building for multiple screens

big android bbq 2013

Paul Oremland
http://tech.infospace.com
https://github.com/poremland
http://www.linkedin.com/in/pauloremland

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	
small	9.5%						9.5%
normal	0.1%	15.7%		33.6%	23.1%	7.1%	79.6%
large	0.6%	3.4%	1.2%	0.4%	0.5%		6.1%
xlarge		4.4%		0.3%	0.1%		4.8%
	10.2%	23.5%	1.2%	34.3%	23.7%	7.1%	

* as of sept 4 2013 source: http://developer.android.com/about/dashboards/index.html

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

building for multiple screens

- plan your applications workflow and navigation
  - understand how to approach layouts
  - understand what built-in android idioms are available
- create your layouts to take advantage of your target screens
- implement your application workflow using the built-in idioms

approaching layout

plan your applications workflow and navigation

approaching layout

two main approaches to laying out your content to support multiple screens

- layouts that can scale naturally
- layouts that target specific screen sizes/orientations

approaching layout

two main approaches to laying out your content to support multiple screens

- layouts that can scale naturally
- layouts that target specific screen sizes/orientations

approaching layout

two main approaches to laying out your content to support multiple screens

- layouts that can scale naturally
- layouts that target specific screen sizes/orientations

layouts that can scale naturally

- have a UI that naturally fills the screen
- have the same screen to screen workflow and navigation
- scale using a combination of techniques

layouts that can scale naturally

- have a UI that naturally fills the screen
- have the same screen to screen workflow and navigation
- scale using a combination of techniques

layouts that can scale naturally

- have a UI that naturally fills the screen
- have the same screen to screen workflow and navigation
- scale using a combination of techniques

scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

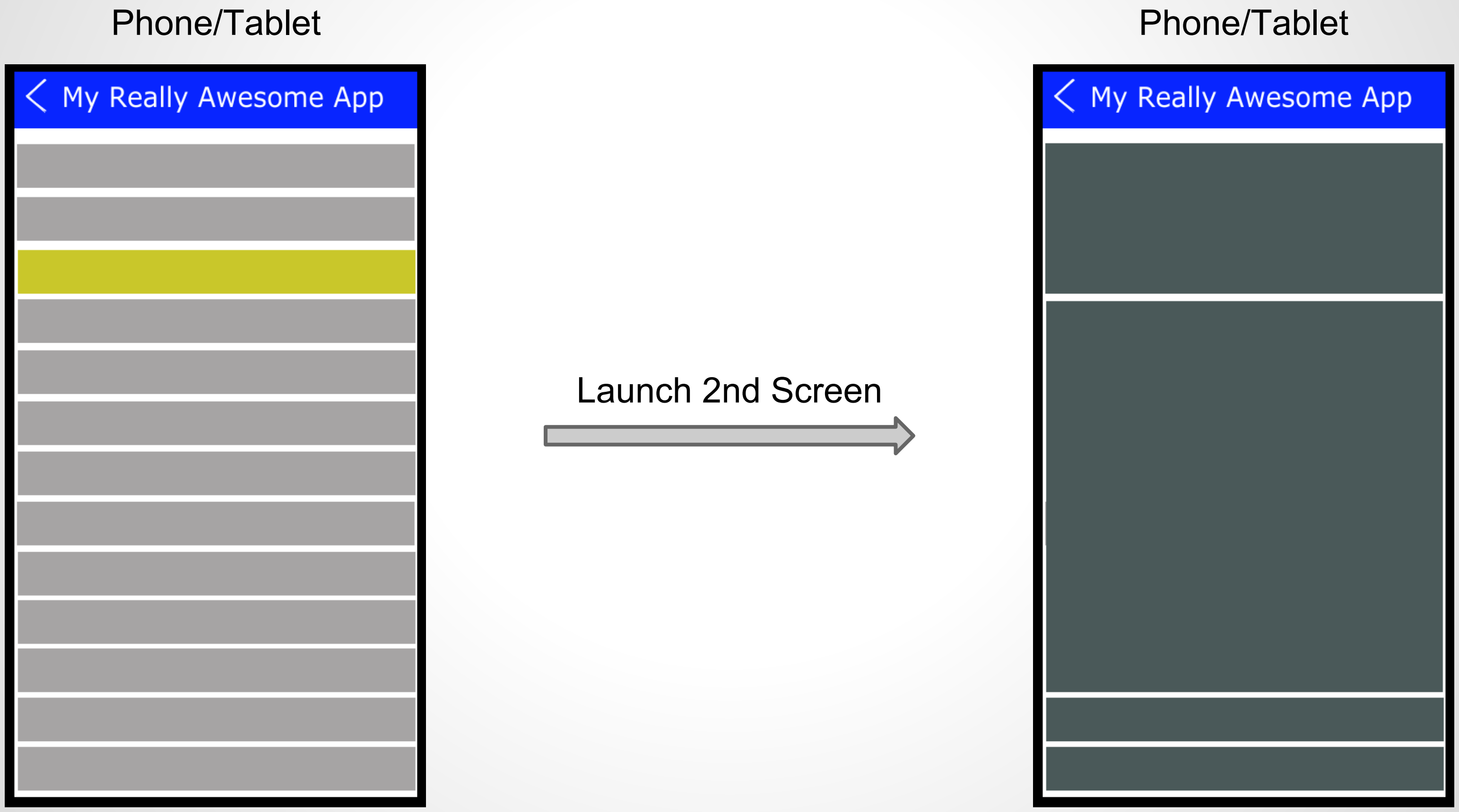
scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

scaling techniques

- use of match_parent and wrap_content in layout_width & layout_height
- use layout_weight and weightSum to keep the size ratio the same between views
- use relative layouts to align left/right/top/bottom regardless of size
- use density independent pixels (DP/SP instead of px)
- create bitmaps for each supported screen density (mdpi, hdpi, etc)
- use nine-patched images to naturally adapt image resources to the size/orientation of your screen

layouts that can scale naturally



layouts that target specific screen sizes/orientations

- allow for a different workflow and UI for different screen sizes
- use qualifiers to select the correct layouts/resources
- aided by the use of fragments from the support library
- allow for better use of screen space to show more content or show more depth in content
- require a higher level of workflow/user interaction coordination between an Activity and its Fragments

layouts that target specific screen sizes/orientations

- allow for a different workflow and UI for different screen sizes
- use qualifiers to select the correct layouts/resources
- aided by the use of fragments from the support library
- allow for better use of screen space to show more content or show more depth in content
- require a higher level of workflow/user interaction coordination between an Activity and its Fragments

layouts that target specific screen sizes/orientations

- allow for a different workflow and UI for different screen sizes
- use qualifiers to select the correct layouts/resources
- aided by the use of fragments from the support library
- allow for better use of screen space to show more content or show more depth in content
- require a higher level of workflow/user interaction coordination between an Activity and its Fragments

