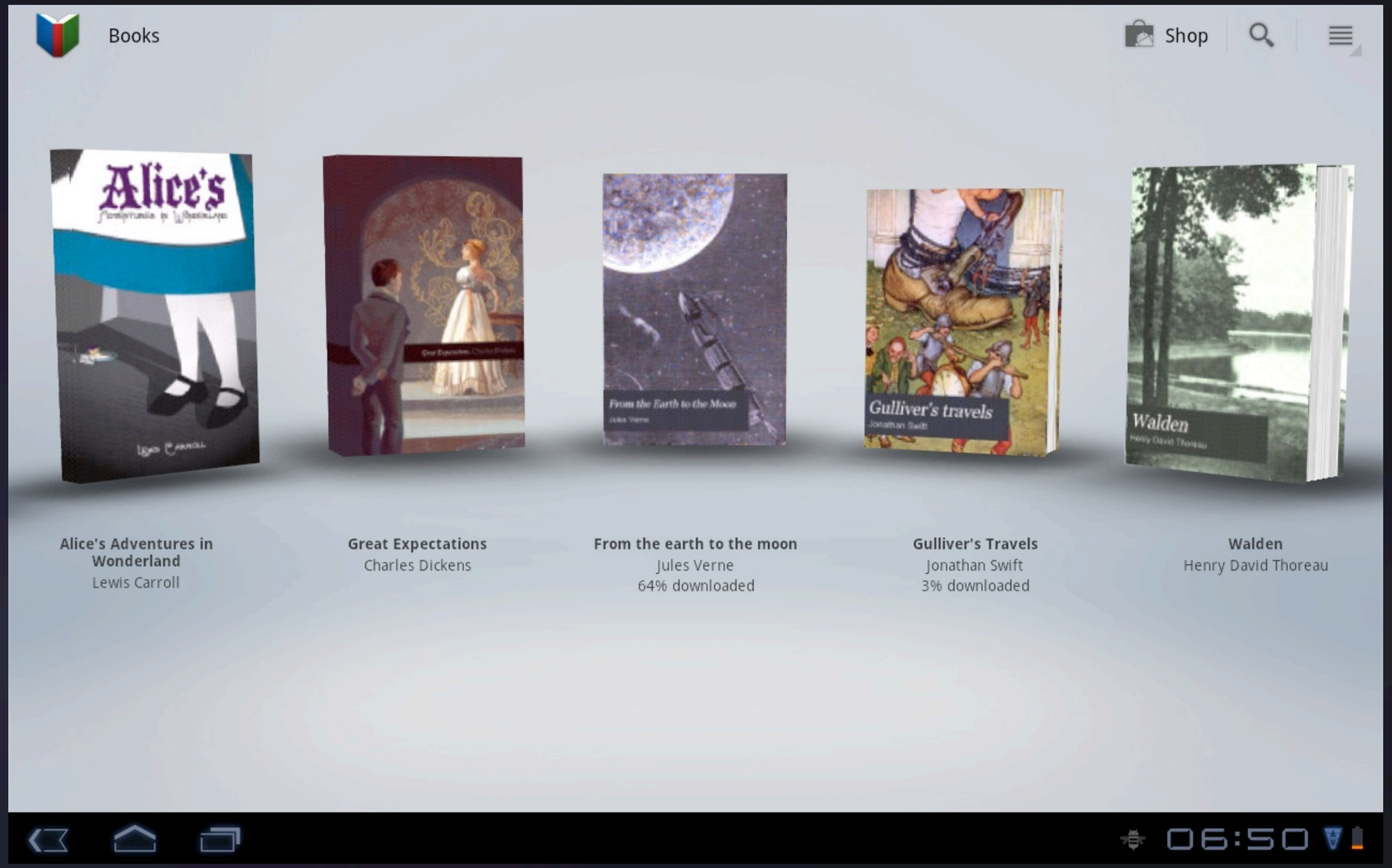


1 / Enable flexible, dynamic UI

1 / Enable flexible, dynamic UI

1 / Enable flexible, dynamic UI

1 / Enable flexible, dynamic UI



Holo Theme: Design Goals

1 / Enable flexible, dynamic UI

2 / Simplify, open up design

3 / Enable extensibility

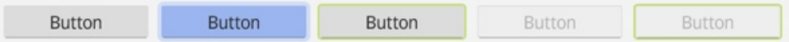
Honeycomb

CORE ELEMENTS | **HOLO LIGHT**

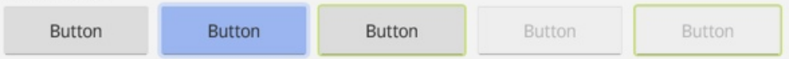
ACTION BAR

CONTEXTUAL ACTION BAR

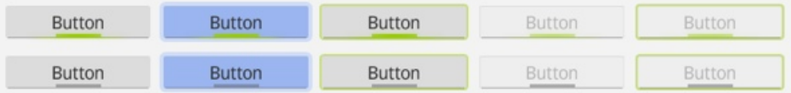
SMALL BUTTONS



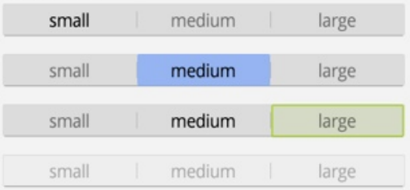
DEFAULT BUTTONS



TOGGLE BUTTONS



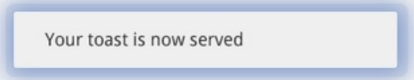
GROUP BUTTONS



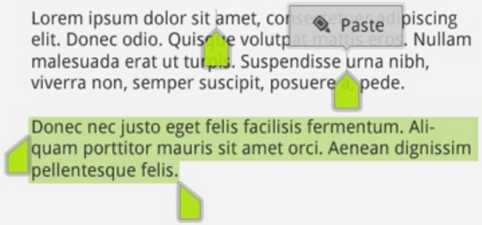
MENU DROPDOWN

DIALOG

TOAST

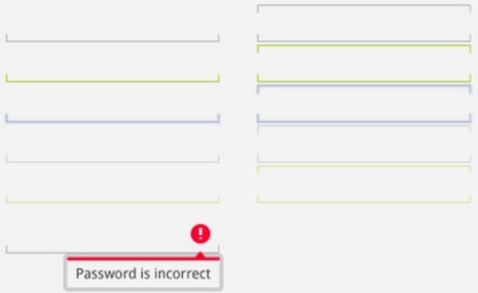


TEXT SELECTION (COPY & PASE)



QUICK ACTION/QUICK CONTACT

TEXTFIELDS

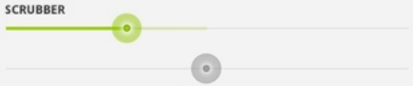


TABS

SCROLLBARS

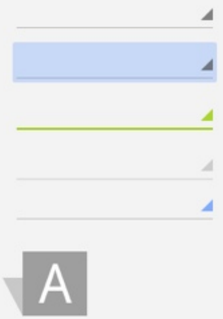
PROGRESS BAR

SCRUBBER



FASTSCROLLER

SPINNER (DROPDOWN)



SWITCHES

RADIO BUTTONS

CHECKBOXES

BUTTON STARS

RATING STARS

INDETERMINATE PROGRESS SPINNER

SINGLE LINE ITEM LIST

- Lorem ipsum dolor
- Aliquam tincidunt mauris
- Fusce pellentesque
- Cras ornare

