Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Smart Home Control
Android Styles Making It Nice

Prof.Dr.Ing.Birgit Wendholt

STYLE =

Geräte und Bildschirmtypen

Themes

Iconographie

Metriken

Farbe

Typographie

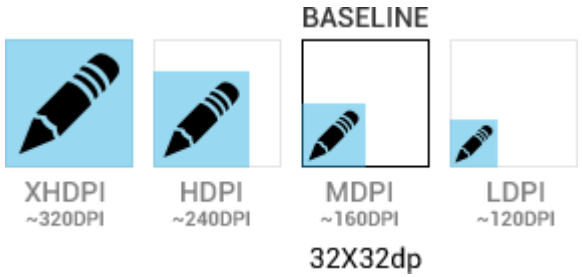
Geräte und Bildschirmtypen

Ziele

1. **Flexibilität**: Anpassen der Layouts für verschiedene Höhen und Breiten

2. **Optimierung**: Zusammengesetzte Views auf größeren Bildschirmen

3. **Assets for all:** Resourcen für verschiedene Bildschirmauflösungen



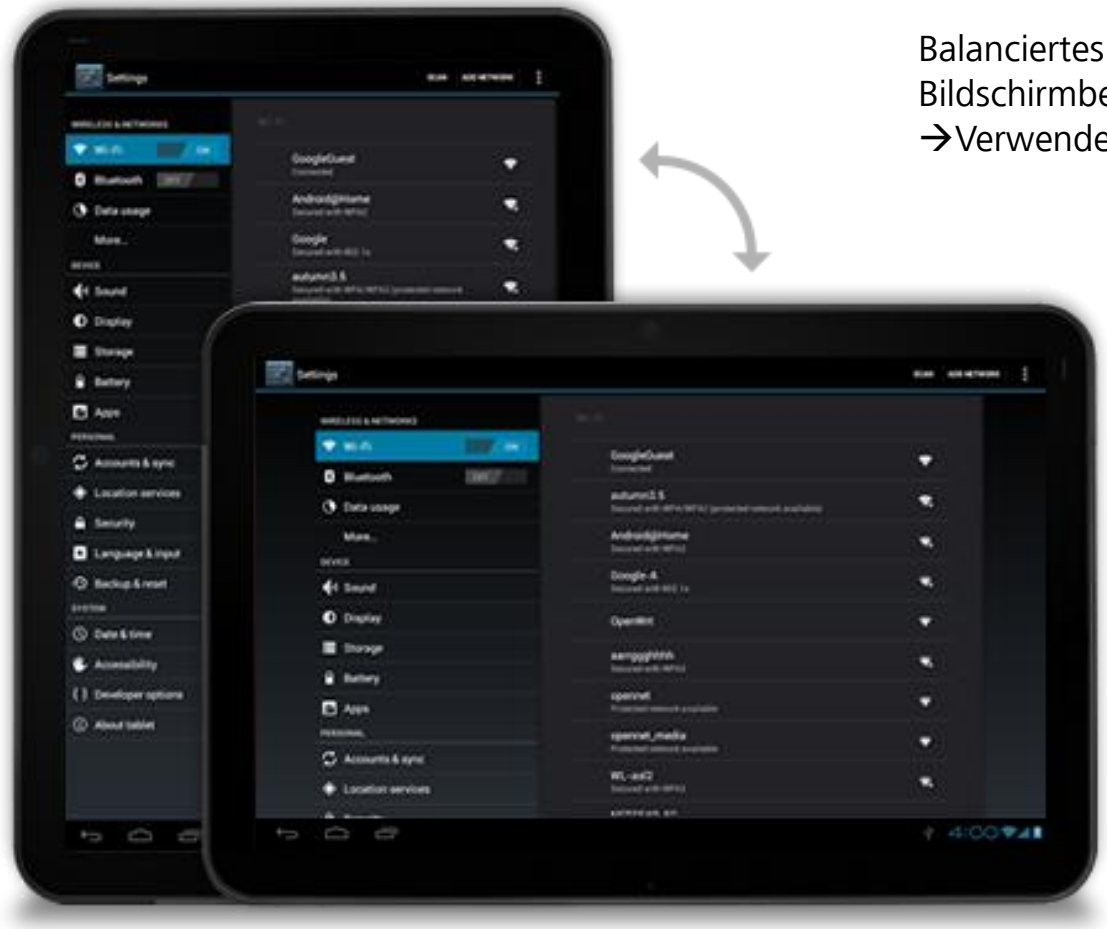
Lösungen

1. **Flexibilität**:
   a. Entwurf für mehrere Bildschirme
   b. UI Entwicklung mit Fragments