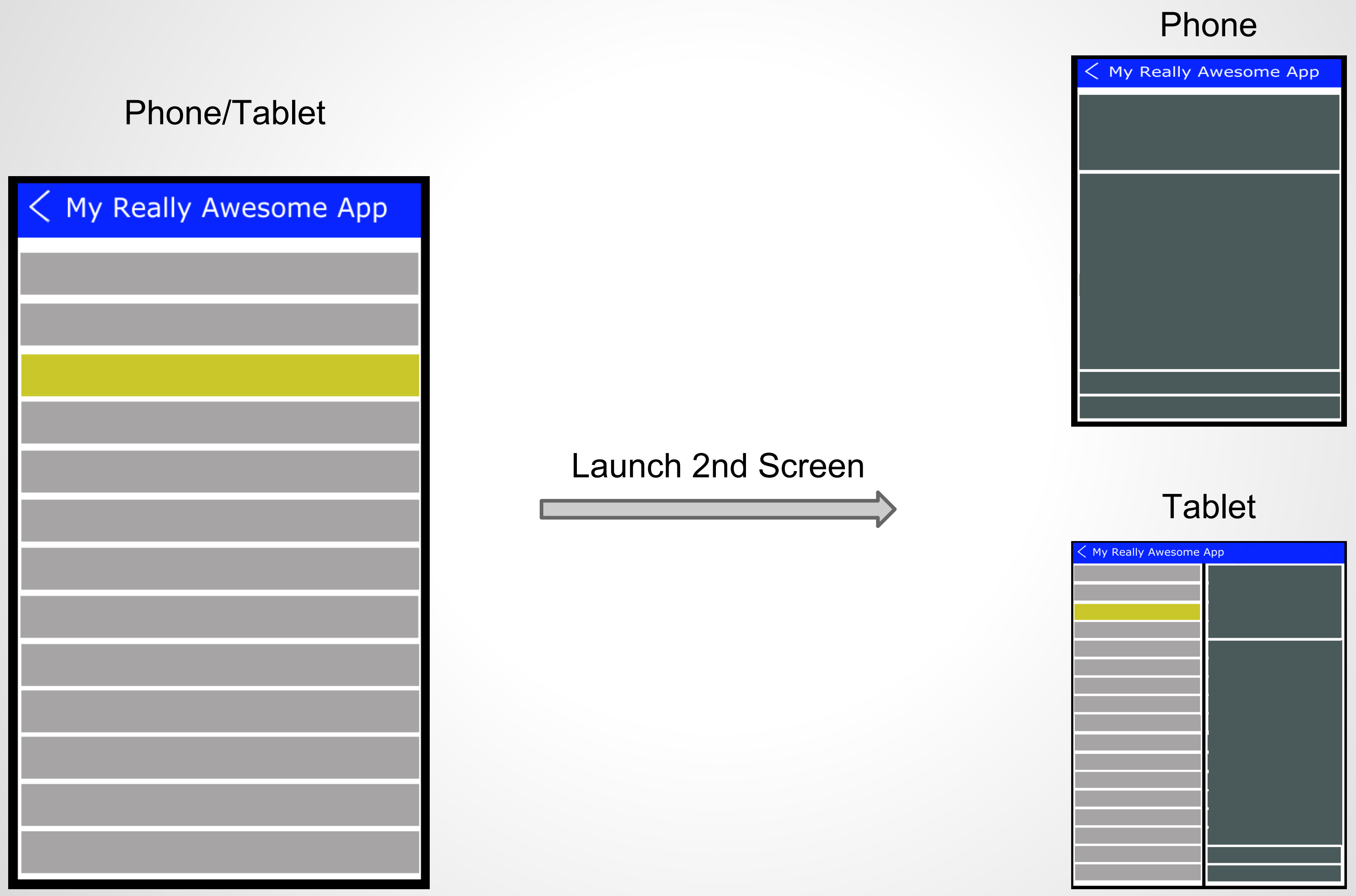
layouts that target specific screen sizes/orientations

- allow for a different workflow and UI for different screen sizes
- use qualifiers to select the correct layouts/resources
- aided by the use of fragments from the support library
- allow for better use of screen space to show more content or show more depth in content
- require a higher level of workflow/user interaction coordination between an Activity and its Fragments

layouts that target specific screen sizes/orientations

- allow for a different workflow and UI for different screen sizes
- use qualifiers to select the correct layouts/resources
- aided by the use of fragments from the support library
- allow for better use of screen space to show more content or show more depth in content
- require a higher level of workflow/user interaction coordination between an Activity and its Fragments

layouts that target specific screen sizes/orientations



which one do I use?

- approach 1 and 2 can be used in combination with each other; they are NOT mutually exclusive

- approach 1 works better to display rich content (web pages, videos, etc…)

- approach 2 works better when drilling into content where the larger the screen size the more natural it is to “show” the drilling down

which one do I use?

- approach 1 and 2 can be used in combination with each other; they are NOT mutually exclusive

- approach 1 works better to display rich content (web pages, videos, etc…)

- approach 2 works better when drilling into content where the larger the screen size the more natural it is to “show” the drilling down

which one do I use?

- approach 1 and 2 can be used in combination with each other; they are NOT mutually exclusive

- approach 1 works better to display rich content (web pages, videos, etc…)

- approach 2 works better when drilling into content where the larger the screen size the more natural it is to “show” the drilling down

built-in idioms

plan your applications workflow and navigation

built-in idioms
configuration qualifiers

Allow you to provide resources targeted towards specific screen configurations

- size
  - physical screen size as measured diagonally
- density
  - quantity of pixels in a physical area of the screen
- orientation
  - orientation of the device from the perspective of the user
- aspect ratio
  - ratio of a devices width to height

built-in idioms
configuration qualifiers

Allow you to provide resources targeted towards specific screen configurations

- size
  - physical screen size as measured diagonally
- density
  - quantity of pixels in a physical area of the screen
- orientation
  - orientation of the device from the perspective of the user
- aspect ratio
  - ratio of a devices width to height

built-in idioms
configuration qualifiers

Allow you to provide resources targeted towards specific screen configurations

- size
  - physical screen size as measured diagonally
- density
  - quantity of pixels in a physical area of the screen
- orientation
  - orientation of the device from the perspective of the user
- aspect ratio
  - ratio of a devices width to height

built-in idioms
configuration qualifiers

Allow you to provide resources targeted towards specific screen configurations

- size
  - physical screen size as measured diagonally
- density
  - quantity of pixels in a physical area of the screen
- orientation
  - orientation of the device from the perspective of the user
- aspect ratio
  - ratio of a devices width to height

configuration qualifiers
size

small	screens that are similar in size to low-density QVGA	min: 320x426 dp
normal	screens that are similar in size to medium-density HVGA	min: 320x470 dp
large	screens that are similar in size to medium-density VGA	min: 480x640 dp
xlarge	screens **much larger** than medium-density HVGA	min: 720x960 dp

