Android 5.0 Lollipop

About Me

Chief Android @ greenrobot

Markus Junginger

- Android Developer '07
- Java Developer '98
- Open Source Author
- GDG Android Munich
- Google+ Profile
- @greenrobot_de (DE)

Android 5.0 - History

- Preview “L” revealed during I/O 2014
  API Level 20
  Preview images
- November 2014
  Final API Level 21
  Firmware Downloads for Nexus devices
  Nexus 6/9 “available”

Biggest Update ever

Material Design & UI Components

Material Design Principles – Quote Poetry

„We challenged ourselves to create a visual language…”

„inspired by the study of paper and ink, yet technologically advanced ”

“These elements do far more than please the eye. They create hierarchy, meaning, and focus.”

Material Design Principles – Quote Poetry

There‘s more:

http://www.google.com/design/spec/material-design/introduction.html

+ Design guidelines

Material Design Principles 1/2

- Colors
  Large areas, suggested color palette
- Images
  More personal & emotional content
- „3D“
  Mostly 2D & 2.5D to give structure.
- Light and Shadow
  Cards and overlays.

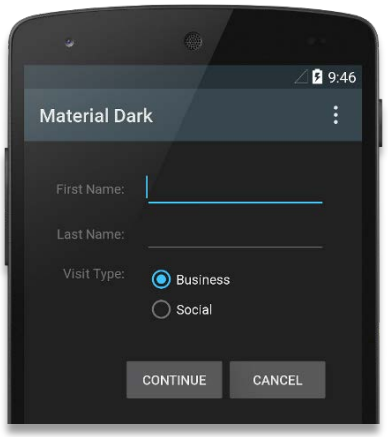
Material Design Principles 2/2

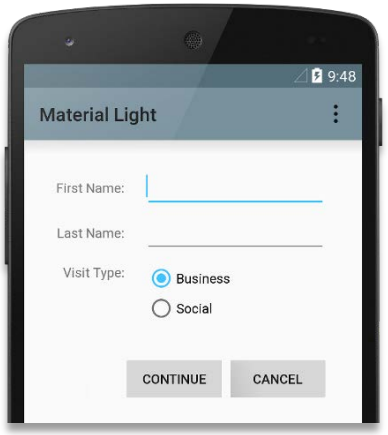
- **Flat**
  No bevels, gradients, effects. Just KISS.
- **Animations**
  Explains interaction. And does boom-wow.
- **Typography**
  Roboto and font style definitions
- **Layout templates**
  Margins, key lines, etc.

Material Design Theme

- Set in AndroidManifest.xml

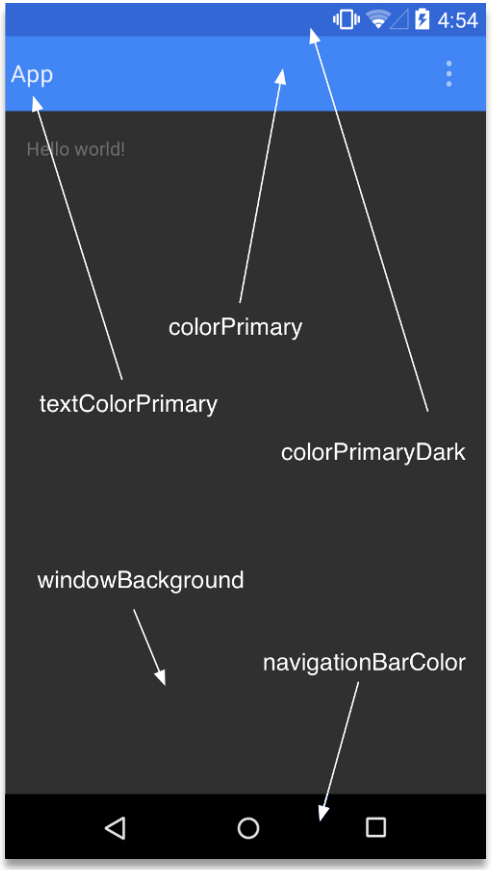
```
@android:style/Theme.Material

@android:style/Theme.Material.Light

@android:style/
Theme.Material.Light.DarkActionBar
```