2. **Optimierung**: Multi-pane Layout mit Anpassung an die Orientierung :
   a. Dehnen und Stauchen
   b. Stapeln
   c. Expandieren und Einklappen
   d. Zeigen / Verbergen

3. **Asset for all**: IconSets

Multi-pane Layout Dehnen und Stauchen



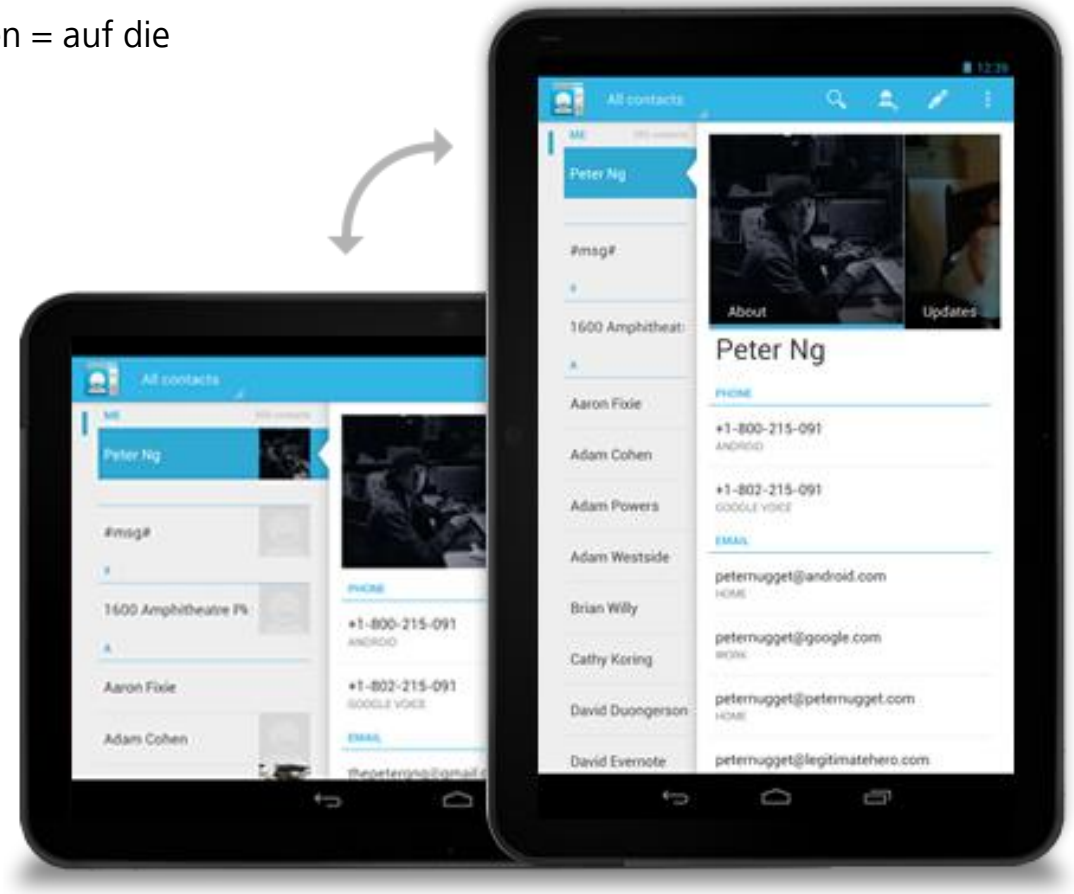
Balanciertes Verhältnis der beiden Bildschirmbereiche für beide Orientierungen
→Verwenden des layout_weight Attributs