Default is a non-tablet layout

res/values/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">false</bool>
      …
    <bool name="some_other_bool">true</bool>
</resources>
```

For “large” screens we set this to true

res/values-large/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">true</bool>
</resources>
```

configuration qualifiers
size

small	screens that are similar in size to low-density QVGA	min: 320x426 dp
normal	screens that are similar in size to medium-density HVGA	min: 320x470 dp
large	screens that are similar in size to medium-density VGA	min: 480x640 dp
xlarge	screens **much larger** than medium-density HVGA	min: 720x960 dp

Default is a non-tablet layout

For “large” screens we set this to true

res/values/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">false</bool>
      ...
    <bool name="some_other_bool">true</bool>
</resources>
```

res/values-large/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">true</bool>
</resources>
```

configuration qualifiers
size

small	screens that are similar in size to low-density QVGA	min: 320x426 dp
normal	screens that are similar in size to medium-density HVGA	min: 320x470 dp
large	screens that are similar in size to medium-density VGA	min: 480x640 dp
xlarge	screens **much larger** than medium-density HVGA	min: 720x960 dp

Default is a non-tablet layout

res/values/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">false</bool>
      ...
    <bool name="some_other_bool">true</bool>
</resources>
```

For “large” screens we set this to true

res/values-large/bools.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="isTablet">true</bool>
</resources>
```

configuration qualifiers
size: smallest width and minimum width/height

- Allows targeting of screens based on a smallest width (specified in DP), a minimum width, or a minimum height.
- Easier to target “tablet” screen sizes
    - layout-sw600dp allows you to serve the “phone” layout to larger screen phones, like 5” ones, but the “tablet” layout to 7” tablets
    - layout-large matches most 5” phones as well as most 7” tablets
- Only available on Android 3.2 and above!

Provide a default layout

Provide a different layout for 600dp or greater devices

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   ...
</LinearLayout>
```

res/layout-sw600dp/mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <Fragment ... />
   <Fragment ... />
</LinearLayout>
```

configuration qualifiers

size: smallest width and minimum width/height

- Allows targeting of screens based on a smallest width (specified in DP), a minimum width, or a minimum height.
- Easier to target "tablet" screen sizes
  - layout-sw600dp allows you to serve the "phone" layout to larger screen phones, like 5" ones, but the "tablet" layout to 7" tablets
  - layout-large matches most 5" phones as well as most 7" tablets
- Only available on Android 3.2 and above!

Provide a default layout

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   ...
</LinearLayout>
```

Provide a different layout for 600dp or greater devices

res/layout-sw600dp/mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <Fragment ... />
   <Fragment ... />
</LinearLayout>
```

configuration qualifiers

size: smallest width and minimum width/height

- Allows targeting of screens based on a smallest width (specified in DP), a minimum width, or a minimum height.
- Easier to target “tablet” screen sizes
  - layout-sw600dp allows you to serve the “phone” layout to larger screen phones, like 5” ones, but the “tablet” layout to 7” tablets
  - layout-large matches most 5” phones as well as most 7” tablets
- Only available on Android 3.2 and above!

Provide a default layout

Provide a different layout for 600dp or greater devices

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   ...
</LinearLayout>
```

res/layout-sw600dp/mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <Fragment ... />
   <Fragment ... />
</LinearLayout>
```

configuration qualifiers

size: smallest width and minimum width/height

- Allows targeting of screens based on a smallest width (specified in DP), a minimum width, or a minimum height.
- Easier to target "tablet" screen sizes
    - layout-sw600dp allows you to serve the "phone" layout to larger screen phones, like 5" ones, but the "tablet" layout to 7" tablets
    - layout-large matches most 5" phones as well as most 7" tablets
- Only available on Android 3.2 and above!

Provide a default layout

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
  ...
</LinearLayout>
```

Provide a different layout for 600dp or greater devices

res/layout-sw600dp/mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
  <Fragment ... />
  <Fragment ... />
</LinearLayout>
```

configuration qualifiers

size: smallest width and minimum width/height

- Allows targeting of screens based on a smallest width (specified in DP), a minimum width, or a minimum height.
- Easier to target “tablet” screen sizes
  - layout-sw600dp allows you to serve the “phone” layout to larger screen phones, like 5” ones, but the “tablet” layout to 7” tablets
  - layout-large matches most 5” phones as well as most 7” tablets
- Only available on Android 3.2 and above!

Provide a default layout

Provide a different layout for 600dp or greater devices

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   ...
</LinearLayout>
```

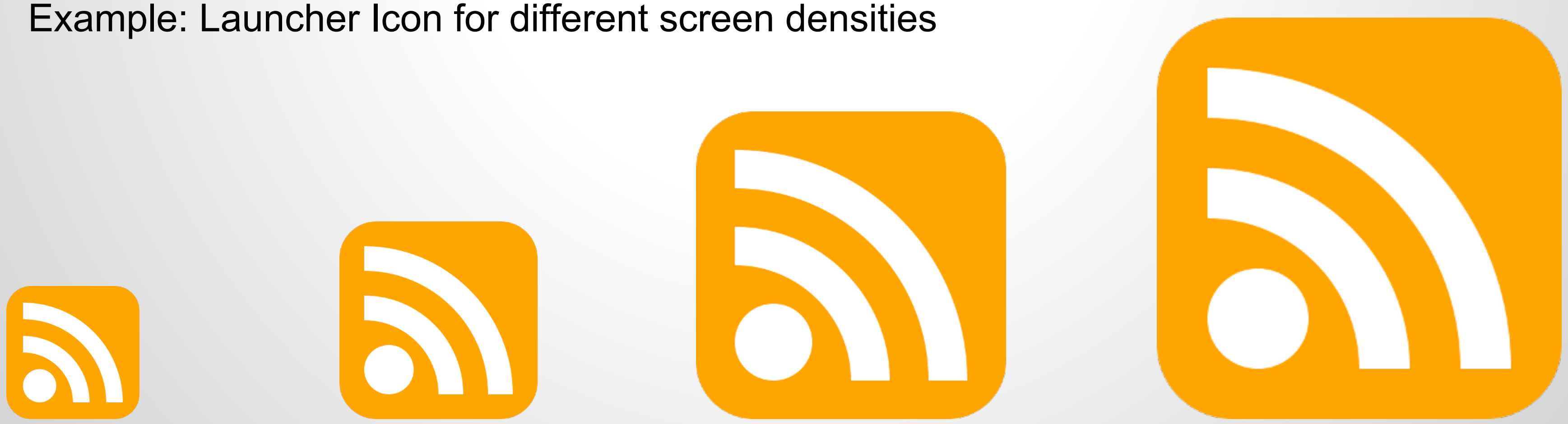
res/layout-sw600dp/mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <Fragment ... />
   <Fragment ... />
</LinearLayout>
```

configuration qualifiers
density

ldpi	resources for low density devices	~120 dpi
mdpi	resources for medium density devices	~160 dpi
hdpi	resources for high density devices	~240 dpi
xhdpi	resources for extra high density devices	~320 dpi
nodpi	resources for all densities	
tvdpi	resources for screens somewhere between mdpi and hdpi	approx 213 dpi

Example: Launcher Icon for different screen densities



res/drawable-ldpi/... res/drawable-mdpi/... res/drawable-hdpi/... res/drawable-xhdpi/...

configuration qualifiers
orientation

land	resources for screens in landscape orientation
port	resources for screens in portrait orientation

res/layout-port/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    ...
</LinearLayout>
```

res/layout-land/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    ...
</LinearLayout>
```

configuration qualifiers
orientation

land	resources for screens in landscape orientation
port	resources for screens in portrait orientation

res/layout-port/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    ...