Material Design Theme – Custom Colors

```
<style name="AppTheme“
    parent="android:Theme.Material">
  <item name="android:colorPrimary">
      #3333cc</item>
  <item name="android:colorPrimaryDark">
      #000099</item>
  <item name="android:colorAccent">
      #999933</item>
</style>
```



Toolbar

- ToolBar is a generalized ActionBar
  More flexible
- setActionBar(toolBar)
  Option menu actions
- Can be placed anywhere in the layout
  For example, in a pop up Fragment
- Toolbar is just another View

Toolbar Example

```
<!-- For example inside some RelativeLayout -->
<android.widget.Toolbar
    android:id="@+id/mytoolbar"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:minHeight="?attr/actionBarSize"
    android:background="?attr/colorPrimary" />

// Inside Activity, after inflating the layout
Toolbar toolbar =
     (Toolbar) findViewById(R.id.mytoolbar);
```

Toolbar – Standalone with Option Menu

```
toolbar.inflateMenu(R.menu.mytoolbar_menu);

toolbar.setOnMenuItemClickListener(
  new Toolbar.OnMenuItemClickListener() {
    @Override
    public boolean onMenuItemClick(MenuItem item) {
      // Do something
    }
  });
```

Shadows and Tints – Less Drawbles!

- Say good bye to shadow.png

```
<View … android:elevation="8dp" />
```

- Change the color of drawables

```
drawable.setTint(color);
// XML: android:tint="#ff00ff"
```

Renderer Thread

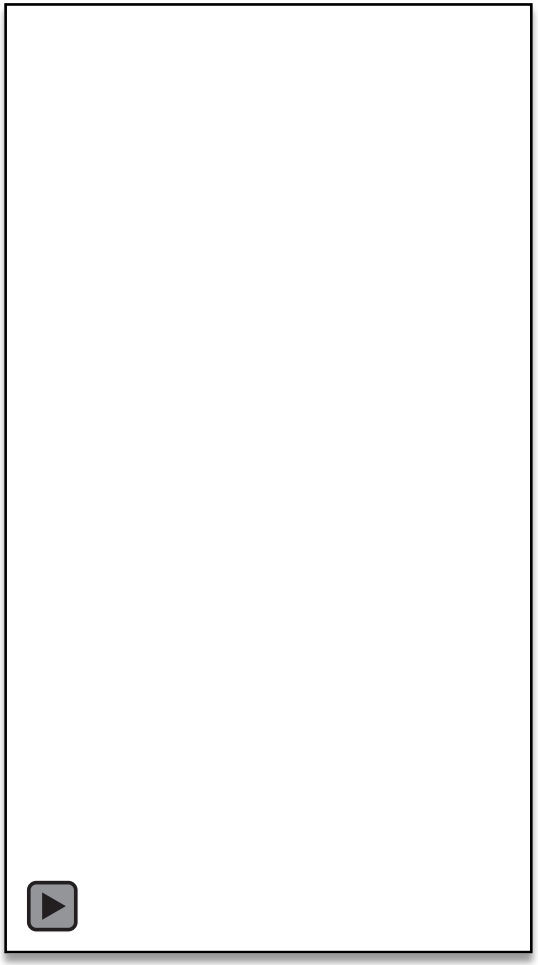
- System thread independent from main thread
- Tasks (main thread can be busy meanwhile)
  Processes DisplayList
  Animations
- DisplayList: List of low level graphic ops
  Used to render GPU accerated UIs (OpenGL)
  Created/invalidated by main thread
- Also used for ripples (touch feedback)

Activity Transitions

- Share „hero“ view elements
  Existing Activity gives views as options
  Animation during Activity transition
- Activities can be in different processes
  Transitions between different apps possible
- Based on Transitions from Android 4.4
  Define start and end situation

Activity Transitions - Example App

- Romain Guy‘s „guest appearance“ @ I/O
- Combination with other animations
- https://github.com/romainguy/google-io-2014
  Apache License, Version 2.0



