

# Informatikwerkstatt

## Android - Einführung

28.11.2018



Bildquelle: [https://developer.android.com/images/brand/Android\\_Robot\\_200.png?hl=de](https://developer.android.com/images/brand/Android_Robot_200.png?hl=de)

# TU Clausthal

## Literaturhinweis

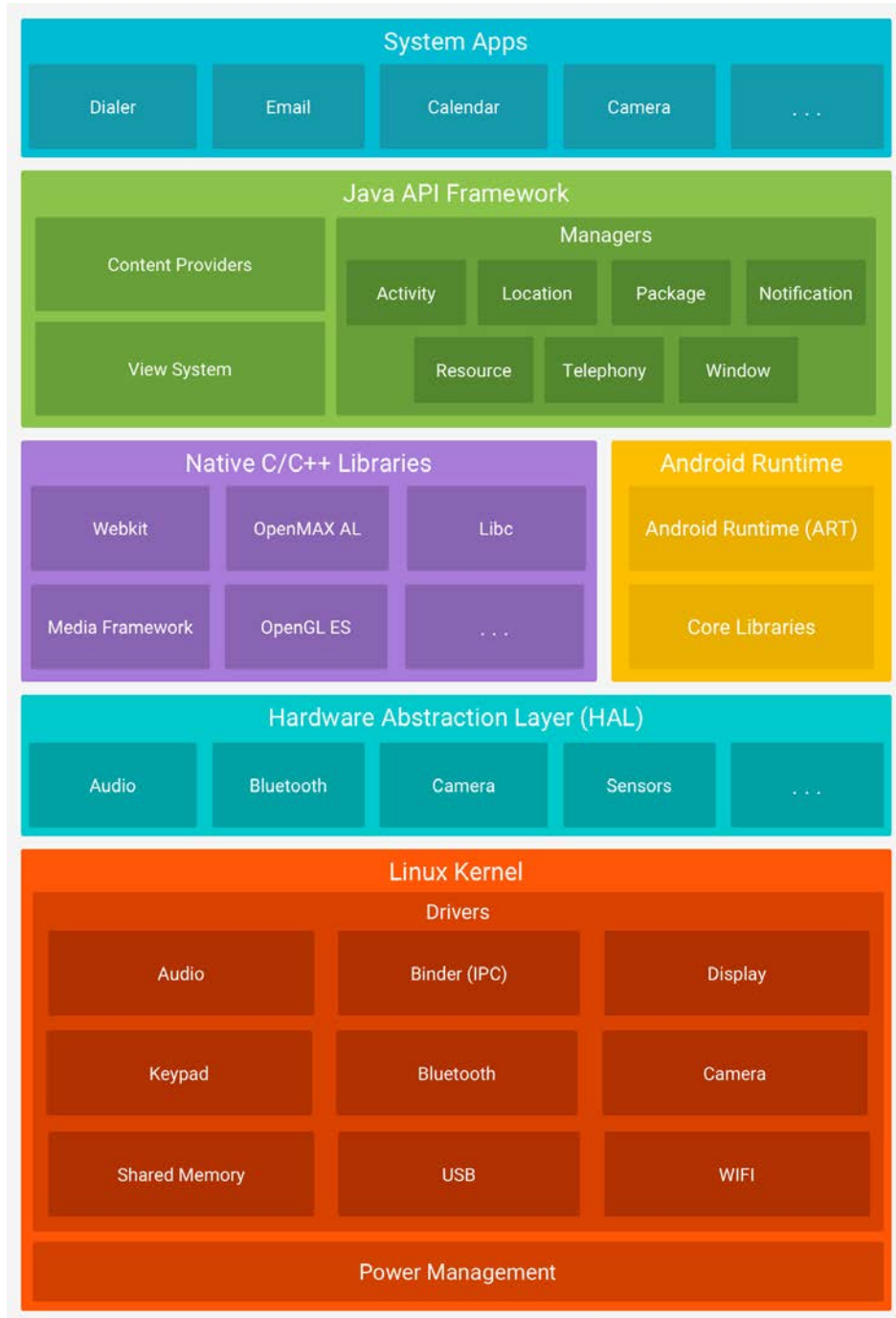
- Folien und Inhalte beruhen teilweise auf
  - <https://informatikwerkstatt.github.io/android-grundlagen/#/>
  - [Informatikwerkstatt — Mobile App Entwicklung mit Android](#)
  - [Philipp Kraus](#) — 23. Nov 2018 - 16:21
  - Folien von Dr.-Ing. A. Reinhardt, TU-Clausthal, 2016