</LinearLayout>
```

res/layout-land/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    ...

</LinearLayout>
```

configuration qualifiers
orientation

land	resources for screens in landscape orientation
port	resources for screens in portrait orientation

res/layout-port/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    ...

</LinearLayout>
```

res/layout-land/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http:
//schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    ...

</LinearLayout>
```

configuration qualifiers
aspect ratio qualifiers

long	resources for screens significantly taller than the baseline configuration
notlong	resources for screens with an aspect ratio closer to the baseline

res/layout-long/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <!-- Fragment with some content -->
   <Fragment ... />
   <!-- Fragment with some additional meta
data -->
   <Fragment ... />
</LinearLayout>
```

res/layout-notlong/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
   <!-- Fragment with some content -->
   <Fragment ... />
   <!-- We don’t show the Fragment with
additional meta data -->
</LinearLayout>
```

configuration qualifiers
aspect ratio qualifiers

long	resources for screens significantly taller than the baseline configuration
notlong	resources for screens with an aspect ratio closer to the baseline

res/layout-long/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
    <!-- Fragment with some content -->
    <Fragment ... />
    <!-- Fragment with some additional meta
data -->
    <Fragment ... />
</LinearLayout>
```

res/layout-notlong/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
    <!-- Fragment with some content -->
    <Fragment ... />
    <!-- We don’t show the Fragment with
additional meta data -->
</LinearLayout>
```

configuration qualifiers
aspect ratio qualifiers

long	resources for screens significantly taller than the baseline configuration
notlong	resources for screens with an aspect ratio closer to the baseline

res/layout-long/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
    <!-- Fragment with some content -->
    <Fragment ... />
    <!-- Fragment with some additional meta
data -->
    <Fragment ... />
</LinearLayout>
```

res/layout-notlong/main.xml

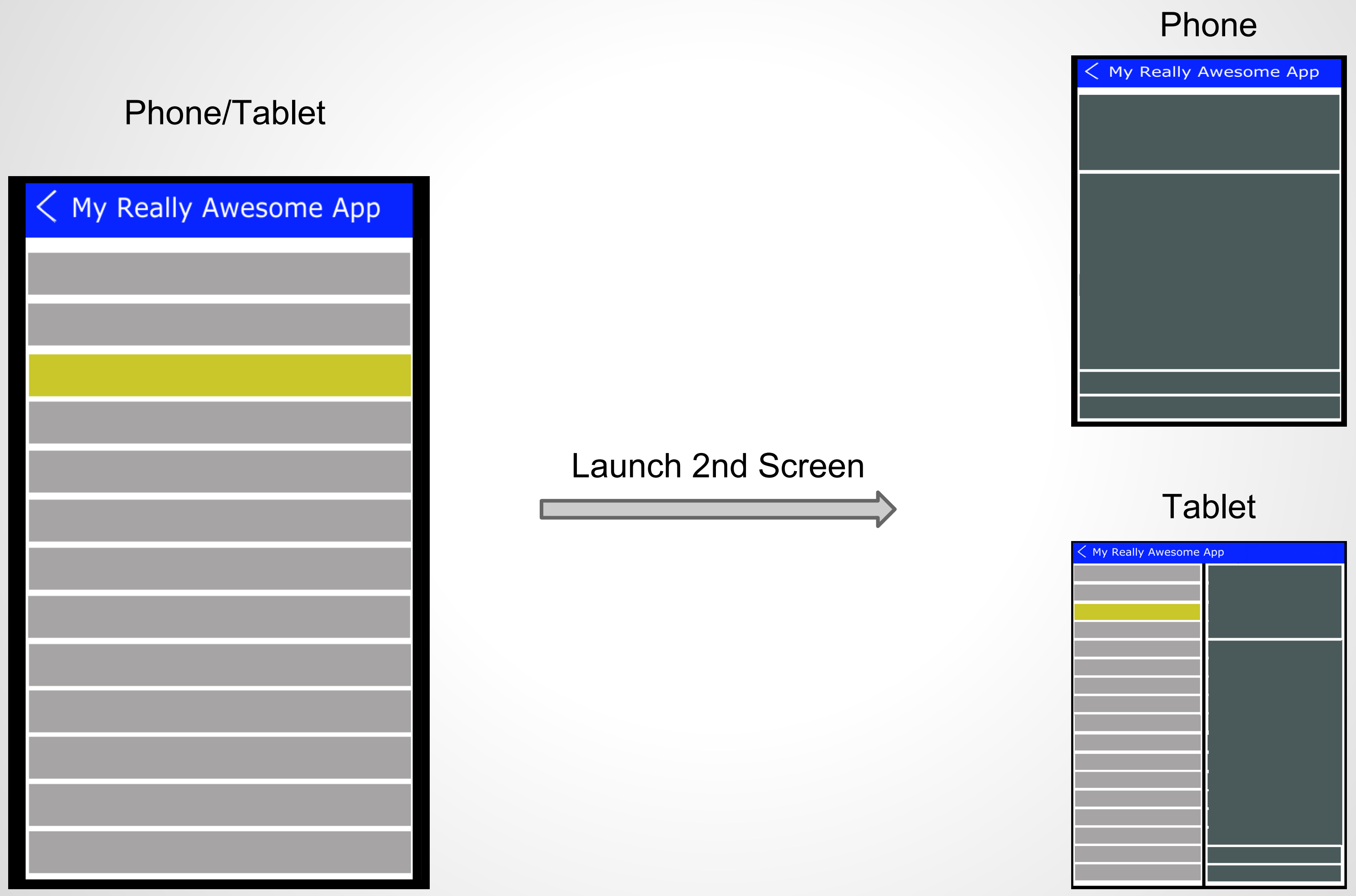
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout… >
    <!-- Fragment with some content -->
    <Fragment ... />
    <!-- We don’t show the Fragment with
additional meta data -->
</LinearLayout>
```