RecyclerView

- Where does its name come from?
  Recycled views (aka convert views)
- Powerful adapter-backed view
  More flexible than ListView and GridView
- NOT a framework class (!)
  Support library on its own
- Gradle dependency
  com.android.support:recyclerview-v7:21.0.+

RecyclerView - LayoutManager

- LayoutManager places child views
- Must be set in RecyclerView
  `recyclerView.setLayoutManager(lm);`
- Default LayoutManagers
  LinearLayoutManager (vertical & horizontal!)
  StaggeredGridLayoutManager
  GridLayoutManager

RecyclerView.Adapter<ViewHolder>

- RecyclerView.ViewHolder contains View
  Must be sub-classed, avoids findByView(…)
- Implement abstract RecyclerView.Adapter

```
// create new view and its holder (no binding)
ViewHolder onCreateViewHolder(ViewGroup g, int pos)

// bind data values to View
void onBindViewHolder(ViewHolder h, int pos)

int getItemCount()
```

RecyclerView.Adapter – Data notifications

- Problem with notifyDataSetChanged (ListV.)
  Which elements have changed?
  Individual animations are hard to implement
- Fine grained notifications

```
notifyItemChanged(int)
notifyItemInserted(int)
notifyItemRemoved(int)
notifyItemRangeChanged(int, int)
notifyItemRangeInserted(int, int)
notifyItemRangeRemoved(int, int)
```

RecyclerView.Adapter Callbacks

- ViewHolders might use expensive resources
  Bitmaps
- Callbacks useful to release resources
  `onViewAttachedToWindow(VH holder)`
  `onViewDetachedFromWindow(VH holder)`
  `onViewRecycled(VH holder)`

RecyclerView Animations

- Item modifications are animated by default
- Customize with RecyclerView.ItemAnimator

```
// Parameters: ViewHolder + change info
animateAdd(…)
animateChange(…)
animateMove(…)
animateRemove(…)

// Plus some house keeping methods
```

Vector Drawables

- Wouldn‘t be SVG great?
  Instead of bitmaps for many resolutions?
- Not quite there yet
  But…
- android.graphics.drawable.VectorDrawable
  Simple vector definitions (SVG path element)

Vector Drawable – Example XML

```
<vector xmlns:android="…"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData=
                "M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

AnimatedVectorDrawable

- Allows animating group and path properties
- 3 XML files required

  Vector drawable (last slide)

  Animation (res/anim/...)

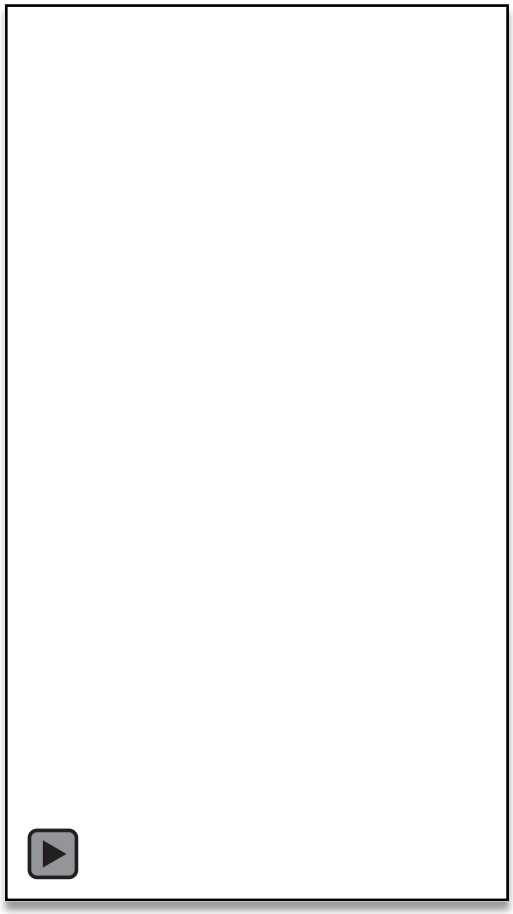
  Animated vector drawable (drawble-nodpi/...)

AnimatedVectorDrawable – avd.xml