LIST BACKGROUND STATES

- Normal
- Pressed
- Focused
- Disabled
- Activated
- Selected

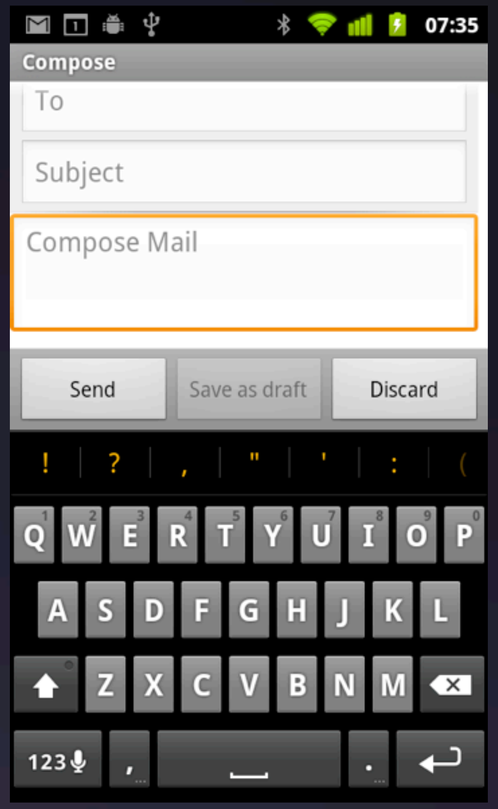
ASSORTED LIST ITEMS

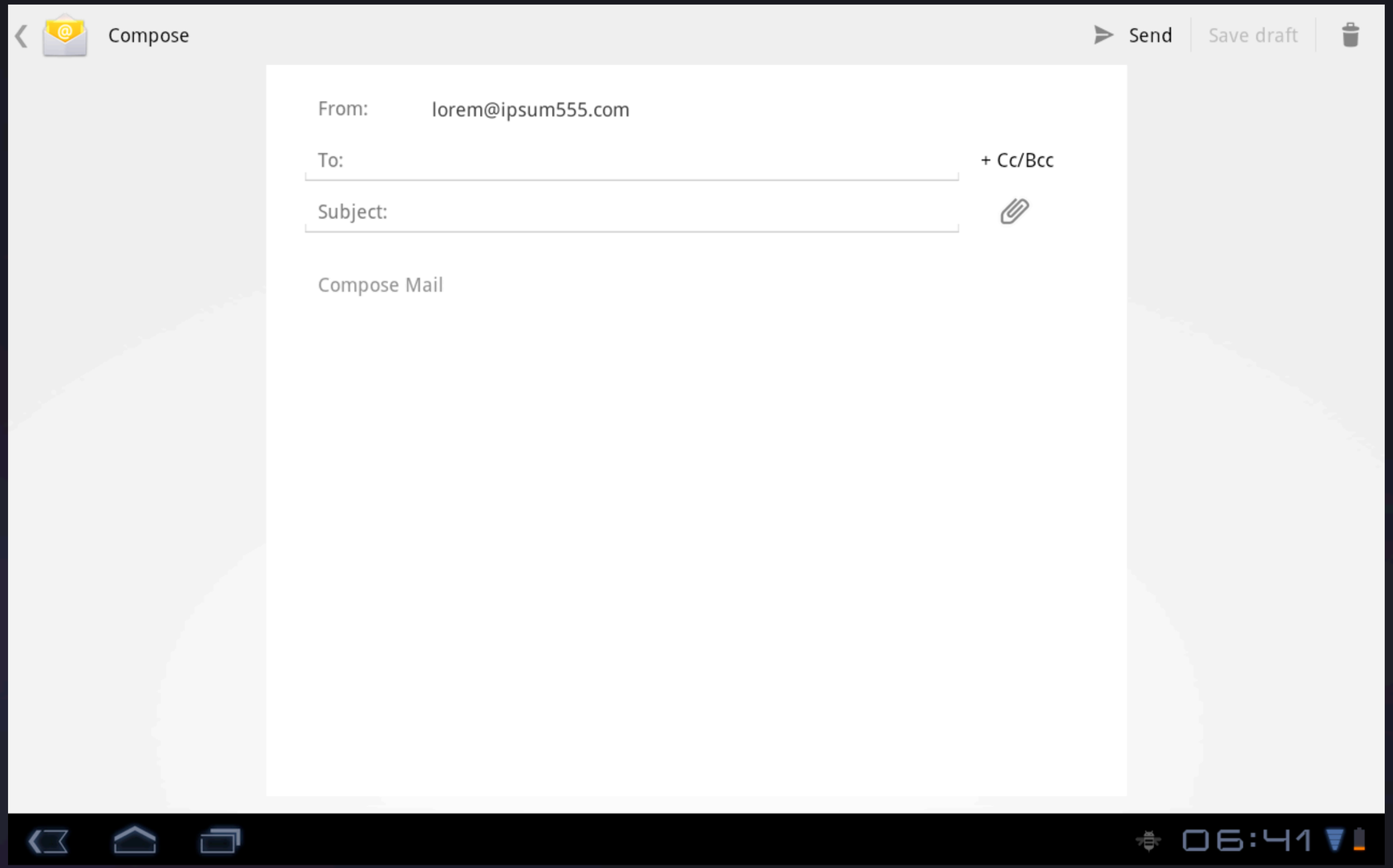
Morbi in sem quis dui placerat ornare
Pellentesque odio nisi, euismod in, pharetra a, ultricies in, diam Sed arcu. Cras consequat

Pellentesque fermentum dolor
Aliquam quam lectus, facilisis auctor, ultrices ut, elementum