Multi-pane Layout Stapeln

Ändern der Anordnung der Bildschirmbereiche

Multi-pane Layout Expandieren und Kollabieren

Linken Bildschirmbereich kollabieren = auf die wesentlichen Inhalte einschränken



Multi-pane Layout Zeigen und Verbergen

Zeigen und Verbergen, wenn keine der Lösungen den Inhalt des linken Bereichs hinreichend anzeigt
→ Verwenden von 2 Aktivitäten plus UP-Navigation

Entwurf für verschiedene Bildschirme

http://developer.android.com/training/multiscreen/index.html

- Unterstützung für verschieden Bildschirmgrößen
  http://developer.android.com/training/multiscreen/screensizes.html

- Unterstützung für verschiedene Pixeldichten
  http://developer.android.com/training/multiscreen/screendensities.html

- Implementieren adaptiver UI-Flows (→
  http://developer.android.com/training/multiscreen/adaptui.html)

Unterstützung für verschiedene Bildschirmgrößen

1. relative Größen für Breite und Höhe: ***layout_width***, ***layout_height***: ***wrap_content*** und ***match_parent***

2. ***RelativeLayout***: Ausrichten relativ zu anderen Komponenten

3. Size Qualifier

4. Smallest Width Qualifier

5. Layout Aliases

6. Orientation Qualifiers (→ http://developer.android.com/guide/practices/screens_support.html#qualifiers)

RelativeLayout


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dp"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:layout_below="@id/entry"
        android:text="Cancel" />
</RelativeLayout>
```

Size Qualifier

- **Ziel** → Konfiguration alternativer Layouts für unterschiedliche Bildschirmgrößen (Teilmenge der Configuration Qualifier)

① Konfiguration für kleine Bildschirme von SmartPhones in ***res/layout/main.xml***

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment android:id="@+id/headlines"
              android:layout_height="fill_parent"
              android:name="com.example.android.newsreader.HeadlinesFragment"
              android:layout_width="match_parent" />
</LinearLayout>
```

Size Qualifier

- **Ziel** → Konfiguration alternativer Layouts für unterschiedliche Bildschirmgrößen (Teilmenge der Configuration Qualifier)

② Konfiguration für große Bildschirme von Tablets in ***res/layout-large/main.xml***

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <fragment android:id="@+id/headlines"
              android:layout_height="fill_parent"
              android:name="com.example.android.newsreader.HeadlinesFragment"
              android:layout_width="400dp"
              android:layout_marginRight="10dp"/>
    <fragment android:id="@+id/article"
              android:layout_height="fill_parent"
              android:name="com.example.android.newsreader.ArticleFragment"
              android:layout_width="fill_parent" />
</LinearLayout>
```

Smallest Width Qualifier

http://developer.android.com/guide/practices/screens_support.html#qualifiers

- enthalten Layouts für einen Bildschirm mit einer Mindestdichte (hier 600 dp)
- andere gängige Qualifier: 320dp smartPhone, 720dp: 10" tablet

③ Konfiguration in ***res/layout-sw600dp/main.xml***

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <fragment android:id="@+id/headlines"
              android:layout_height="fill_parent"
              android:name="com.example.android.newsreader.HeadlinesFragment"
              android:layout_width="400dp"
              android:layout_marginRight="10dp"/>
    <fragment android:id="@+id/article"
              android:layout_height="fill_parent"
              android:name="com.example.android.newsreader.ArticleFragment"
              android:layout_width="fill_parent" />
</LinearLayout>
```

Layout Aliase

http://developer.android.com/guide/practices/screens_support.html#qualifiers

- mehrere Layout Konfigurationen
  - → Dubletten (hier s2600dp und large) durch Layout Aliase vermeiden
  - → Extrahieren der Gemeinsamkeiten der 2-Fenster-Sicht in z.B. ***main_twopanes.xml*** (liegt in ***res/layout***)
  - → Referenzieren in den spezifischen Subverzeichnisse in den ***main.xml*** Layout-Dateien

② ***res/layout-large/main.xml***

```
<resources>
    <item name="main" type="layout">@layout/main_twopanes</item>
</resources>
```

③ ***res/layout-sw600dp/main.xml***

```
<resources>
    <item name="main" type="layout">@layout/main_twopanes</item>
</resources>
```