creating your layouts

responding to size

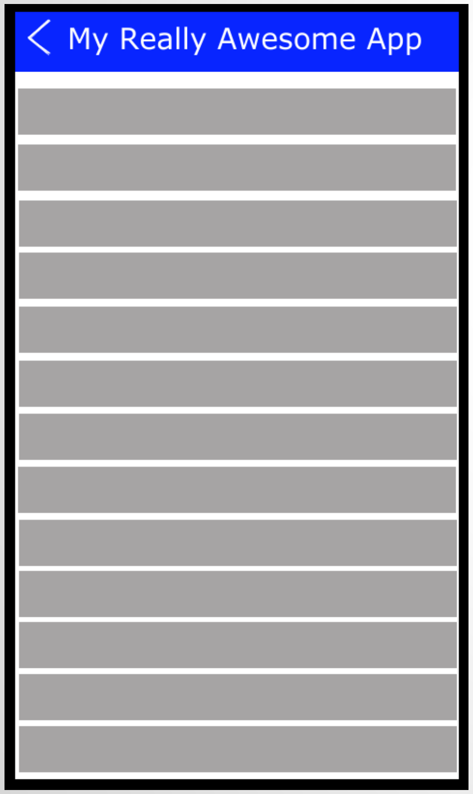
creating your layouts

responding to size
layouts



responding to size
default layout

Phone



res/layout/main.xml

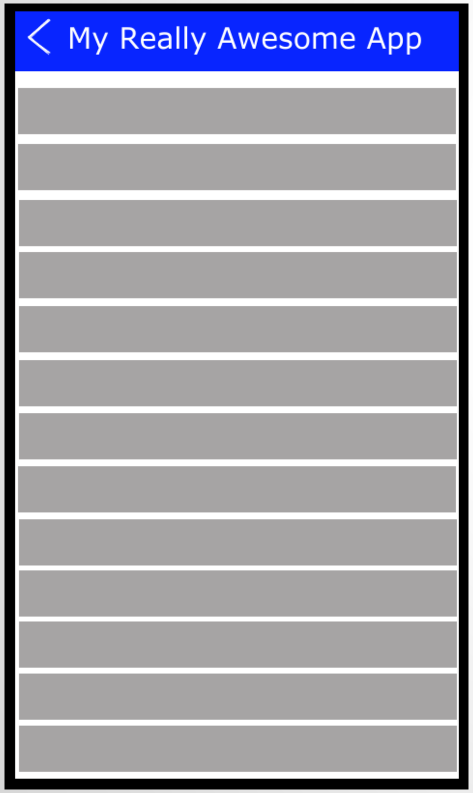
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- No Fragments defined -->

</LinearLayout>
```

responding to size
default layout

res/layout/main.xml

default layout used if device does not match any provided qualifier.

Phone

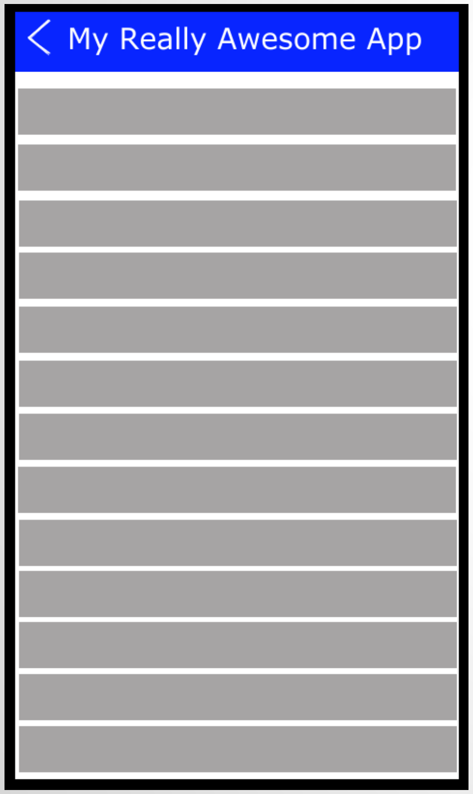


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- No Fragments defined -->

</LinearLayout>
```

responding to size
default layout

res/layout/main.xml

Phone



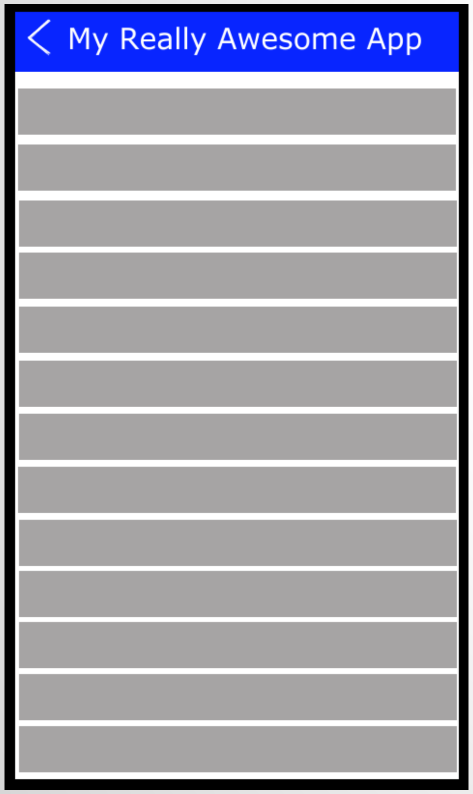
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- No Fragments defined -->

</LinearLayout>
```

We need to give the container of our fragments an id so that, in the default mode, we can programmatically add/replace fragments in it.

responding to size
default layout

res/layout/main.xml

Phone



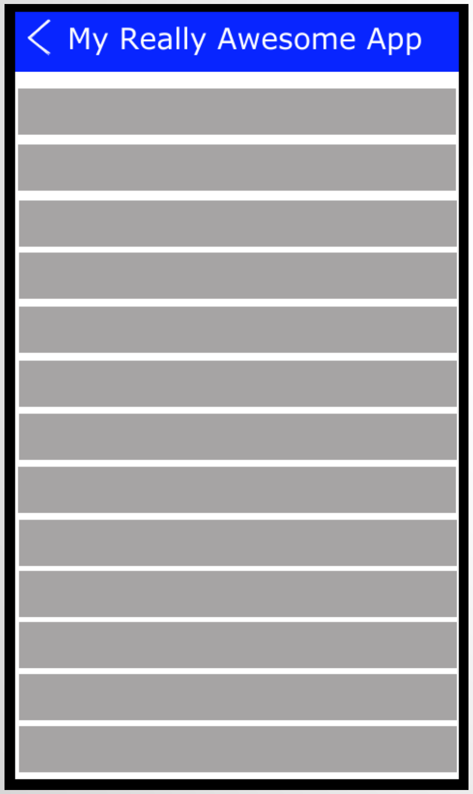
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
   android:layout_height="match_parent">

    <!-- No Fragments defined -->

</LinearLayout>
```

We want this container to fill the entire screen.