```
<animated-vector xmlns:android="…"
      android:drawable="@drawable/vectordrawable" >
    <target
        android:name="rotationGroup"
        android:animation="@anim/rotation" />
    <target
        android:name="v"
        android:animation="@anim/path_morph" />
</animated-vector>
```

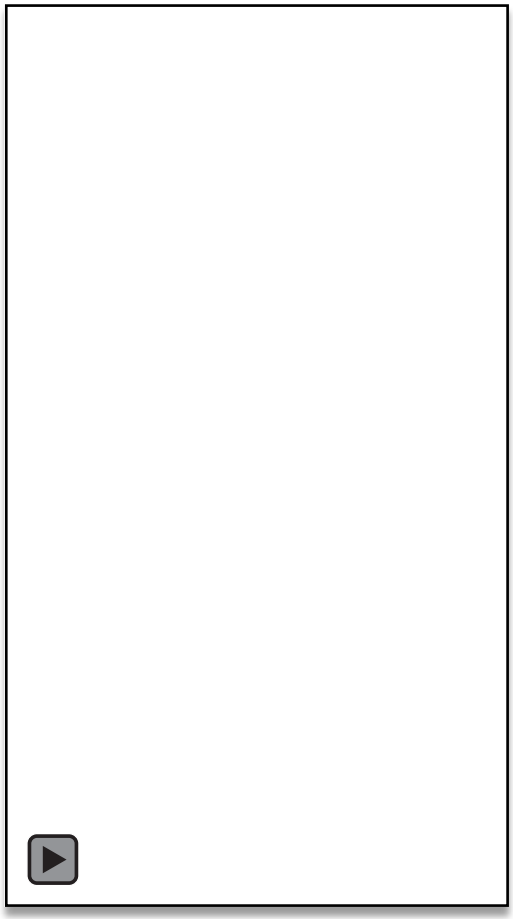
AnimatedVectorDrawable – rotation.xml

```
<objectAnimator xmlns:android="…"
  android:duration="6000"
  android:propertyName="rotation"
  android:valueFrom="0"
  android:valueTo="360"/>
```



AnimatedVectorDrawable – path_morph.xml

```
<objectAnimator xmlns:android="…"
  android:duration="3000"
  android:propertyName="pathData"
  android:valueFrom=
    "M300,70 l 0,-70 70,70 0,0 -70,70z"
  android:valueTo=
    "M300,70 l 0,-70 70,0 0,140 -70,0 z"
  android:valueType="pathType"/>
```



AnimatedVectorDrawable – Delta to Docs

- Start animation

```
Animatable animatable =
    (Animatable) imageView.getDrawable();
animatable.start()
```

- Width and height increased → Quality