SECTION DIVIDERS & EXPANDING ITEMS

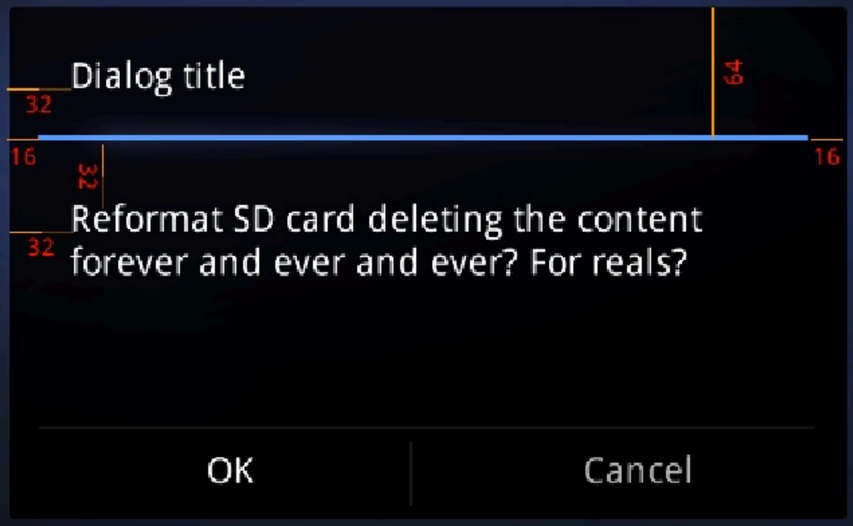
2 / Simplify: removing boxes



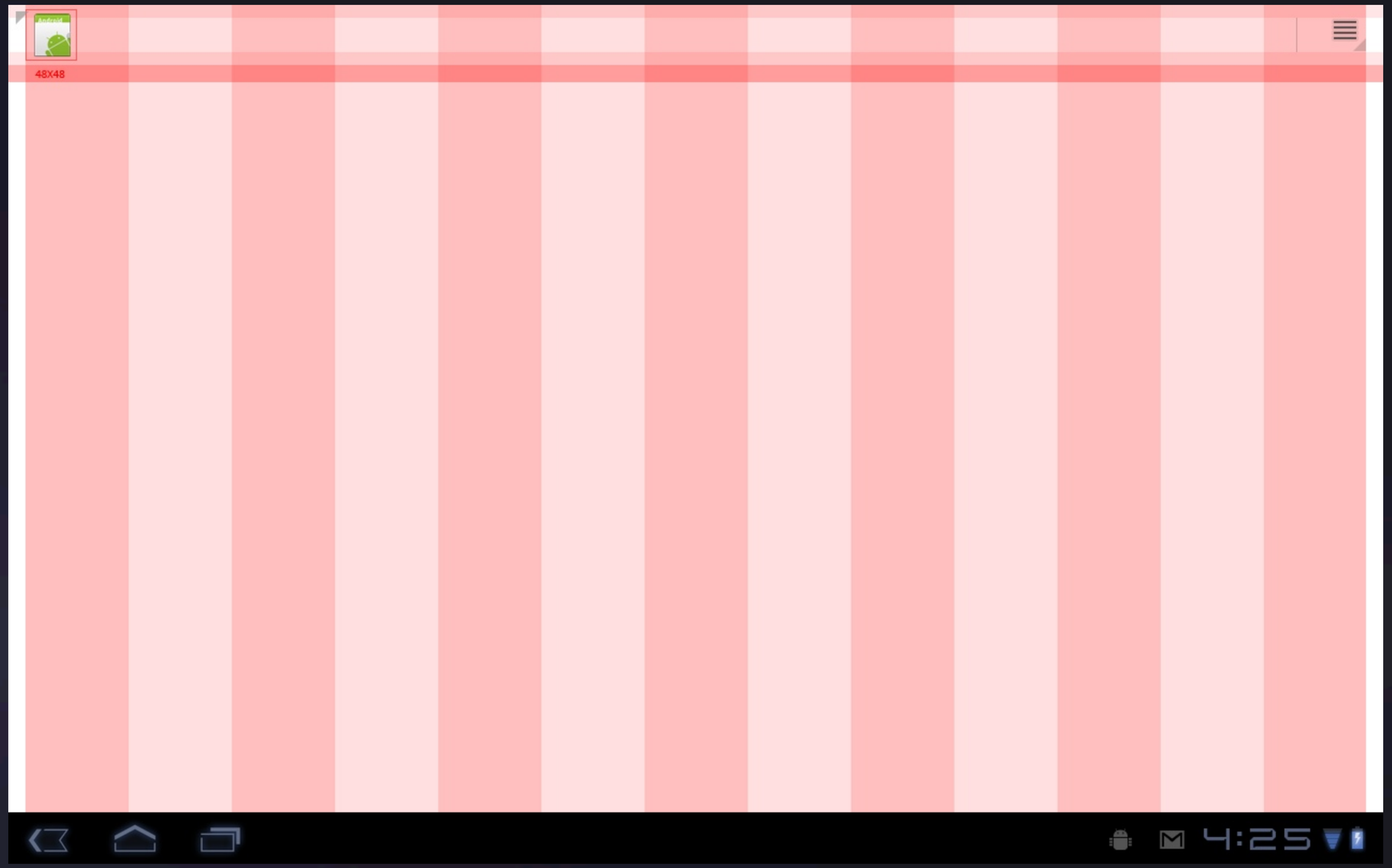


2 / Simplify: removing boxes

2 / Simplify: robust spacing, grid metrics



2 / Simplify: robust spacing, grid metrics



Holo Theme: Design Goals

1 / Enable flexible, dynamic UI

2 / Simplify, open up design

3 / Enable extensibility

3 / Enable extensibility: Design tips

- Choose light or dark theme as a start point
  - Consider changing highlight color and background image
  - When overriding styles and themes, override all assets
- Open design, removing boxes where appropriate
  - Spacing and metrics matter
  - “Stretch to fill” on tablets doesn’t work for many UI elements
- Spend time on icons & promo graphics ...

Application Branding

Density-specific app icons

Hi-res Market icon

Application Branding

Feature banner graphic

Promotional graphic

Honeycomb UI patterns and framework features

UI Patterns

- Like a software design pattern, a UI design pattern describes a general solution to a recurring problem
- Framework-supported
- Guidelines, not restrictions
- Topics we'll discuss today:
  1. Action Bar
  2. Multi-pane Layouts
  3. App Navigation
  4. Beyond the List

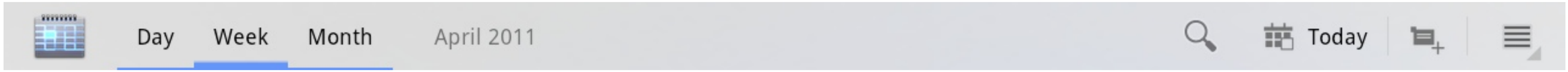
Action Bar

Introduction

- Not a new pattern
  - Presented as phone UI pattern at last year's I/O
  - Used in many apps through Android Market
  - Honeycomb has greatly extended its usefulness
- Dedicated real estate at the top of each screen
  - Generally persistent throughout application
- Used to make frequently used actions prominent
- Supports navigation
- Convenient means of handling Menu and Search

Action Bar

General organization



- App icon
- View details
- Action buttons

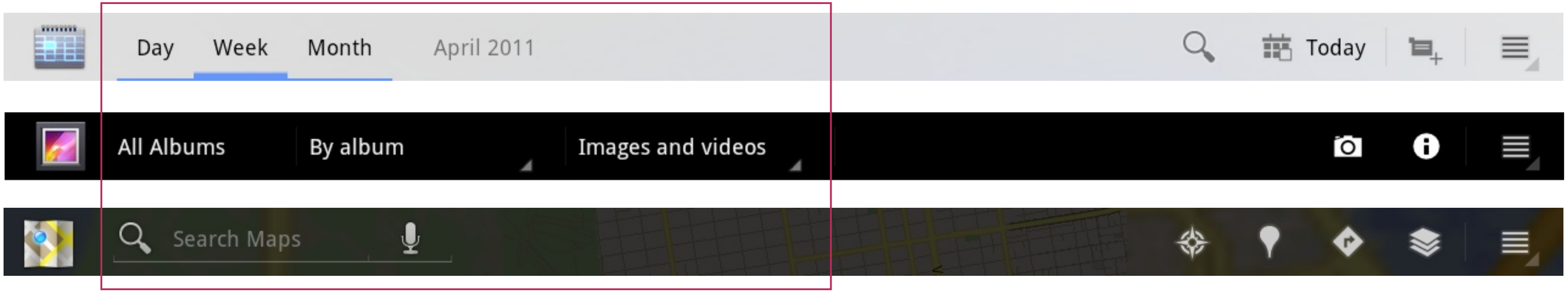
Action Bar

General organization

- App icon
  - Can be replaced with logo or other branding
  - Used to support “upward” navigation within the app

Action Bar

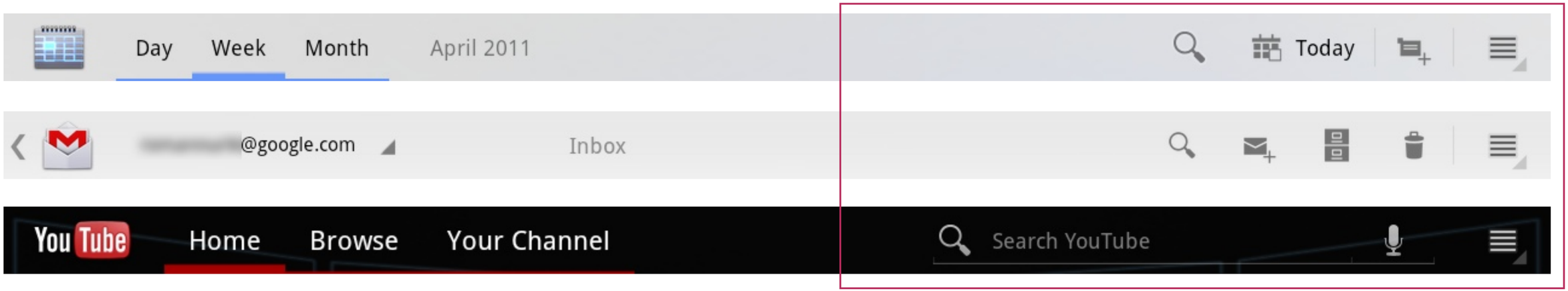
General organization



- View details
  - Simple: non-interactive title bar replacement
  - Richer: Tabs, drop-down menus, breadcrumbs

Action Bar

General organization



- Action buttons
  - More important / frequently-accessed action at left
  - Buttons can be icon-only, text-only, or icon-and-text
  - Overflow menu

Action Bar

Contextual actions

- Action bar can transform its appearance when items are selected
  - Useful for single or multiple selection
  - Typically invoking via touch and hold
- Like normal action bar, three sections:
  - **Done** button (for releasing selection)
  - Selection details
  - Action Buttons
- Implemented using `ActionMode`