## Android

- Veröffentlicht 2008 von Google
- Betriebssystem für viele Endgeräte
  - Smartphone u. Tablet
  - Smartwatches
  - Car Infotainment
  - HDMI Sticks

## Architektur

- **System und andere Apps**
  - Installierte Anwendungsprogramme
- **API Framework**
  - Programmierschnittstelle von Android
  - Abstrahieren darunterliegende komplexe Systeme
- **Software**
  - Bibliotheken für grundlegende Funktionen
  - Android Runtime zum Ausführen von Apps
- **Hardware Abstraction Layer**
  - Hardwareabstraktion für die Bibliotheken
  - Funktionen steuern konkrete Hardware an
- **Linux Kernel**
  - Kern des Betriebssystems
  - Steuert Hardware Zugriffe
  - Stark durch Google Angepasst
  - Basierend auf einem Kernel der Version 3 oder höher



## Schnittstellen – API

- API = „application programming interface“
- Jede Schicht stellt Schnittstellen als API bereit
- @Profis: Übersicht der APIs
  - <https://developer.android.com/reference/packages>

# TU Clausthal

## Android Studio

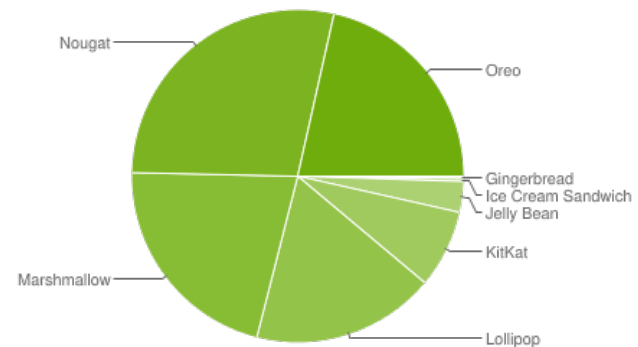
- Entwicklungsumgebung für Apps
- Basis von [IntelliJ](#) der Firma [JetBrains](#)
- [IntelliJ](#) ist eine Alternative zu Eclipse

Kommando	Shortcut
Klasse finden	Control + N
Suchen	Control + F
Ersetzen	Control + R
Optimiere Imports	Control + Alt + O
Reformat code	Control + Alt + L
Quick-Fix	Alt + Enter
Basis Code-Ergänzung	Control + Space
Smarte Code-Ergänzung	Control + Shift + Space
Erzeugen & Ausführen	Shift + F10

## SDK – „Software development kit“

- Sammlung von Programmierwerkzeugen und Programmbibliotheken
- Hilft Softwareentwicklern als Schnittstelle
- JDK -> SDK

## Verteilung - 2018



Version	Codename	API	Distribution
2.3.3 – 2.3.7	Gingerbread	10	0.2%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x – 4.2.x – 4.3	Jelly Bean	16 – 17 – 18	3%
4.4	KitKat	19	7.6%
5.0 – 5.1	Lollipop	21 – 22	17,9%
<b><u>6.0</u></b>	<b><u>Marshmallow</u></b>	<b><u>23</u></b>	<b><u>21,3%</u></b>
7.0 - 7.1	Nougat	24 – 25	28,2%
8.0 - 8.1	Oreo	26 – 27	21,5%

Stand: 26. Oktober 2018

Bildquelle:

<https://chart.googleapis.com/chart?chf=bg%2Cs%2C00000000&chd=t%3A0.2%2C0.3%2C3.0%2C7.6%2C17.9%2C21.4%2C28.2%2C21.5&chco=c4df9b%2C6fad0c&chl=Gingerbread%7CIce%20Cream%20Sandwich%7CJelly%20Bean%7CKitKat%7CLollipop%7CMarshmallow%7CNougat%7COreo&chs=500x250&cht=p>