Orientation Qualifier

http://developer.android.com/guide/practices/screens_support.html#qualifiers

- ***NewsReader*** Applikation reagiert auf verschiedene Bildschirmgrößen und Orientierungen:
  - **small screen, portrait:** einzelnes Fenster mit Logo
  - **small screen, landscape:** einzelnes Fenster mit Logo
  - **7" tablet, portrait:** einzelnes Fenster mit Action Bar
  - **7" tablet, landscape:** zwei Fenster breit mit Action Bar
  - **10" tablet, portrait:** zwei Fenster schmal mit Action Bar
  - **10" tablet, landscape:** zwei Fenster breit mit Action Bar
- Dazu verwendet werden die folgenden Layout Dateien
  - res/layout/onepane.xml
  - res/layout/onepane_with_bar.xml
  - res/layout/twopanes.xml
  - res/layout/twopanes_narrow.xml
- **TODO**: Mapping des korrekten Layouts auf eine der 6 Geräte Konfiguration

Orientation Qualifier
Mapping von Layouts und Gerätekonfiguration

- Über die Layout Alias Technik

① ***res/values/layouts.xml***

```
<resources>
    <item name="main_layout" type="layout">@layout/onepane_with_bar</item>
    <bool name="has_two_panes">false</bool>
</resources>
```

② ***res/values-sw600dp-land/layouts.xml***

```
<resources>
    <item name="main_layout" type="layout">@layout/twopanes</item>
    <bool name="has_two_panes">true</bool>
</resources>
```

③ ***res/values-sw600dp-port/layouts.xml***

```
<resources>
    <item name="main_layout" type="layout">@layout/onepane</item>
    <bool name="has_two_panes">false</bool>
</resources>
```

Unterstützung für verschiedene Pixeldichten

http://developer.android.com/training/multiscreen/screendensities.html

- Größen in Dichte-unabhängigen Pixelwerten angeben (dp für Dimensionen und sp für Text)

```
<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/clickme"
    android:layout_marginTop="20dp" />
```

```
<TextView android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="20sp" />
```

Unterstützung für verschiedene Pixeldichten

http://developer.android.com/training/multiscreen/screendensities.html

- Bitmaps für unterschiedliche Pixeldichten anlegen
  - xhdpi: 2.0
  - hdpi: 1.5
  - mdpi: 1.0 (baseline)
  - ldpi: 0.75

  - 200x200 Image für xhdpi Geräte
  - 150x150 für hdpi
  - 100x100 für mdpi
  - 75x75 Image für ldpi Geräte

```
res/
  drawable-xhdpi/
      awesomeimage.png
  drawable-hdpi/
      awesomeimage.png
  drawable-mdpi/
      awesomeimage.png
  drawable-ldpi/
      awesomeimage.png
```