responding to size
default layout

res/layout/main.xml

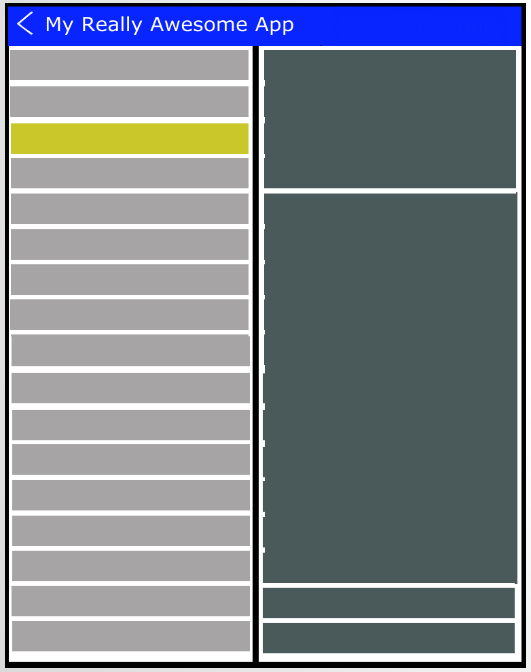
Phone



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <!-- No Fragments defined -->

</LinearLayout>
```

We'll programmatically add/replace fragments in this container as the default.

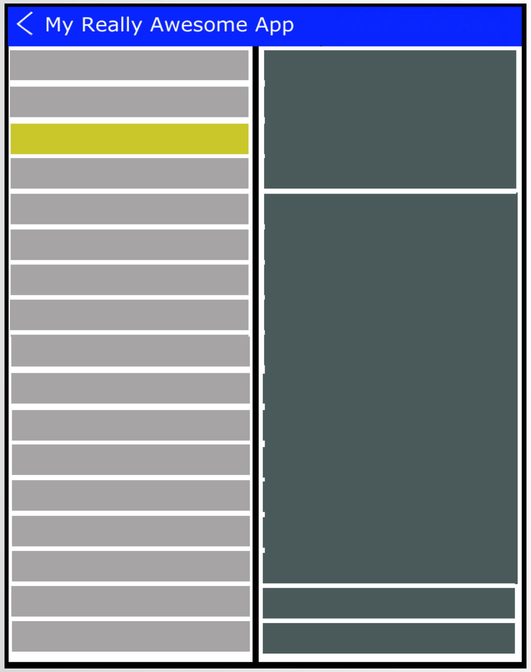
responding to size
large layout



res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

responding to size
large layout



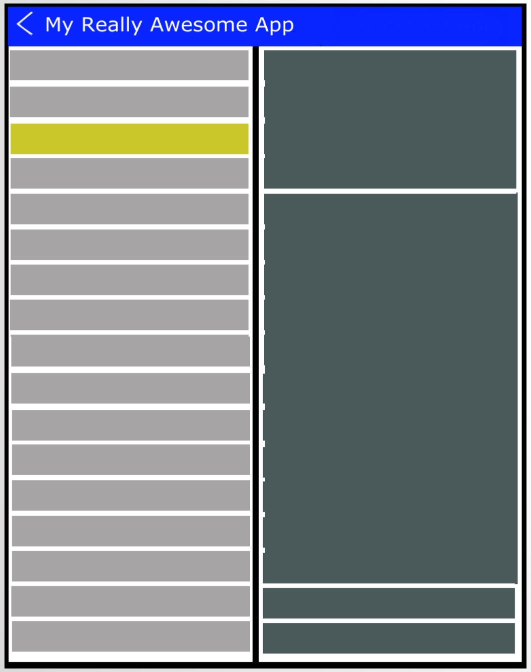
res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

Layout used on "large" devices. If not developing with backwards compatibility for < Android 3.2 smallest width qualifiers are recommended.

responding to size

large layout

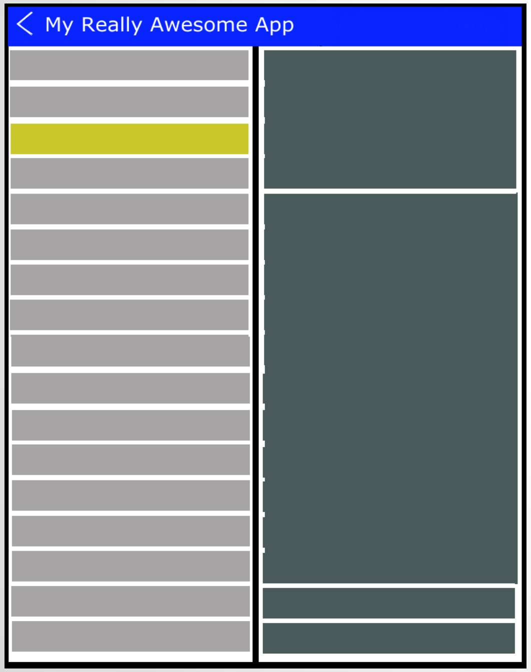


res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

We need to use the same container id that we used in our default layout so that we can programmatically load/update the fragments content.

responding to size
large layout



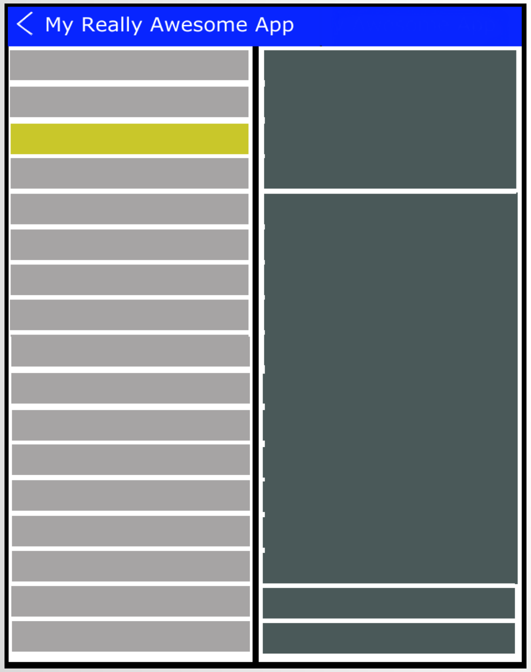
res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

As with the default layout, we want this container to fill the entire screen.

responding to size

large layout

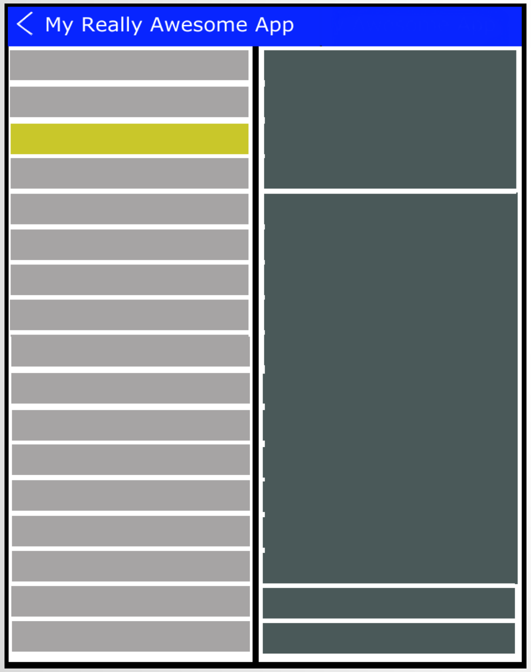


res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

With a width of 0dp and a weight of one this fragment will fill the entire screen until the other fragment has content.

responding to size
large layout



res/layout-large/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment
        android:name="your.package.FirstFragment"
        android:id="@+id/first_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <Fragment
        android:name="your.package.SecondFragment"
        android:id="@+id/second_fragment"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

With a width of 0dp and no weight this fragment will not take up any space until it has content.

implementing your application workflow

displaying your content

implementing your application workflow

displaying your content
activity start up

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    if(savedInstanceState == null)
    {
        this.displayFragment(R.id.first_fragment);
    }
}
```