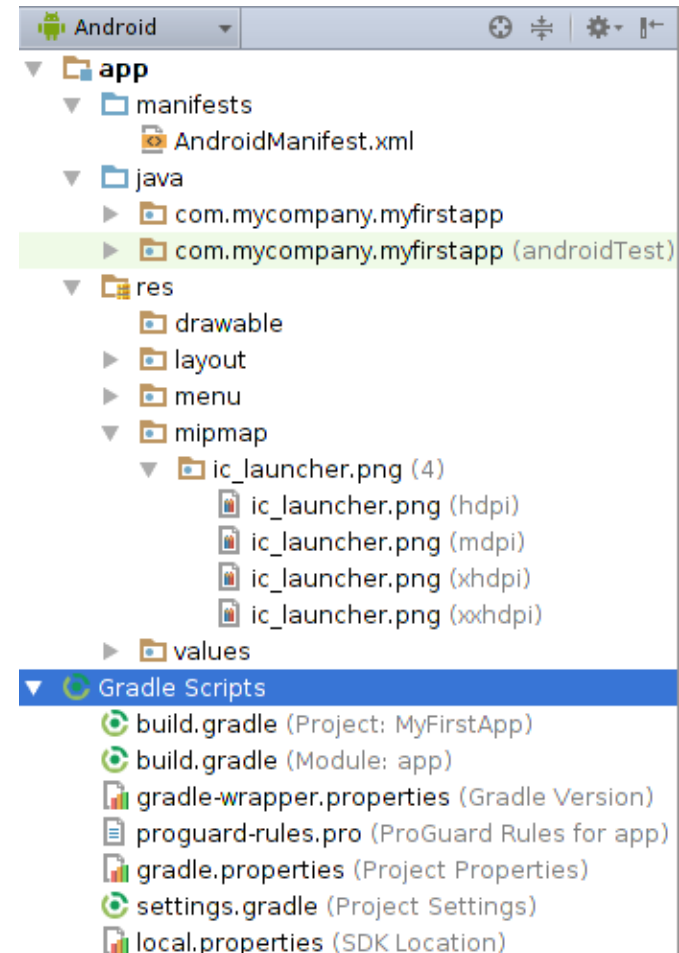
# TU Clausthal

## @LET'S TRY

- Empty Activity
  - <https://vimeo.com/287431166>

## Projektstruktur

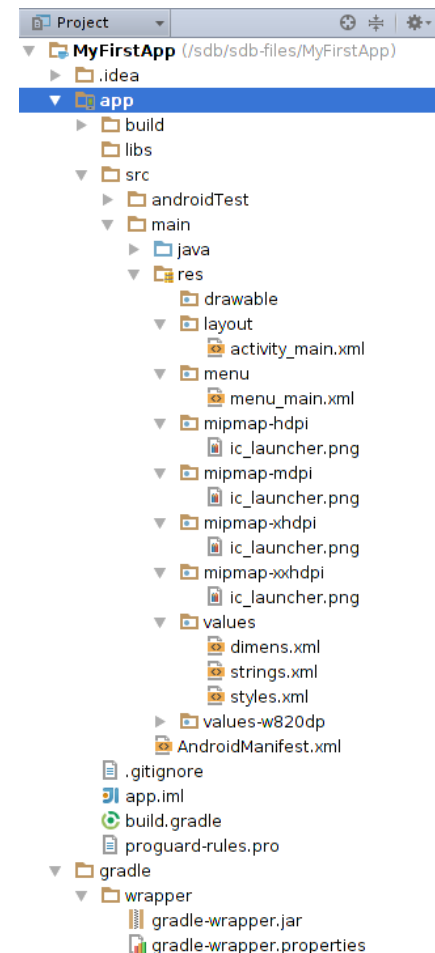
- *app*
  - Hauptverzeichnis
- *manifests*
  - Konfiguration des Projekts
- *java*
  - Quellcodedateien
    - Quellcode der App
    - Quellcode für Tests
- *res*
  - Bilder, Icons
  - Layouts
    - Grafische Darstellungen
  - Values
    - Colors, strings, styles - Ressourcen
- *Gradle Scripts*
  - Wird z.B dafür verwendet die App zu compilieren