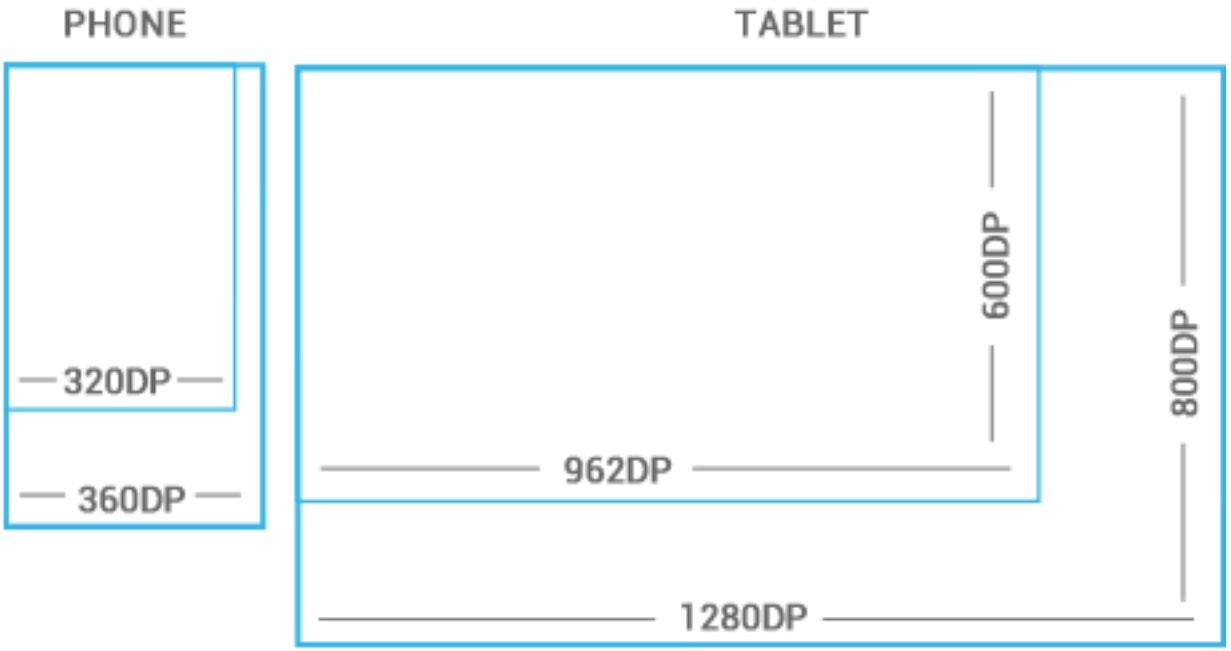
Implementieren adaptiver UI-Flows

http://developer.android.com/training/multiscreen/adaptui.html

→ Folien zu v4-b-1 LP Controller Navigation Templates

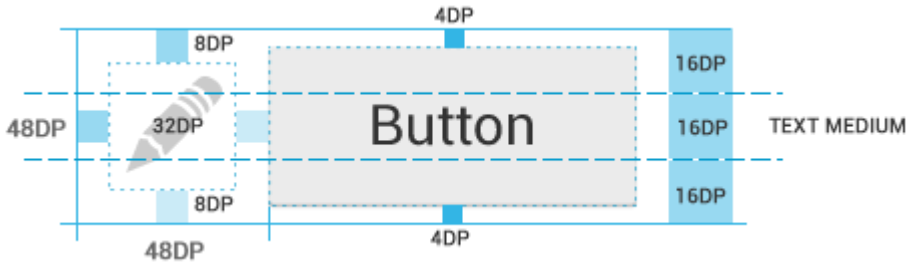
Metriken

- Bildschirmdichte dots per inch (dpi)
- LDPI, MDPI, HDPI, XDPI
- dp- steht für Dichte-unabhängige Pixel