displaying your content
activity start up

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    if(savedInstanceState == null)
    {
        this.displayFragment(R.id.first_fragment);
    }
}
```

Phone: res/layout/main.xml

Tablet: res/layout-large/main.xml

displaying your content
activity start up

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    if(savedInstanceState == null)
    {
        this.displayFragment(R.id.first_fragment);
    }
}
```

if there is no saved state that already exists we will need to start from scratch. Otherwise the saved state can be restored using the onRestoreInstanceState method.

displaying your content
activity start up

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    if(savedInstanceState == null)
    {
        this.displayFragment(R.id.first_fragment);
    }
}
```

Phone: add/replace fragment

Tablet: update existing fragment

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content

displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

First, we need to know if the fragment is already being displayed in the layout.

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

Phone: fragment is null

Tablet: fragment is not null

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

Since the fragment does not exist in the layout, we first need to create the fragment.

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

Then we add the fragment to the back stack, which will push it on the screen.

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content

displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

The fragment already exists in the layout. So all we need to do is tell it to load it's data.

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content
checking for an existing fragment

```
public void getFragment(int id)
{
    FragmentManager manager = getSupportFragmentManager();
    Fragment fragment = manager.findFragmentByTag(Integer.toString(id));

    if(fragment == null)
    {
        fragment = manager.findFragmentById(id);
    }
    return fragment;
}
```

displaying your content
checking for an existing fragment

You obtain a fragment from the layout/backstack using the FragmentManager

```
public void getFragment(int id)
{
    FragmentManager manager = getSupportFragmentManager();
    Fragment fragment = manager.findFragmentByTag(Integer.toString(id));

    if(fragment == null)
    {
        fragment = manager.findFragmentById(id);
    }
    return fragment;
}
```

displaying your content
checking for an existing fragment

```
public void getFragment(int id)
{
    FragmentManager manager = getSupportFragmentManager();
    Fragment fragment = manager.findFragmentByTag(Integer.toString(id));

    if(fragment == null)
    {
        fragment = manager.findFragmentById(id);
    }
    return fragment;
}
```

Fragments can be referenced by tag and/or id. It doesn't matter in which order you check as long as you check both.

displaying your content
checking for an existing fragment

```
public void getFragment(int id)
{
    FragmentManager manager = getSupportFragmentManager();
    Fragment fragment = manager.findFragmentByTag(Integer.toString(id));

    if(fragment == null)
    {
        fragment = manager.findFragmentById(id);
    }
    return fragment;
}
```

If we're here that means we didn't find the fragment by it's tag. So now we need to try to find the fragment by it's id.

displaying your content
checking for an existing fragment

```
public void getFragment(int id)
{
    FragmentManager manager = getSupportFragmentManager();
    Fragment fragment = manager.findFragmentByTag(Integer.toString(id));

    if(fragment == null)
    {
        fragment = manager.findFragmentById(id);
    }
    return fragment;
}
```

If the fragment exists in the layout/backstack this will not be null.

displaying your content

displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

create an instance of the fragment and perform any necessary setup.

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content
adding a fragment to the back stack

```
public void addFragmentToBackStack(
        Fragment fragment, int id) {
        FragmentManager manager = getSupportFragmentManager();
        if(fragment != null && manager != null && !fragment.isInLayout()) {
                FragmentTransaction transaction = manager.beginTransaction();
                // perform any custom fragment workflows here.
                // like hiding/showing fragments declared in the layout
                // using transaction.hide(...) or transaction.show(../)
                transaction.replace(R.id.fragment_container,
                        fragment,
                        Integer.toString(id));
                transaction.addToBackStack(null);
                transaction.commit();
        }
}
```

displaying your content
adding a fragment to the back stack

```
public void addFragmentToBackStack(
	Fragment fragment, int id) {
	FragmentManager manager = getSupportFragmentManager();
	if(fragment != null && manager != null && !fragment.isInLayout()) {
		FragmentTransaction transaction = manager.beginTransaction();
		// perform any custom fragment workflows here.
		// like hiding/showing fragments declared in the layout
		// using transaction.hide(...) or transaction.show(../)
		transaction.replace(R.id.fragment_container,
			fragment,
			Integer.toString(id));
		transaction.addToBackStack(null);
		transaction.commit();
	}
}
```

displaying your content
adding a fragment to the back stack

Sanity checks to make sure what we're trying to do can actually be accomplished.

```
public void addFragmentToBackStack(
        Fragment fragment, int id) {
        FragmentManager manager = getSupportFragmentManager();
        if(fragment != null && manager != null && !fragment.isInLayout()) {
                FragmentTransaction transaction = manager.beginTransaction();
                // perform any custom fragment workflows here.
                // like hiding/showing fragments declared in the layout
                // using transaction.hide(...) or transaction.show(../)
                transaction.replace(R.id.fragment_container,
                        fragment,
                        Integer.toString(id));
                transaction.addToBackStack(null);
                transaction.commit();
        }
}
```

displaying your content

adding a fragment to the back stack

```
public void addFragmentToBackStack(
        Fragment fragment, int id) {
        FragmentManager manager = getSupportFragmentManager();
        if(fragment != null && manager != null && !fragment.isInLayout()) {
                FragmentTransaction transaction = manager.beginTransaction();
                // perform any custom fragment workflows here.
                // like hiding/showing fragments declared in the layout
                // using transaction.hide(...) or transaction.show(../)
                transaction.replace(R.id.fragment_container,
                        fragment,
                        Integer.toString(id));
                transaction.addToBackStack(null);
                transaction.commit();
        }
}
```

Start a series of edit operations on fragments.

displaying your content

adding a fragment to the back stack