Bildquelle: <https://developer.android.com/images/tools/projectview-p1.png>

## Empty Activity

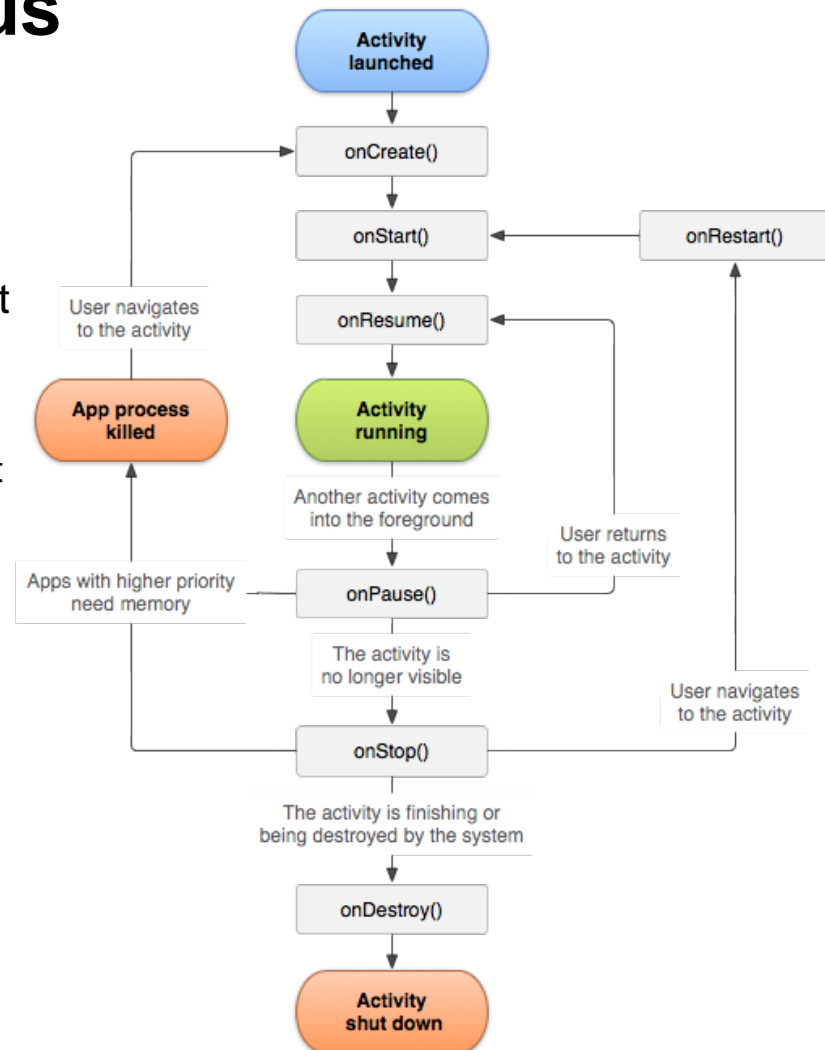
- `app/src/main/res/layout/`
  - `activity_main.xml` enthält unsere erste Activity
  - Jede weitere Activity wird im selben Ordner hinterlegt
- [app/src/main/AndroidManifest.xml](#)
  - Beinhaltet Konfigurationen der App (Label, Icon, MainActivity)
  - Berechtigungen können erteilt werden (Sensoren, Kamera)



Bildquelle: <https://developer.android.com/images/tools/projectview-p2.png>

## Activity - Lebenszyklus

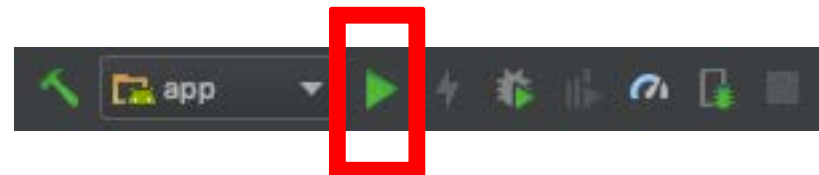
- Wichtig für unser Vorhaben:
  - onCreate()
    - Aufruf, bevor Activity sichtbar ist
  - onStart()
    - Activity ist bereits sichtbar, Änderungen geschehen zur Laufzeit



Bildquelle: [https://developer.android.com/guide/components/images/activity\\_lifecycle.png](https://developer.android.com/guide/components/images/activity_lifecycle.png)