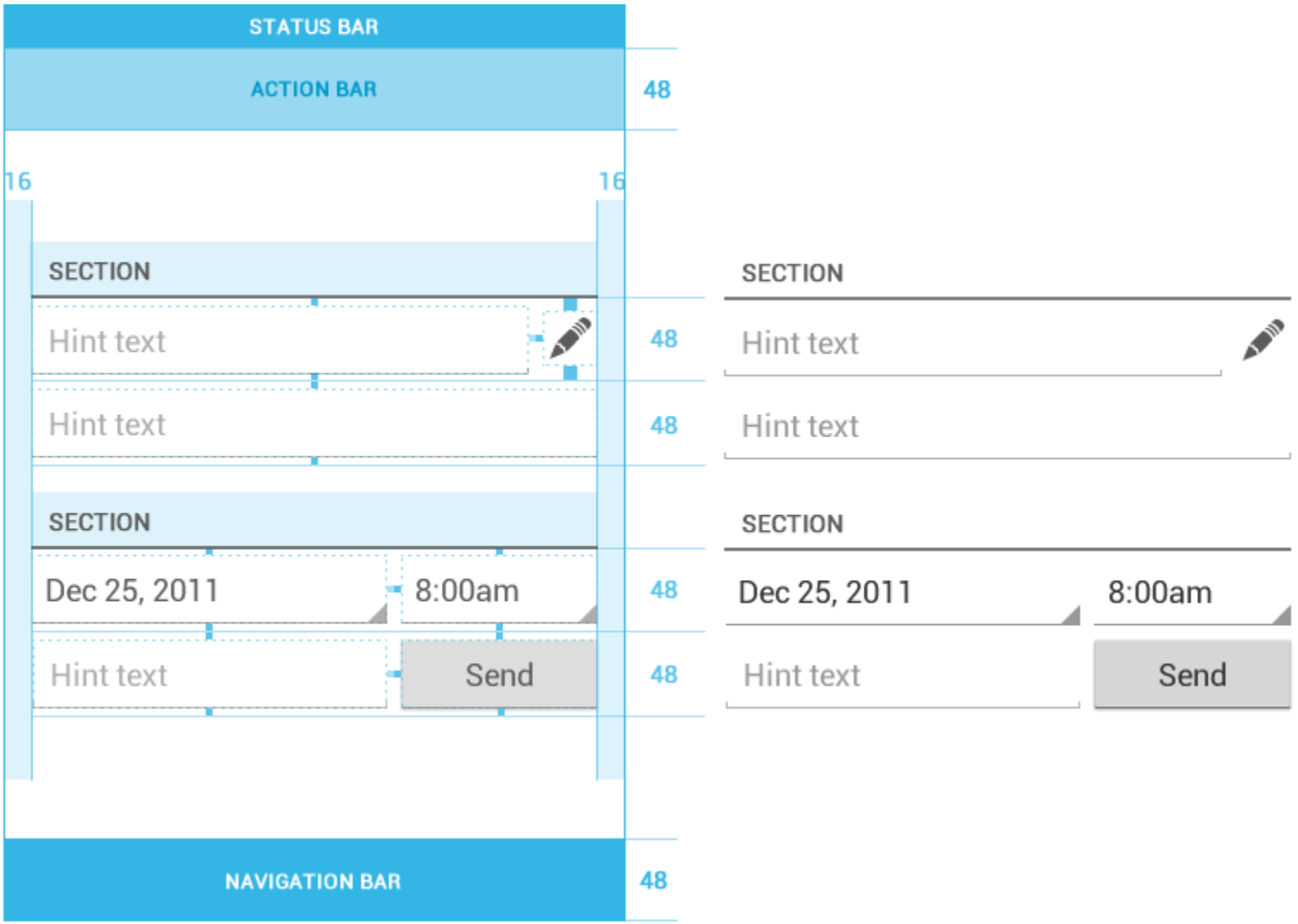


Richtlinien – Touchable UI Komponenten

- Touchable UI Komponenten haben typischerweise eine Größe von 48 dp
- Entspricht einer physikalischen Größe von circa 9 mm (in Bereich der Norm)
- Garantiert:
  - Physikalische Größe ist nie kleiner als 7 mm
  - Kompromiss zwischen Informationsdichte und Handhabbarkeit von Touchables



Abstände zwischen UI Komponenten (8dp)



Themes

- garantieren einheitliche Darstellung einer Applikation
- definieren die visuellen Eigenschaften von Elementen der Benutzerschnittstelle, wie Farbe, Höhe, Abstände, Fonts, Fontgrößen
- vordefinierte Themes:
  - Holo Light
  - Holo Dark
  - Holo Light with dark action bars
- Entwicklung eigener Styles und Themes auf Basis der vordefinierten Themes:
  → http://developer.android.com/guide/topics/ui/themes.html

Entwicklung eigener Styles und Themes

→ http://developer.android.com/guide/topics/ui/themes.html

Style

- eine Sammlung von Eigenschaften, die das Aussehen und das Format einer View bestimmen.
- legt Eigenschaften wie z.B. Höhe, Abstand, Fontfarbe, Fontgröße, Hintergrundfarbe fest
- definiert in einer XML Resource unabhängig von der Layout Resource
- Philosophie ähnlich CSS

Theme

- ein Style, der auf eine Aktivität oder die Applikation angewendet wird, im Gegensatz zu Styles für individuelle Views
- ***CodeFont*** Style als Theme einer Activity → der gesamte Text wird in grünem Monospace Font gerendert
- Attribute für Themes → R.styleable.Theme

Beispiel für Verwendung von styles

- Style bezogene Attribute
  - werden in einer Style-Definition genannt ***CodeFont*** zusammengefasst
  - und über das ***style*** Attribut referenziert

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#00FF00"
    android:typeface="monospace"
    android:text="@string/hello" />
```