```
<vector android:height="320dp"
        android:width="320dp"
```

Android Runtime (ART)

Android VM Basics: Dalvik

- No Java VM
- Dalvik VM
- Java source → .class → DEX
- DEX: Dalvik executable, register-based
- JIT compiler since Android 2.2
- Several optimizations, but…
- Unlike Java, Dalvik never challenged native

ART – The new Android Runtime

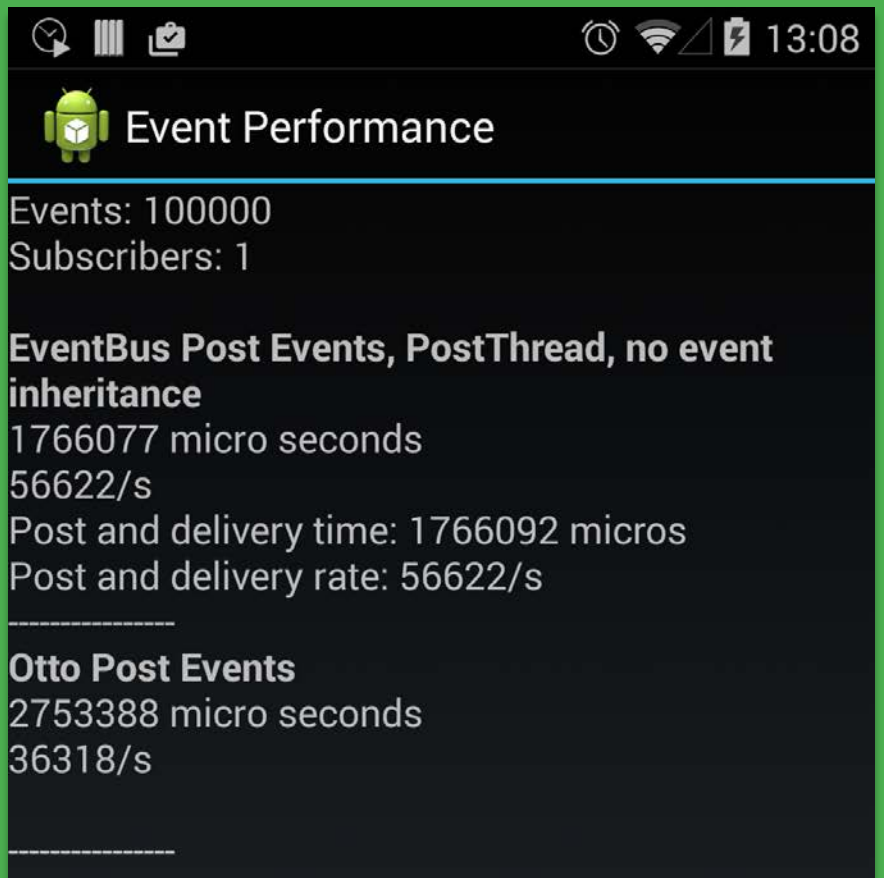
- First appearance in Android 4.4
  Dalvik still default, ART somewhat hidden
- Replaced Dalvik in Android 5.0
- Ahead of time compilation (AOT)
- Better Garbage Collection (GC)