## Build-Prozess & APK (Android Package)

- Der *Build-Prozess* erstellt aus allen Elementen (Quellcode, Ressourcen) ein fertiges Paket
- Das Paket wird in einer APK-Datei zusammengefasst
- Die Datei wird auf das Tablet übertragen und installiert
- Mit einer APK Datei ist es möglich, Apps unabhängig vom Google Playstore zu installieren
- Der Build-Prozess wird mit dem *grünen „Play“-Button* in der rechten oberen Fensterecke gestartet




## Logger

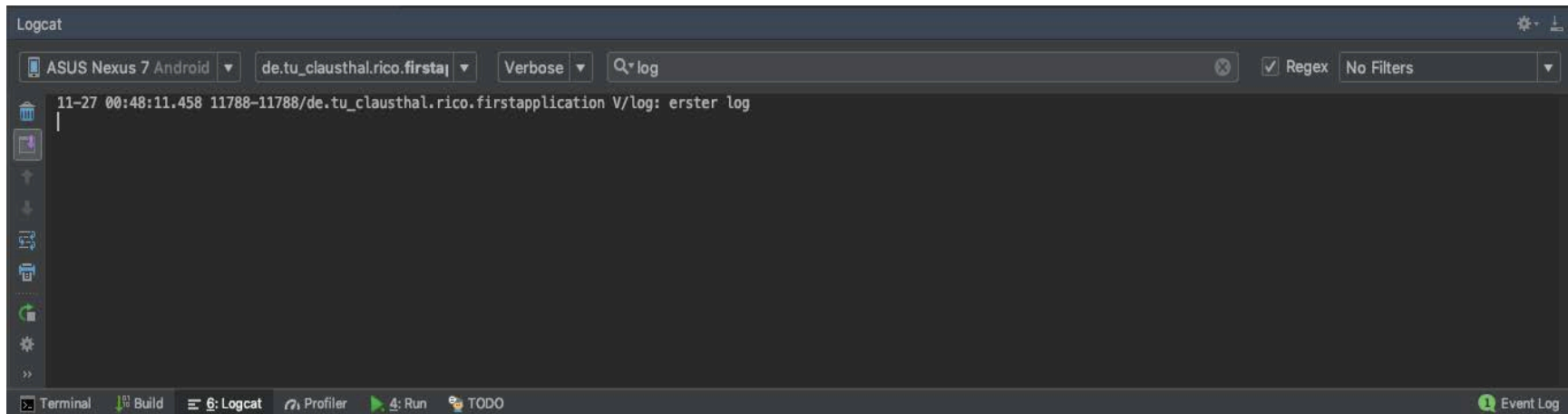
- *System.out.println* mit erweiterter Funktionalität
- Ausgaben können besser strukturiert werden
- Es können mehrere Loglevels ausgewählt werden

Log Level	Source Code
Verbose	<code>Log.v( "tag", "text" )</code>
Debug	<code>Log.d( "tag", "text" )</code>
Info	<code>Log.i( "tag", "text" )</code>
Warn	<code>Log.w( "tag", "text" )</code>
Error	<code>Log.e( "tag", "text" )</code>

Beispiel: **Log.e**( “rechenoperation”, “Division durch null” );