```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

Definition von Styles

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

- XML Datei im Verzeichnis res/values/
- Wurzel Knoten: <resources>
- pro Style ein <style> Element
- pro Eigenschaft ein <item> Element
- Namen des <item> Elementes refernzieren eines der Style Attribute
- referenziert in einem XML Layout über ***@style/CodeFont***

Vererbung von Styles

- Android Styles → Verwenden des Attributes ***parent***

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

- eigene Styles → Style von dem geerbt wird, ist der Präfix in der Punktnotation
- Vererbungshierarchien: ***CodeFont.Red.Big***
- Referenz auf den Style ***@style/CodeFont.Red @style/CodeFont.Red.Big***

```
<style name="CodeFont.Red">
    <item name="android:textColor">#FF0000</item>
</style>
```

```
<style name="CodeFont.Red.Big">
    <item name="android:textSize">30sp</item>
</style>
```

Style Eigenschaften

- View-spezifische → definiert in der Klassenreferenz der View (siehe z.B.: TextView XML attributes)

```
<EditText
    android:inputType="number"
    ... />
```

- typische TextView Style Eigenschaft → android:inputType

  1. innerhalb einer EditView
  2. als separater Style

```
<style name="Numbers">
  <item name="android:inputType">number</item>
  ...
</style>
```

- Dokumentation aller Style Attribute in der R.attr Referenz

```
<EditText
    style="@style/Numbers"
    ... />
```

Anwenden von Styles und Themes

- Einzelne Views:
  - ***style*** Attribut der View Definition

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

- Aktivität / Applikation
  - ***android:theme*** Attribut im ***<activity>*** oder ***<application>*** Element des Android Manifests

① `<application android:theme="@style/CustomTheme">`

② `<activity android:theme="@android:style/Theme.Translucent">`

③
```
<color name="custom_theme_color">#b0b0ff</color>
<style name="CustomTheme" parent="android:Theme.Light">
    <item name="android:windowBackground">@color/custom_theme_color</item>
    <item name="android:colorBackground">@color/custom_theme_color</item>
</style>
```

```
<activity android:theme="@style/CustomTheme">
```

Themes für Plattform-Versionen

- neuere Android Versionen → zusätzliche Themes
- Kompatibilität mit älteren Versionen → Themes für Plattformen konfigurieren
- XML Datei für ältere Versionen → res/values
- XML Datei für neuere Versionen (ab API Level 11) → res/values-v11
- Attribute für Themes → R.styleable.Theme
- Konfiguration plattformabhängiger Ressourcen → Providing Resources

① Theme für Android Version < 3.0

```
<style name="LightThemeSelector" parent="android:Theme.Light">
    ...
</style>
```

② Theme für Android Version >= 3.0

```
<style name="LightThemeSelector" parent="android:Theme.Holo.Light">
    ...
</style>
```

Verwenden von Platform Styles und Themes

- Sammlung der Styles und Themes → R.style
- Namen von Styles und Themes → Worte getrennt durch _ (Bsp. ***Theme_NoTitleBar***)
- Verwenden der Styles und Themes in Layouts → Ersetzen des Underscores durch Punkt (Bsp. ***@android:style/Theme.NoTitleBar***)

- Bessere Dokumentation der Styles und Themes („Lernen am Beispiel“)
  - Android Styles (styles.xml)
  - Android Themes (themes.xml)

- Dokumentation der Style-Attribute → R.attr

- Syntax für Styles und Themes in XML → Style Resource

Typographie

- Neue Schrift Roboto für hochauflösende Geräte (Download → Download Roboto)

- Default Farben für Schrifttypen:
  - textColorPrimary / textColorSecondary
  - light themes: textColorPrimaryInverse / textColorSecondaryInverse.

- Schriftgrößen

Text Size Micro	12sp
Text Size Small	14sp
Text Size Medium	18sp
Text Size Large	22sp

Iconographie

- Launcher Icons für die App auf dem All App Screen (48dpx48dp)

- ActionBar Icons (32x32 dp)
  Farbe grau / weiss

- Kleine Kontext Icons (16x16dp) Farbe prominent

- Notification Icons (24x24dp) Farbe weiss