- 64 bit support
- Better Profiling and Debugging
- Underdocumented

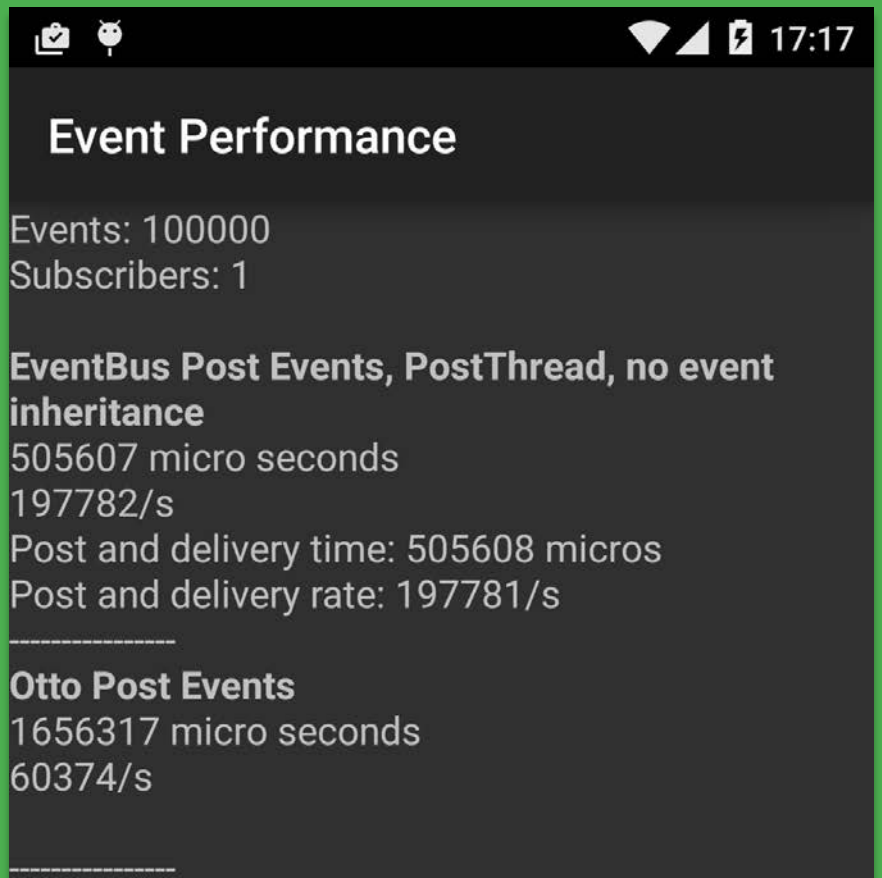
ART - Ahead of Time Compilation

- Compilation during installation
  Installation takes longer
  More storage required (DEX + Compiled)
- Better startup time
- No compilation lags during execution
- Compiled ART code is faster
  than compiled Dalvik code
- Better battery life, less memory consuption

ART - Android 4.4 vs. 5.0 Performance

- Reference: ~80,000 Events/s Dalvik 4.4





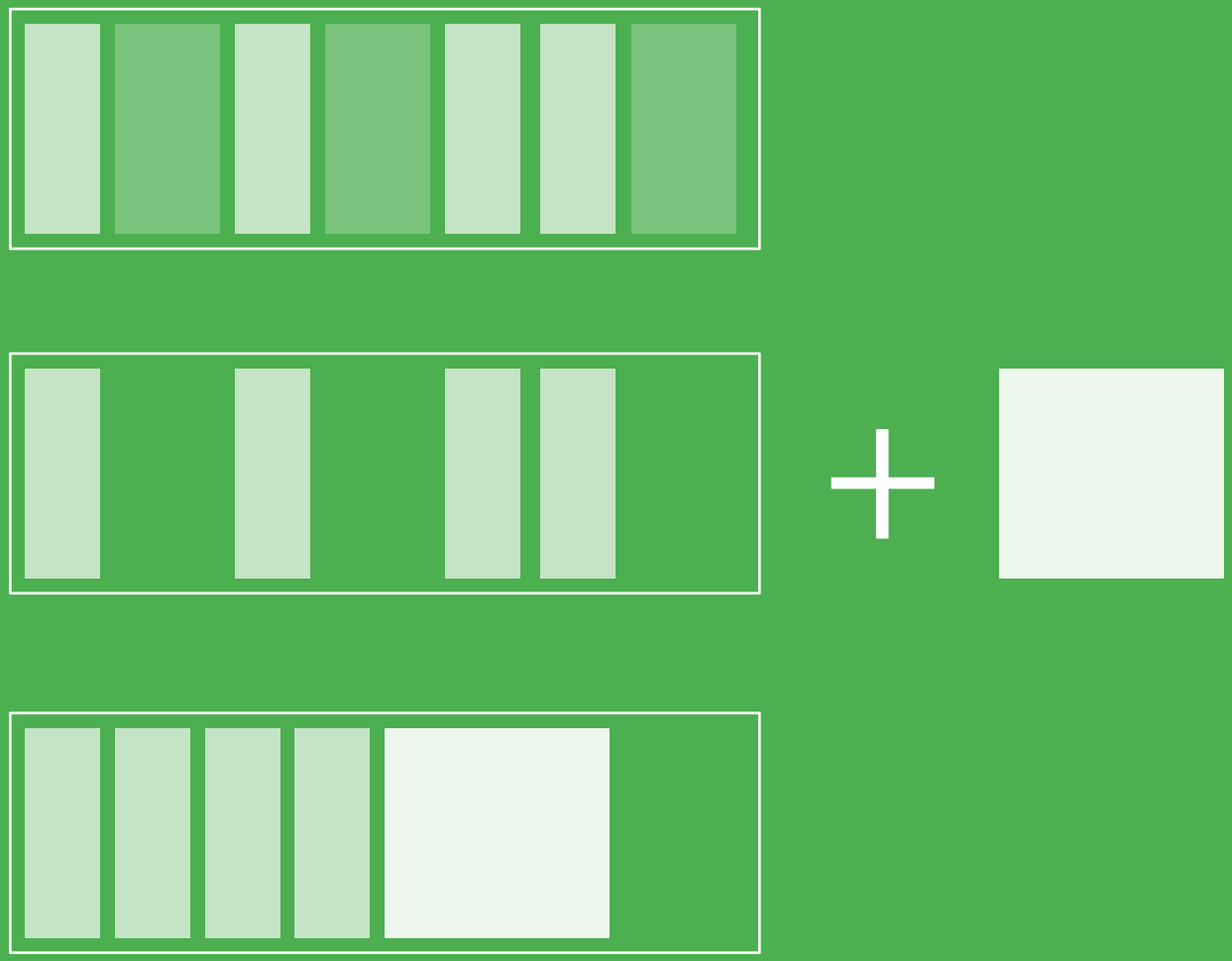
Is AOT always faster?