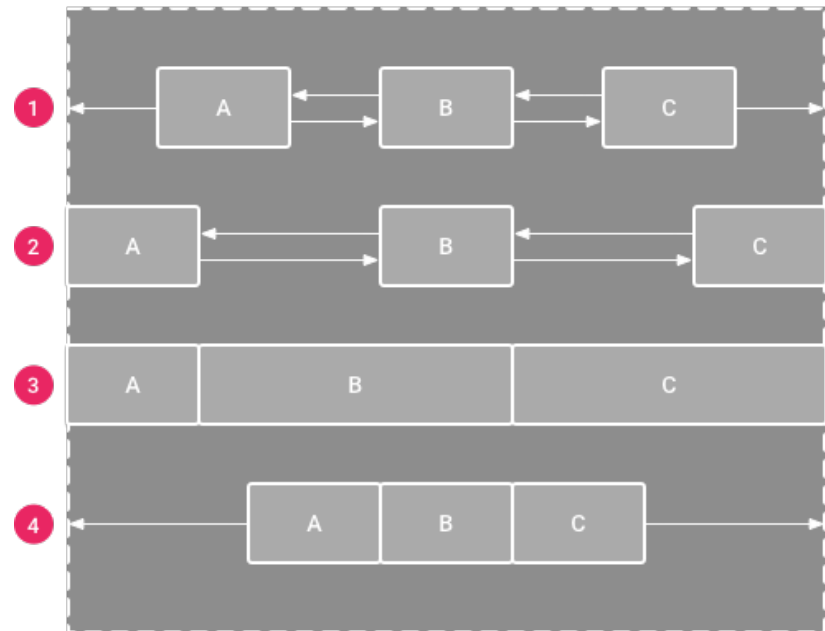
## Log anzeigen

- Zuerst muss die App gestartet werden
- Im unteren Teil des Fensters wählen Sie  Logcat
- Im Suchfeld können Sie den Tag eingeben



## Layout - ConstraintLayout

- Aktuelle und gängige Wahl für ein Layout
- Vorteil:
  - Flexibel bei verschiedenen Displaygrößen
  - Bietet eine übersichtliche horizontale und vertikale Darstellung
- Gedanklich: in eine große Box werden viele kleinere Boxen verbaut
  - Es ergibt sich die Layout-Hierarchie des Component Tree



Bilderquelle: [https://developer.android.com/training/constraint-layout/images/constraint-chain-styles\\_2x.png](https://developer.android.com/training/constraint-layout/images/constraint-chain-styles_2x.png)



## Widgets

- Jedes GUI-Element „UI Widget“ besitzt eine eindeutige ID und kann damit gefunden werden
- ID's werden unter **R.id.ID** gefunden
- Zuweisung einer TextView Variable:

```
TextView textview = findViewById(R.id.ID);
```

- In älteren SDK ist es nötig zu casten!

```
TextView textview =  
(TextView)findViewById(R.id.ID);
```

- Zuweisung einer Button Variable:

```
Button button = findViewById(R.id.button);
```

- Jetzt ist es möglich, z.B den Text zu ändern

```
textview.setText("Hallo Welt!");
```

## Listener

- Viele Datenquellen sind ereignisgesteuert
- Ein *Listener* bietet die Möglichkeit, Anwendungen über Ereignisse zu informieren
- Wenn ein Listener benötigt wird, muss erst ein passendes Listener-Interface implementiert werden
- Als Alternative dazu ist es möglich, „kleine“ Listener hinzuzufügen



- Beispiel: OnClickListener:

```
button.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
    }  
});
```

## @LET'S TRY: PFLICHTABGABE ZUM 4.12.18

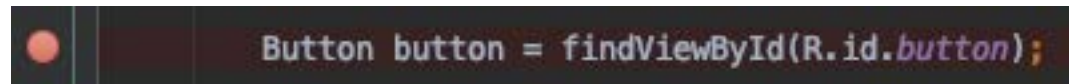
- Füge mindestens einen Button dem [ConstraintLayout](#) hinzu und erstelle einen passenden `OnClickListener` mit einer Log Ausgabe
- BONUSAUFGABE
  - Sei kreativ, überlege dir eine „Hello World“ Anwendung





## Debugger

- Der häufigste Fehler ist der Syntaxfehler, der bereits vom Compiler erkannt wird
- Bei weiteren Fehlern, die zur Laufzeit auftreten, hilft der Debugger bei der aktiven Fehlersuche
-  Start des Debuggers mit Installation der APK
-  Hängt den Debugger den Android Prozessen an
- Breakpoints unterbrechen die Ausführung, damit der aktuelle Zustand beobachtet werden kann
- Zum aktuellen Zustand gehört der Zustand des Speichers sowie der Variablenbelegung