Action Bar

Implementation

- Basic action bar
  - `Theme.Holo` or `targetSdkVersion` ≥ **11**.
  - Action items from `res/menu/`
- Customizing the action bar
  - `ActionBar` class
  - `showAsAction` for menu items

http://j.mp/customizing-action-bar

Action Bar

Compatibility

1. Write a custom action bar implementation pre-Honeycomb

http://code.google.com/p/iosched

2. Alternatively, defer to the standard Options menu

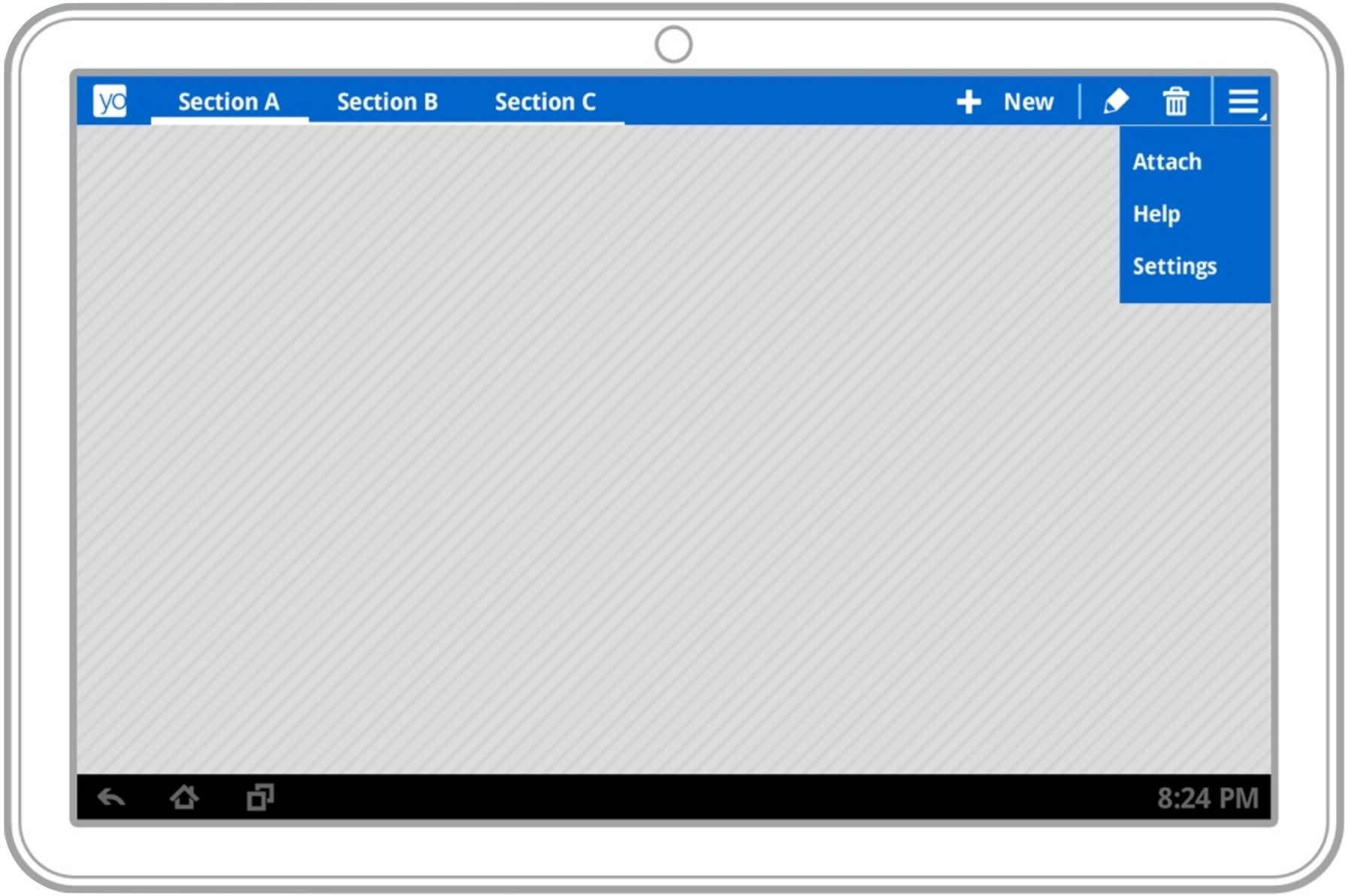
Action Bar

Phones and smaller screens

- Swap out elements for space-conservative variants
  - Icon + text reduced to just icon
  - Overflow icon hidden, invoked using **MENU** key
- Split single bar into two
  - View portion such as Tabs can become second row, below action bar
  - Actions can move into bottom action bar
- 2-3 main action buttons, others placed in Overflow
  - Determined by `showAsAction` = “`ifRoom`” or “`always`”

Action Bar

Phones and smaller screens



Multi-pane Layouts

Introduction

- Take advantage of vastly increased real estate
  - Give more context
  - Consolidate multiple related phone screens into a single compound view
  - Avoid excessively long line lengths

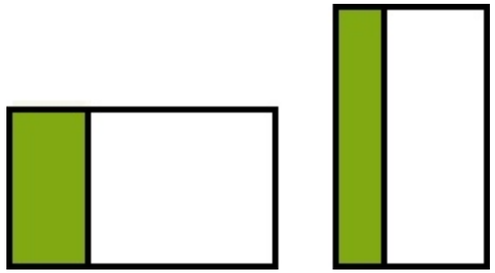
- Panes to the right should generally present more content or details for items selected in the panes on the left.

Multi-pane Layouts

Orientation change

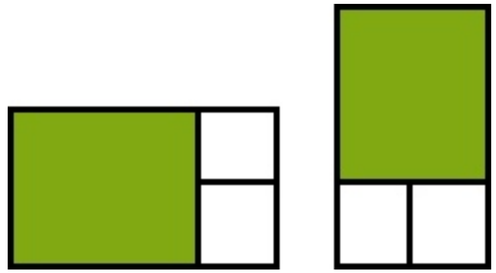
- Orientation changes should preserve functional parity
  - User shouldn't have to rotate device to achieve a task
- Strategies apply per-screen, not per app
- For the **show/hide** orientation strategy, use **UP** navigation to show the master pane
  - e.g. Gmail conversation view

Strategies



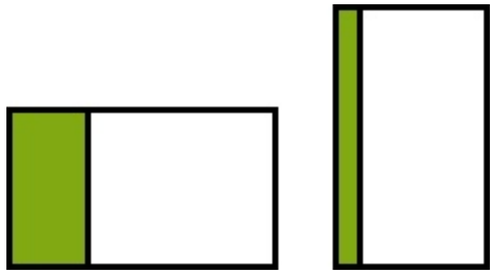
Stretch

(e.g. Settings)



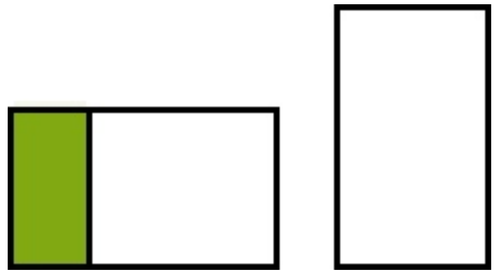
Stack

(e.g. Calendar)



Expand/collapse

(e.g. Google Talk)



Show/hide

(e.g. Gmail)

Multi-pane Layouts

Implementation — Fragments

- `Fragment` class
- Optionally use the `<fragment>` tag in layout XML

Multi-pane Layouts

A quick intro to Fragments

- “Fragments” of an Activity
- Unit of reuse between Activities
- Separation of concerns
- Fragments don’t necessarily have views
  - **Fragments are a lifecycle construct**, not necessarily a visual construct
- ...but this talk is about UI.