```
final boolean flag;

int calc(int[] bigData) {
     int answer = 42;
     for(int x: bigData) {
          answer += x;
          if(flag) {
               answer *= 23;
          }
     }
}
```

ART – Garbage Collection

- More Parallelism
  One GC cycle: 1 Pause instead of 2 (Dalvik)
- Shorter Pauses
  About half the delay of Dalvik
- Short-lived objects are collected faster
- Separate section for large Objects (Bitmaps)
- GC_FOR_ALLOC on Dalvik: ~60 ms
- Allocation: no more malloc/free

ART – Compacting Garbage Collection



Compacting GC - Consequences

- Avoiding OutOfMemoryError
- Objects may move in memory
- Check your JNI code!
- Compacting is expensive
- Might not run while the app is active

ARTsy Exceptions

- NullPointerException in line 123

```
user.login(ui.getUser().toString())
```

- ART gives extra info

```
java.lang.NullPointerException: Attempt to
invoke virtual method 'java.lang.String
java.lang.Object.toString()' on a null object
reference
```

- Native crashes show Java stack

Detect ART, GC for Allocation

- Developer could help Dalvik out

```
System.gc(); // in certain situtations only!
```

- ART handles those situations better
- Detect ART to optimize

```
System.getProperty("java.vm.version")
// "2.0.0" or above == ART
```

Interesting Bits & Pieces

Binding to a Service

- Context.bindService()
- Requires explicit intents
- Throws exception with implicit intents
- Verify your code and libs!
- Example: Old Google Analytics V2 Jar
  Crashes the app on Android 5.0

PDF Rendering

- Render bitmaps from PDFs
- File based (ParcelFileDescriptor)
- android.graphics.pdf.PdfRenderer
  Used to query page count and „open“ pages
- PdfRenderer.Page
  Get a page‘s dimensions
  Render page into a bitmap

WebView

- Chromium 37
- WebGL
- WebAudio
- Updateable from Google Play (!)
- Target SDK 21 has different defaults
  Blocks mixed content (HTTPS & HTTP)
  Blocks 3rd party cookies
- Permissions for camera, microphone, …

We could go on and on and on…

- Even more powerful Notifications
  Privacy setting for lockscreen
  Heads up notifications (floating)
- Camera2 API, deprecates Camera
  More control, burst mode, etc.
- Job scheduling to save battery
  Enqueue jobs and let the system decide when

Compatibility and Support Libraries

Support Android 5.0 optionally

- Set target level in Manifest
  `android:targetSdkVersion="21"`
- Check version in code
  `if (Build.VERSION.SDK_INT >= 21) {…}`
- Use version qualifiers for resource folders
  values-v21/

App Compat Library V21

- History: Started with ActionBar, etc.
- ToolBar
- Material Theme with customizable colors
- Tinting for some Views (Toolbar, Checkbox, …)
- Android 5.0 SearchView Widget

App Compat Library V21 - Integration

- For Android 2.1+ (API level 7)
- Depends on the v4 Support Library
  Fragments, etc.
- Gradle dependency

```
compile "com.android.support:appcompat-v7:21.0.+"
```

App Compat Library – Beware!

- Crashes on Android 4.2.2
  https://code.google.com/p/android/issues/detail?id=78377
  Not Google‘s fault, but our problem
  Workaround: repackage classes with ProGuard
- Some other minor bugs and glitches
- Watch for updates
- Test carefully

More Support Libraries related to Lollipop

- Palette
  Extract primary colors from Bitmap
  com.android.support:palette-v7:21.0.+

- Card Views
  Uses elevation on Android 5.0
  Shadow fallback for Pre-5.0
  com.android.support:cardview-v7:21.0.+

Android 5.0 is different.
Rethink your UI.

Thank You!

Copyright & License Terms

Extended CC Attribution