```
public void addFragmentToBackStack(
        Fragment fragment, int id) {
        FragmentManager manager = getSupportFragmentManager();
        if(fragment != null && manager != null && !fragment.isInLayout()) {
                FragmentTransaction transaction = manager.beginTransaction();
                // perform any custom fragment workflows here.
                // like hiding/showing fragments declared in the layout
                // using transaction.hide(...) or transaction.show(../)
                transaction.replace(R.id.fragment_container,
                        fragment,
                        Integer.toString(id));
                transaction.addToBackStack(null);
                transaction.commit();
        }
}
```

Equivalent of calling transaction.remove for all current fragments in the container and add for the given fragment. DOES NOT remove fragments that are defined in the layout

displaying your content
adding a fragment to the back stack

```
public void addFragmentToBackStack(
    Fragment fragment, int id) {
    FragmentManager manager = getSupportFragmentManager();
    if(fragment != null && manager != null && !fragment.isInLayout()) {
        FragmentTransaction transaction = manager.beginTransaction();
        // perform any custom fragment workflows here.
        // like hiding/showing fragments declared in the layout
        // using transaction.hide(...) or transaction.show(../)
        transaction.replace(R.id.fragment_container,
            fragment,
            Integer.toString(id));
        transaction.addToBackStack(null);
        transaction.commit();
    }
}
```

Adding to the back stack allows us to let Android to remember this transaction and revert it when the back stack is popped.

displaying your content
adding a fragment to the back stack

```
public void addFragmentToBackStack(
    Fragment fragment, int id) {
    FragmentManager manager = getSupportFragmentManager();
    if(fragment != null && manager != null && !fragment.isInLayout()) {
        FragmentTransaction transaction = manager.beginTransaction();
        // perform any custom fragment workflows here.
        // like hiding/showing fragments declared in the layout
        // using transaction.hide(...) or transaction.show(../)
        transaction.replace(R.id.fragment_container,
            fragment,
            Integer.toString(id));
        transaction.addToBackStack(null);
        transaction.commit();
    }
}
```

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

displaying your content
displaying a fragment

```
public void displayFragment(int id)
{
    Fragment fragment = this.getFragment(id);

    if(fragment == null)
    {
        fragment = this.createFragment(id);
        this.addFragmentToBackStack(fragment, id);
    }
    else
    {
        fragment.loadData();
    }
}
```

going back

displaying your content
going back

- hard/soft back button pressed
- action bar home button pressed

displaying your content
going back

- hard/soft back button pressed
- action bar home button pressed

displaying your content
going back

- restore fragment visibility for fragments declared in the layout (i.e. tablets) that have been hidden to push a non-declared fragment into the layout
- set action bar home button (set as up/enabled)
- pop back stack using fragment manager

displaying your content
going back

- restore fragment visibility for fragments declared in the layout (i.e. tablets) that have been hidden to push a non-declared fragment into the layout
- set action bar home button (set as up/enabled)
- pop back stack using fragment manager

displaying your content
going back

- restore fragment visibility for fragments declared in the layout (i.e. tablets) that have been hidden to push a non-declared fragment into the layout
- set action bar home button (set as up/enabled)
- pop back stack using fragment manager

displaying your content

popping the back stack

```
private boolean popBackStack()
{
        boolean wasBackStackPopped = false;
        FragmentManager manager = getSupportFragmentManager();
        if(manager != null)
        {
                if(this.shouldPopBackStack(manager))
                {
                        manager.popBackStack();
                        wasBackStackPopped = true;
                }
                this.invalidateOptionsMenu();
        }

        this.updateHomeButton();
        return wasBackStackPopped;
}
```

displaying your content

popping the back stack

```
private boolean popBackStack()
{
    boolean wasBackStackPopped = false;
    FragmentManager manager = getSupportFragmentManager();
    if(manager != null)
    {
        if(this.shouldPopBackStack(manager))
        {
            manager.popBackStack();
            wasBackStackPopped = true;
        }
        this.invalidateOptionsMenu();
    }

    this.updateHomeButton();
    return wasBackStackPopped;
}
```

caller can decide if action is necessary based on whether the backstack was popped or not.

displaying your content

popping the back stack

```
private boolean popBackStack()
{
	boolean wasBackStackPopped = false;
	FragmentManager manager = getSupportFragmentManager();
	if(manager != null)
	{
		if(this.shouldPopBackStack(manager))
		{
			manager.popBackStack();
			wasBackStackPopped = true;
		}
		this.invalidateOptionsMenu();
	}

	this.updateHomeButton();
	return wasBackStackPopped;
}
```

decide if you're already on the initial view.

displaying your content
popping the back stack

```
private boolean popBackStack()
{
    boolean wasBackStackPopped = false;
    FragmentManager manager = getSupportFragmentManager();
    if(manager != null)
    {
        if(this.shouldPopBackStack(manager))
        {
            manager.popBackStack();
            wasBackStackPopped = true;
        }
        this.invalidateOptionsMenu();
    }

    this.updateHomeButton();
    return wasBackStackPopped;
}
```

if using an action bar make sure the options menu properly reflects the currently displayed menu options.

displaying your content
popping the back stack

```
private boolean popBackStack()
{
    boolean wasBackStackPopped = false;
    FragmentManager manager = getSupportFragmentManager();
    if(manager != null)
    {
        if(this.shouldPopBackStack(manager))
        {
            manager.popBackStack();
            wasBackStackPopped = true;
        }
        this.invalidateOptionsMenu();
    }

    this.updateHomeButton();
    return wasBackStackPopped;
}
```

enable/disable “home as up” for the action bar home button.

putting it all together

putting it all together
building for multiple screens

- use scaling techniques previously outlined to make sure your layouts scale and look wonderful on all screen sizes and resolutions
- identify opportunities for customized layouts that take advantage of all available screen real estate within your apps natural workflow
- use configuration qualifiers to handle size specific workflow when pushing and popping fragments

putting it all together
building for multiple screens

- use scaling techniques previously outlined to make sure your layouts scale and look wonderful on all screen sizes and resolutions
- identify opportunities for customized layouts that take advantage of all available screen real estate within your apps natural workflow
- use configuration qualifiers to handle size specific workflow when pushing and popping fragments

putting it all together
building for multiple screens

- use scaling techniques previously outlined to make sure your layouts scale and look wonderful on all screen sizes and resolutions
- identify opportunities for customized layouts that take advantage of all available screen real estate within your apps natural workflow
- use configuration qualifiers to handle size specific workflow when pushing and popping fragments

Resources

- Paul Oremland’s GitHub:
  https://github.com/poremland
- InfoSpace Technology Blog:
  http://tech.infospace.com/
- Supporting Multiple Screen Sizes:
  http://developer.android.com/training/multiscreen/screensizes.html
- Supporting Different Densities:
  http://developer.android.com/training/multiscreen/screendensities.html
- Using Configuration Qualifiers:
  http://developer.android.com/guide/practices/screens_support.html#qualifiers
- Implementing Adaptive UI Flows:
  http://developer.android.com/training/multiscreen/adaptui.html
- 9 patch tool:
  http://developer.android.com/tools/help/draw9patch.html

questions?