Multi-pane Layouts

Compatibility

- Can use Fragments with the Android support library available through the SDK manager
- Use `getSupportFragmentManager`
- All activities extend `FragmentActivity`

http://j.mp/fragments-for-all

Multi-pane Layouts

Using resources

- You’ve probably seen this before:
  - drawable-ldpi/
  - drawable-mdpi/
  - drawable-hdpi/
- But you can also do this:
  - layout-normal/
  - layout-large/
  - layout-xlarge/
  - layout-xlarge-port/

Multi-pane Layouts

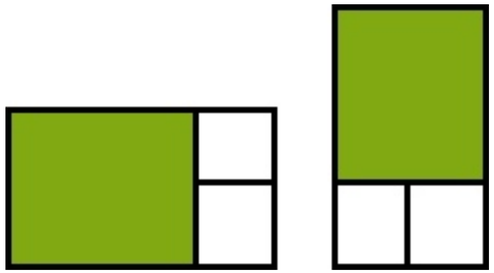
Using resources

- Activities can inflate layouts with different fragment configurations

Multi-pane Layouts

Using resources

- Activities can inflate layouts with different fragment configurations



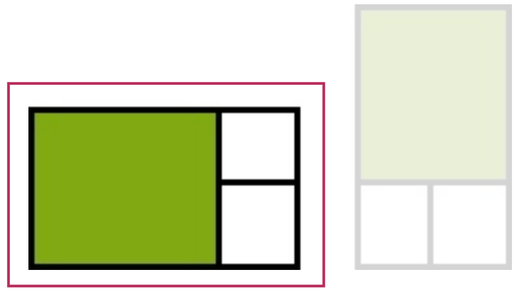
Stack

(e.g. Calendar)

Multi-pane Layouts

Using resources

- Activities can inflate layouts with different fragment configurations



Stack

(e.g. Calendar)

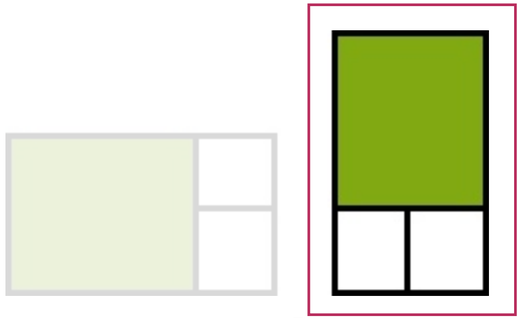
layout-xlarge-land/my_layout.xml

```
<LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
  <fragment android:name="com.example.MainPaneFragment"
      android:id="@+id/main_pane"
      android:layout_width="0dip" android:layout_weight="1"
      android:layout_height="match_parent" />
  <LinearLayout android:orientation="vertical"
      android:layout_width="wrap_content"
      android:layout_height="match_parent">
    <fragment android:name="com.example.MonthFragment"
        android:id="@+id/month_pane"
        android:layout_width="wrap_content"
        android:layout_height="0dip" android:layout_weight="1" />
    <fragment android:name="com.example.CalendarListFragment"
        android:id="@+id/list_pane"
        android:layout_width="wrap_content"
        android:layout_height="0dip" android:layout_weight="1" />
  </LinearLayout>
</LinearLayout>
```

Multi-pane Layouts

Using resources

- Activities can inflate layouts with different fragment configurations



Stack

(e.g. Calendar)

layout-xlarge-port/my_layout.xml

```
<LinearLayout android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
  <fragment android:name="com.example.MainPaneFragment"
      android:id="@+id/main_pane"
      android:layout_width="match_parent"
      android:layout_height="0dip" android:layout_weight="1" />
  <LinearLayout android:orientation="horizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
    <fragment android:name="com.example.MonthFragment"
        android:id="@+id/month_pane"
        android:layout_width="0dip" android:layout_weight="1"
        android:layout_height="wrap_content" />
    <fragment android:name="com.example.CalendarListFragment"
        android:id="@+id/list_pane"
        android:layout_width="0dip" android:layout_weight="1"
        android:layout_height="wrap_content" />
  </LinearLayout>
</LinearLayout>
```

Multi-pane Layouts

Using resources

- Some other handy uses:
  - Fragments can use layouts with different view configurations
  - List items can be more detailed or compact as needed
  - `integer` or `boolean` resources can have different values

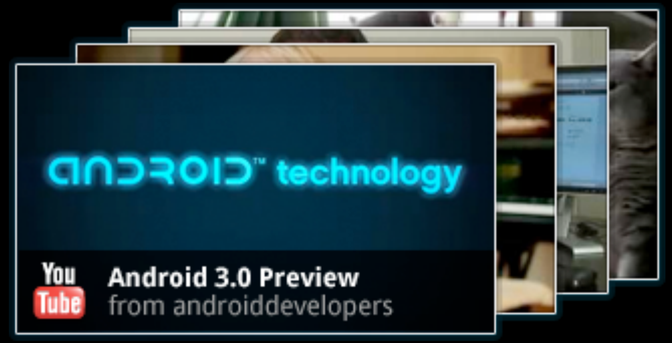
App Navigation

Introduction

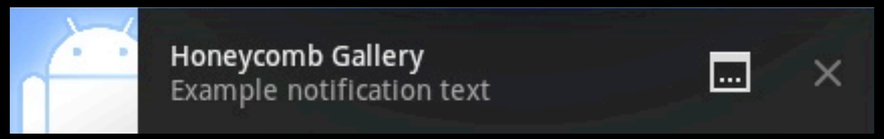
- One of the more dramatic changes in Honeycomb
- **Increased variety of mechanisms for direct, deep navigation into an app**

App Navigation

Highlights



Richer home screen widgets



Richer notifications

‘Recents’

App Navigation

Navigation and user memory

- Android has traditionally relied on **temporal** memory:
  - Users are good at remembering what **just** happened
  - Great for snapping back to one context directly from another
  - Much harder to sequence precise order of events that happened a little while ago
  - More potential for error, surprise
- Users have strong **structural** memory
  - Remember relationships between screens in an app
  - Used to going “Home” in web apps
  - Clearer expectations for behavior

App Navigation

Back versus Up

- SYSTEM **BACK** navigates history between related screens
- APPLICATION **UP** navigates hierarchy within a single app