## Breakpoints

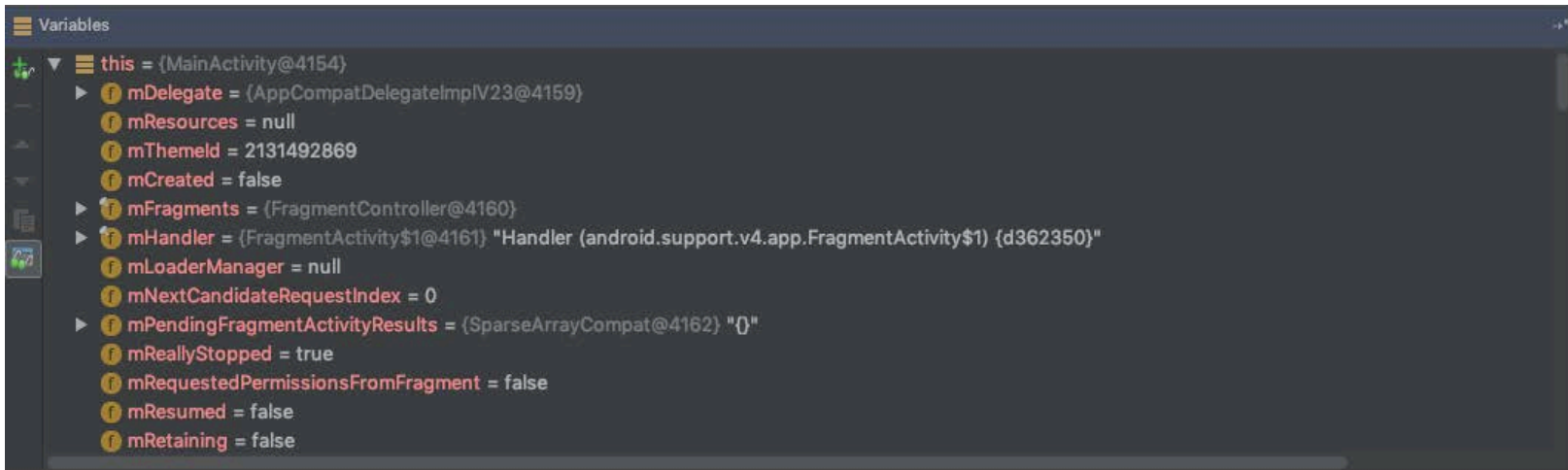
- Können hinzugefügt werden, indem auf die Zeile in dem grauen Bereich geklickt wird.



-  Step Over: Sprung in die nächste Zeile
-  Step Into: Sprung zur ersten Zeile der Methode am Breakpoint
-  Evaluate Expression: Wertet Ausdruck aus
-  Resume Program: Programmausführung fortsetzen bis zum nächsten Breakpoint oder Ende

## Debugger - Variablen

- Im Folgenden werden alle aktuellen Werte zur Laufzeit sichtbar
- Werteveränderungen sind nachvollziehbar



## Alternative zu Java

- „[React](#) ist 2013 erschienen (2015 publiziert von Facebook unter OpenSource Lizenz)“
- „[JavaScript](#) ist die verwendete Programmiersprache“
- „[Instagram](#), [Facebook](#) und [Pinterest](#) nutzen das Framework“
- „die Code-Basis ist durch das Framework für die Webseite (inkl. mobile Version) und Mobile-App identisch“
- „somit ist die Wiederverwertbarkeit mit kürzerer Entwicklungszeit möglich“

## ÜBUNG/@HOME: PFLICHTABGABE 6 ZUM 04.12.2018

1. Ergänzt die Hello-World App um den Logger
2. Probiert die verschiedenen *Log Level* aus
3. Überprüft die Log Nachrichten auf dem PC und dem Tablet

**Wozu könnten die verschiedenen Log Levels sinnvoll sein?**

### 4. BONUSAUFGABE

- Implementiere weitere Variablen-Zuweisungen und kleine Berechnungen
- Setze Breakpoints und überprüfe mit Hilfe des Debuggers die Belegung der Variablen
- Implementiere einen Fehler, so dass eine [Exception](#) geworfen wird und probiere aus, wie man diesen Fehler mittels Breakpoints finden kann

**Was ist ein sinnvolles Vorgehen, um mit  
Breakpoints zu arbeiten?**



## @HOME

- Sinnvoll ist es, sich [Android Studio](#) zu installieren
- Arbeiten ohne Tablet ist möglich mit Hilfe eines Virtual Device
- Je nach Gerät muss eine andere SDK installiert werden

## Quellen

- <https://informatikwerkstatt.github.io/android-grundlagen/#/>
- <https://developer.android.com/about/dashboards/>
- <https://developer.android.com/guide/components/activities/activity-lifecycle>