App Navigation

Example Flows

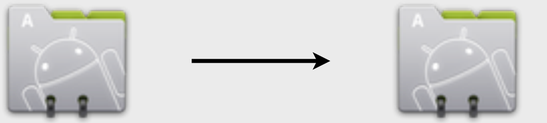
Contacts Task

Contacts

App Navigation

Example Flows

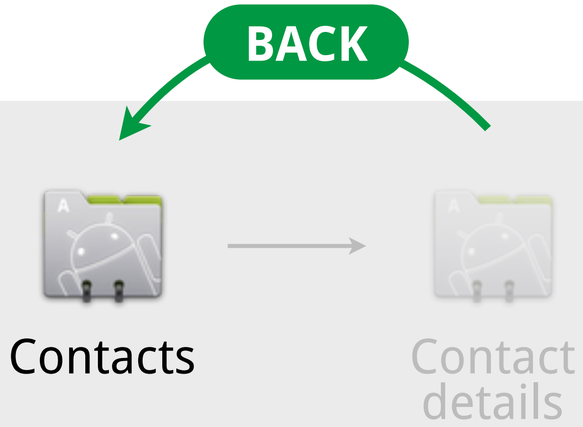
Contacts Task



App Navigation

Example Flows

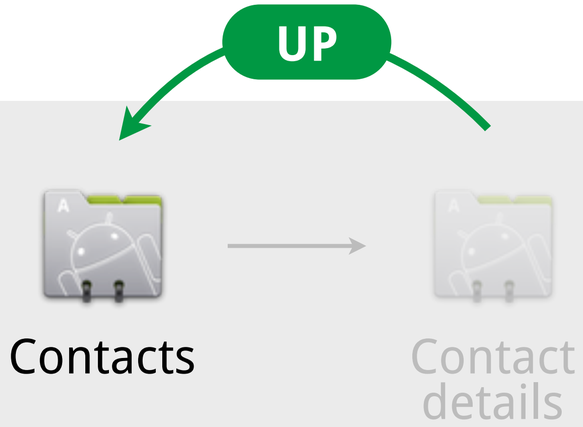
Contacts Task



App Navigation

Example Flows

Contacts Task



App Navigation

Example Flows

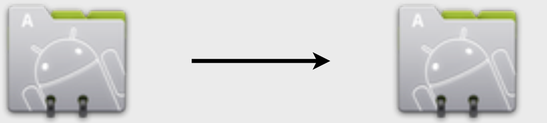
Contacts Task

Contacts

App Navigation

Example Flows

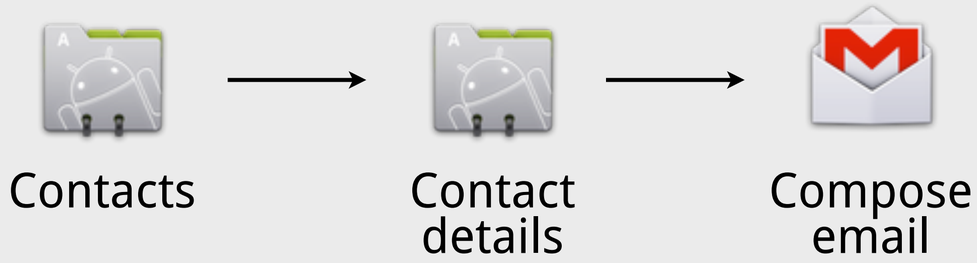
Contacts Task



App Navigation

Example Flows

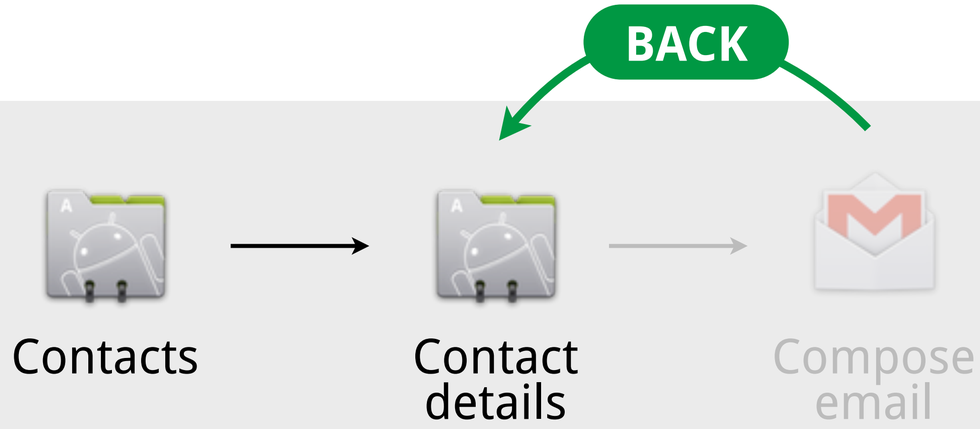
Contacts Task



App Navigation

Example Flows

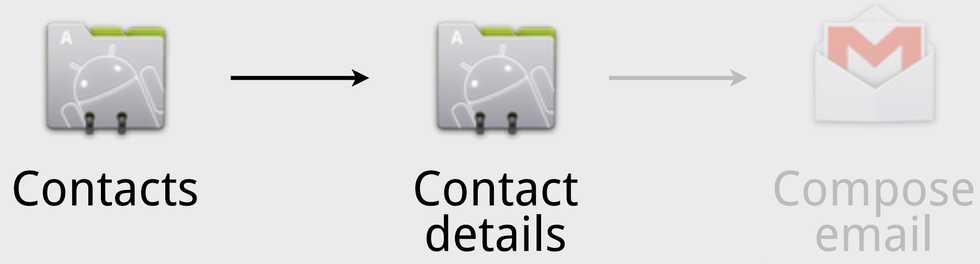
Contacts Task



App Navigation

Example Flows

Contacts Task



Gmail Task

App Navigation

What you need to do

- If you have an app with hierarchy, support **UP** in action bar
- If you support **system** deep links into your app, inject screens “above” the target into the back stack
  - E.g. Deep link from a widget
  - E.g. Deep link from a notification

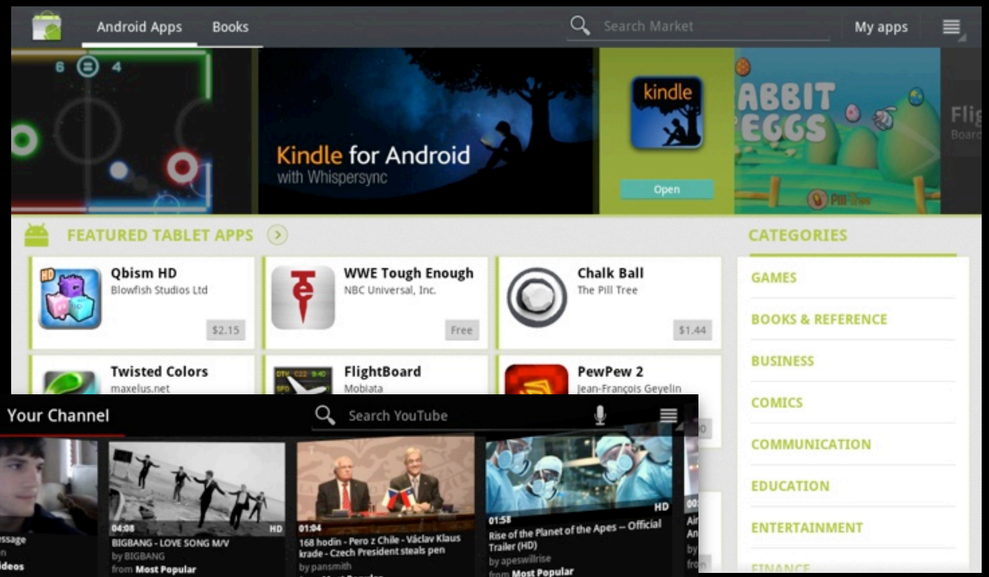
Beyond the List

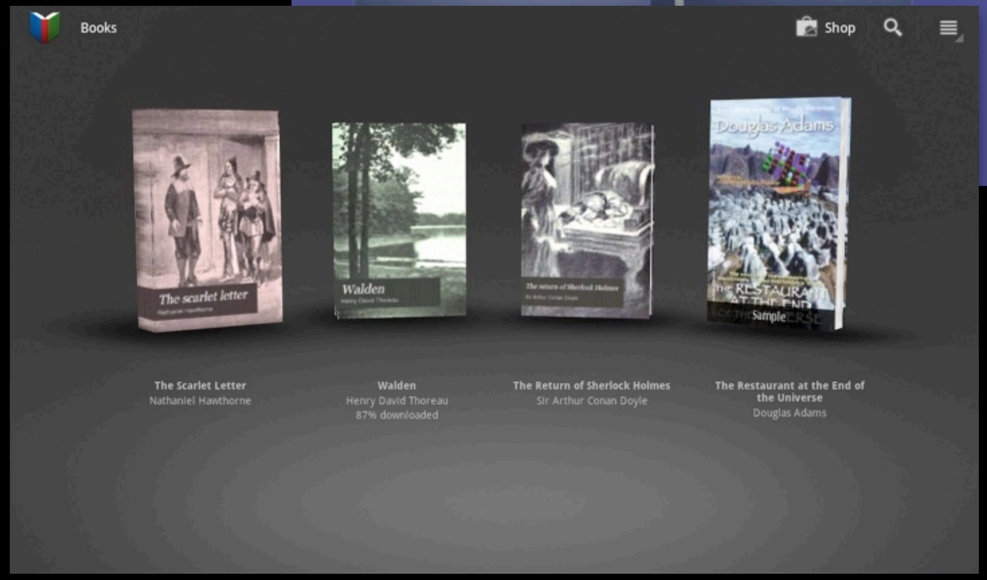
Introduction

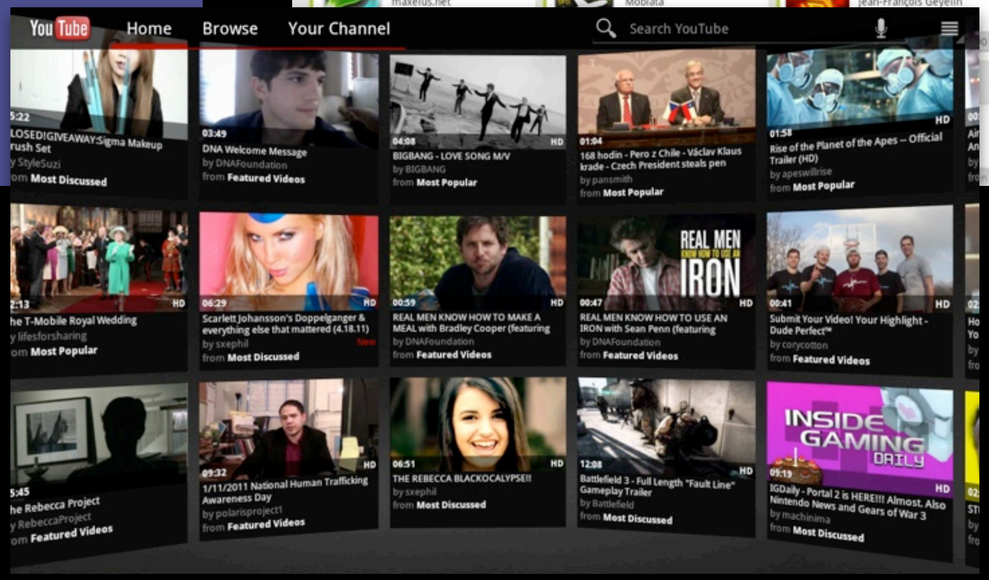
- Views for media-rich applications
- “**Hero moments**” to break the monotony of list views
- Encourage more engaged exploration, akin to flipping through a magazine

Beyond the List

Examples







Beyond the List

Implementation

- **`CarouselView`**
  - Renderscript
  - Intended for customization

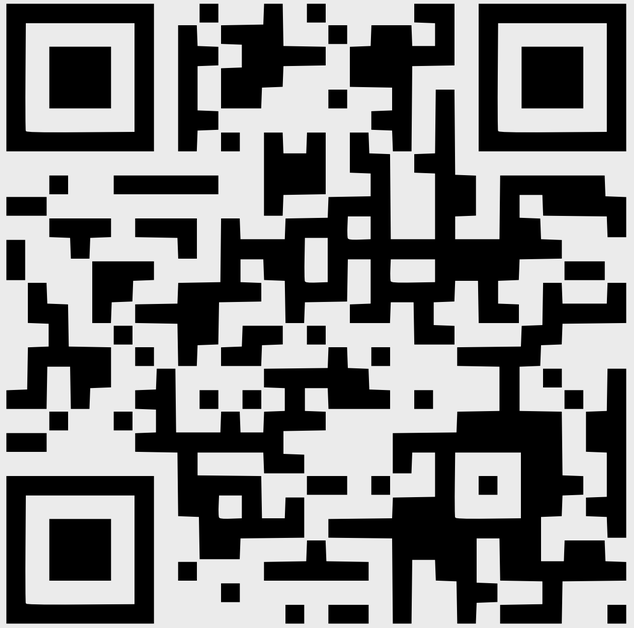
  http://j.mp/io2011-carousel-sample

- **`FragmentPager`**, **`Workspace`** for showing one item or page at a time

- Don’t use the **`Gallery`** widget

Case study — Google I/O 2011 App

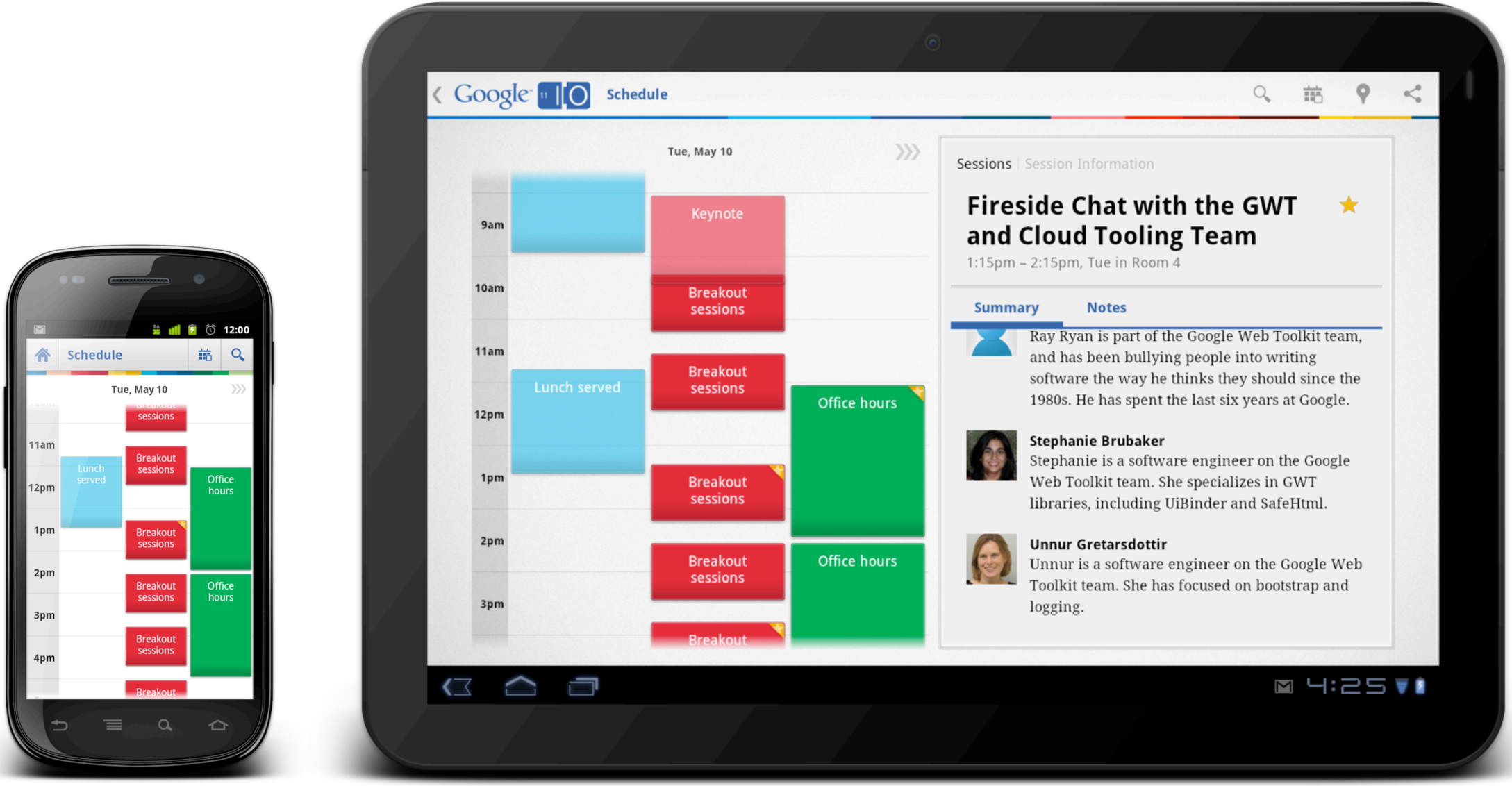
Download the app now.
We'll wait.

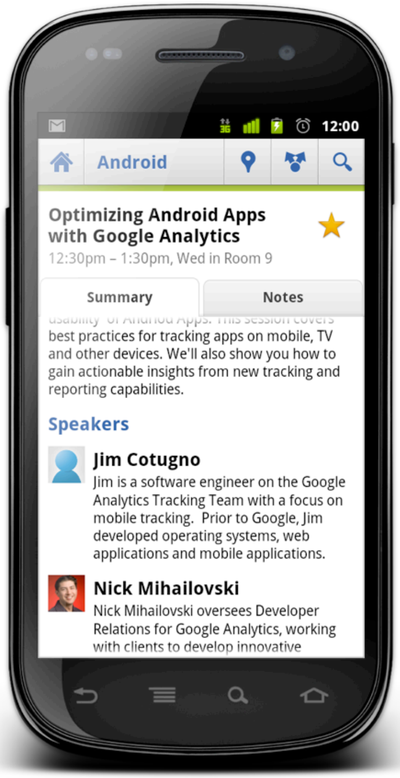


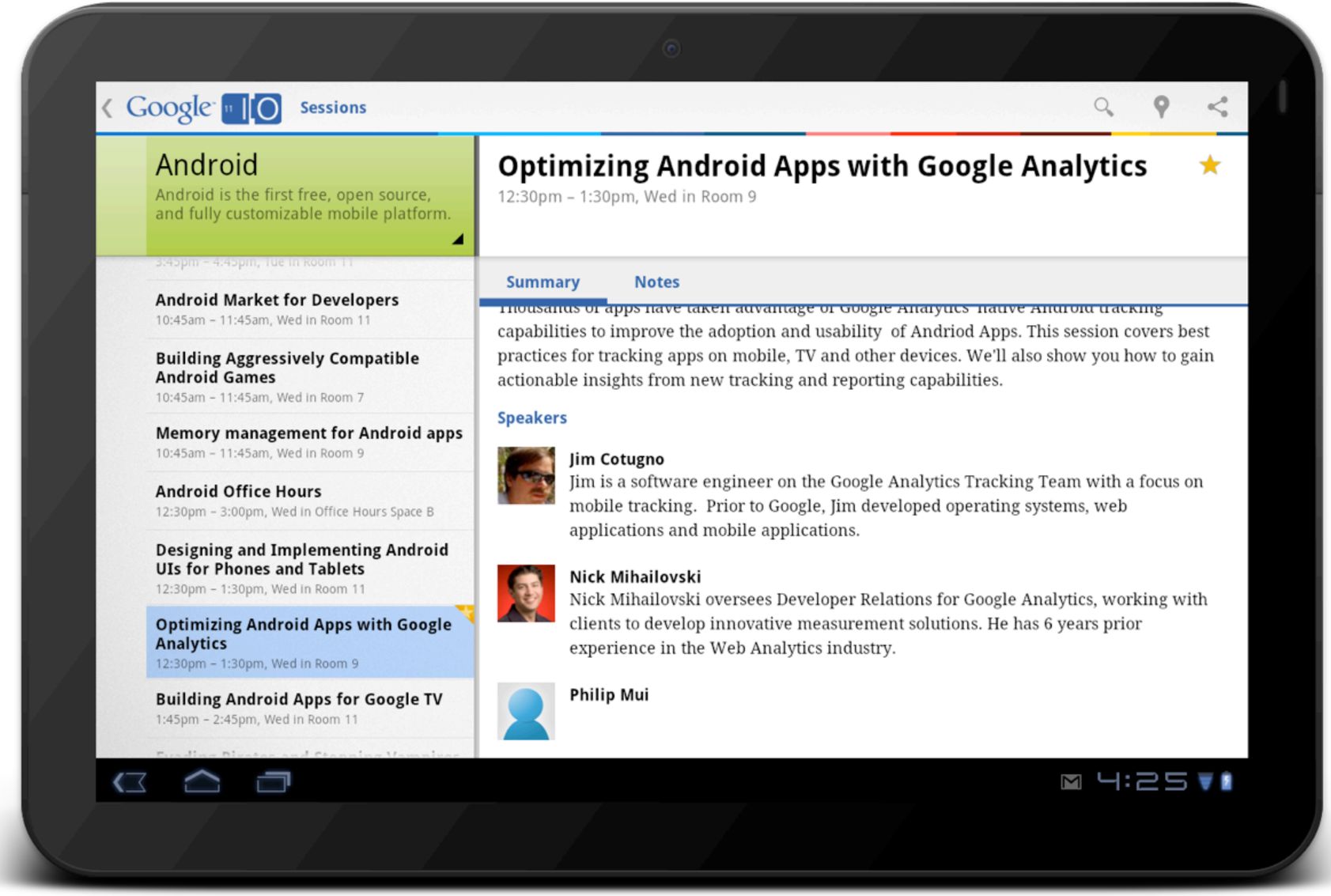
http://goo.gl/UhnLT


Google I/O 11
Schedule
Map
Sessions
Starred
Sandbox
Bulletin
18 days, 16:05:46 until Google I/O
4:25
12:00
20 days, 8:55:32 until Google I/O
Realtime Stream

Google I/O 11


12:00
Schedule
Tue, May 10
11am
12pm
1pm
2pm
3pm
4pm
Lunch served
Breakout sessions
Office hours
Google I/O
Schedule
Tue, May 10
9am
10am
11am
12pm
1pm
2pm
3pm
Keynote
Breakout sessions
Lunch served
Office hours
Sessions
Session Information
Fireside Chat with the GWT and Cloud Tooling Team
1:15pm – 2:15pm, Tue in Room 4
Summary
Notes
Ray Ryan is part of the Google Web Toolkit team, and has been bullying people into writing software the way he thinks they should since the 1980s. He has spent the last six years at Google.
Stephanie Brubaker
Stephanie is a software engineer on the Google Web Toolkit team. She specializes in GWT libraries, including UiBinder and SafeHtml.
Unnur Gretarsdottir
Unnur is a software engineer on the Google Web Toolkit team. She has focused on bootstrap and logging.
4:25


12:00
Android
Optimizing Android Apps with Google Analytics
12:30pm – 1:30pm, Wed in Room 9
Summary
Notes
best practices for tracking apps on mobile, TV and other devices. We'll also show you how to gain actionable insights from new tracking and reporting capabilities.
Speakers
Jim Cotugno
Jim is a software engineer on the Google Analytics Tracking Team with a focus on mobile tracking. Prior to Google, Jim developed operating systems, web applications and mobile applications.
Nick Mihailovski
Nick Mihailovski oversees Developer Relations for Google Analytics, working with clients to develop innovative


Google I/O
Sessions
Android
Android is the first free, open source, and fully customizable mobile platform.
Android Market for Developers
10:45am – 11:45am, Wed in Room 11
Building Aggressively Compatible Android Games
10:45am – 11:45am, Wed in Room 7
Memory management for Android apps
10:45am – 11:45am, Wed in Room 9
Android Office Hours
12:30pm – 3:00pm, Wed in Office Hours Space B
Designing and Implementing Android UIs for Phones and Tablets
12:30pm – 1:30pm, Wed in Room 11
Optimizing Android Apps with Google Analytics
12:30pm – 1:30pm, Wed in Room 9
Building Android Apps for Google TV
1:45pm – 2:45pm, Wed in Room 11
Optimizing Android Apps with Google Analytics
12:30pm – 1:30pm, Wed in Room 9
Summary
Notes
capabilities to improve the adoption and usability of Andriod Apps. This session covers best practices for tracking apps on mobile, TV and other devices. We'll also show you how to gain actionable insights from new tracking and reporting capabilities.
Speakers
Jim Cotugno
Jim is a software engineer on the Google Analytics Tracking Team with a focus on mobile tracking. Prior to Google, Jim developed operating systems, web applications and mobile applications.
Nick Mihailovski
Nick Mihailovski oversees Developer Relations for Google Analytics, working with clients to develop innovative measurement solutions. He has 6 years prior experience in the Web Analytics industry.
Philip Mui
4:25

How it works

Basics

- Single .APK
- Custom **layouts**, **drawable**, **style**, **dimension** resources for **`-xlarge-v11`**
  - Gingerbread tablets will use ‘normal’, phone layout
- Fragment compatibility library

How it works

Activities

- Different types of activities:
  - Universal: **`HomeActivity`**
  - Phone-only: **`SessionsActivity`** (single fragment)
  - Tablet-only: **`SessionsMultiPaneActivity`**

- Activity helpers
  - **`ActivityHelper`** **`BaseSinglePaneActivity`**
  - **`ActivityHelperHoneycomb`** **`BaseMultiPaneActivity`**

How it works

Action bar

- Custom action bar (**`ActionBarCompat`**) for phones
  - Still using **`res/menu/`**
  - Action bar and buttons are just styled Views

```
public void onPostCreate(Bundle savedInstanceState) {
    SimpleMenu menu = new SimpleMenu(mActivity);
    onCreatePanelMenu(Window.FEATURE_OPTIONS_PANEL, menu);
    for (int i = 0; i < menu.size(); i++) {
        MenuItem item = menu.getItem(i);
        addActionButtonCompatFromMenuItem(item);
    }
}
```

Tab Drawables — Phones and Tablets, Nine-patches

- `drawable-hdpi`

- `drawable-xlarge-mdpi-v11`

Body Content Dimensions — Phones and Tablets

Phones: `values/dimens.xml`

```
<dimen name="body_padding_large"> 10dp </dimen>
<dimen name="text_size_xlarge">   18sp </dimen>
<dimen name="speaker_image_size"> 36dp </dimen>
```

Tablets: `values-xlarge-v11/dimens.xml`

```
<dimen name="body_padding_large"> 20dp </dimen>
<dimen name="text_size_xlarge">   32sp </dimen>
<dimen name="speaker_image_size"> 64dp </dimen>
```

Get the code:

http://code.google.com/p/iosched

What We Covered

1. Introduction to tablets

2. Honeycomb visual design

3. Tablet UI patterns + Honeycomb framework features
   - Interaction design
   - Implementation

4. Case study — Google I/O 2011 App

Ask questions

http://goo.gl/mod/zdyR

Give feedback

http://goo.gl/4dTQp

#io2011 #Android


Google™
11
I/O