Beispiele Android

API Check

```
int device = android.os.Build.VERSION.SDK_INT; int required = Build.VERSION_CODES.Q;
if (device >= required) {
    // nur auf Geräten mit API Level >= 29 (Q)
}
```

Activity

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:orientation="vertical"
         android:gravity="center">
        <TextView
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello Android!"
              android:textSize="20sp"/>
</LinearLayout>
```

Constraint Layout

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center">
    <TextView
        android:id="@+id/txtAppTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginVertical="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/txtAppVersion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginVertical="16dp"
        android:text="@string/txtAppVersion"
        android:textColor="?android:textColorSecondary"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtAppTitle" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

TextView

```
<TextView
          android:allowUndo
          android:autoLink
          android:breakStrategy
          android:drawable[Bottom|End|Left|Right|Start|Top]
          android:editable
          android:lineBreakStyle
          android:inputType
          android:textColor
          android:textSize />
```

Menus

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_overflow_holo_dark"
    android:contentDescription="@string/descr_overflow_button"
    android:onClick="showPopup" />
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/menu_1"
            android:title="Menu 1"
            android:icon="@drawable/ic_bulb"
            app:showAsAction="always"/>
        <item android:id="@+id/menu_2"
            android:title="Menu 2"
            android:icon="@drawable/ic_star"
            app:showAsAction="always|withText"/>
        <item android:id="@+id/menu_3"
            android:title="Menu 3"
            app:showAsAction="never"/>
</menu>

fun showPopup(v: View) {
    val popup = PopupMenu(this, v)
    val inflater: MenuInflater = popup.menuInflater
    inflater.inflate(R.menu.actions, popup.menu)
    popup.show()
}
fun onOptionsItemSelected(item: MenuItem) : boolean {
    switch (item.getItemId()) {
        case R.id.menu_1:
        …
        break;
        case R.id.menu_2:
        …
        break;
        case R.id.menu_3:
        …
        break;
    }
    return true;
}
```

ListView mit (eigenem) ArrayAdapter

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="…"
    android:id="@+id/list_example"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ListView>

setContentView(R.layout.activity_main);
String[] data = new String[] { … };
ArrayAdapter<String> adapter = new ArrayAdapter<>(
    this,
    android.R.layout.simple_list_item_1, // Standard-Layout
    android.R.id.text1,                 // Standard-Layout
    data);
ListView listView = findViewById(R.id.list_example);
listView.setAdapter(adapter);

setContentView(R.layout.activity_main);
ArrayList<User> data = UserManager.getUsers();
UsersAdapter adapter = new UsersAdapter(this, data);
ListView listView = findViewById(R.id.list_example);
listView.setAdapter(adapter);

public class User {
    public String name; public int age;
    public User(String name, int age) {
        this.name = name; this.age = age;
    }
}

public class UsersAdapter extends ArrayAdapter<User> {
    public UsersAdapter(ctx c, ArrayList<User> users) {
        super(ctx, 0, users);
    }

@Override
public View getView(int pos, View view, ViewGroup parent) {
    if (view == null) {
        Context context = getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        view = inflater.inflate(
            android.R.layout.simple_list_item_2,
            parent,
            false);
    }
    TextView text1 = view.findViewById(android.R.id.text1);
    TextView text2 = view.findViewById(android.R.id.text2);
    User user = getItem(pos);
    text1.setText(user.name);
    text2.setText(user.age + " Jahre");
    return view;
    }
}
```

ListenAdapter mit View Holder

```
private class ViewHolder { TextView text1; TextView text2; }
@Override
public View getView(int pos, View view, ViewGroup parent) {
    ViewHolder viewHolder;
    if (view == null) {
        Context context = getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        view = inflater.inflate(android.R.layout.simple_list_item_2,
            parent, false);  // bis hierher bekannt

        viewHolder = new ViewHolder();
        viewHolder.text1 = view.findViewById(android.R.id.text1);
        viewHolder.text2 = view.findViewById(android.R.id.text2);
        view.setTag(viewHolder);
    } else { viewHolder = (ViewHolder)view.getTag(); }
    User user = getItem(pos);
    viewHolder.text1.setText(user.name); viewHolder.text2.setText(user.age + " Jahre");
    return view;
}
```

RecyclerView

```
<?xml version="1.0" encoding="utf-8"?> // [ Main Activity ]
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</androidx.recyclerview.widget.RecyclerView>

// MainActivity.java
setContentView(R.layout.activity_main);
RecyclerView recyclerView = findViewById(R.id.recycler_view);

RecyclerView.LayoutManager layoutManager;
layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
ArrayList<User> data = UserManager.getUsers();
UsersAdapter adapter = new UsersAdapter(data);
recyclerView.setAdapter(adapter);

// UsersAdapter.Java
public class UsersAdapter extends RecyclerView.Adapter<ViewHolder>
{
    private ArrayList<User> users;
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int vt) {
        Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        View view = inflater.inflate(android.R.layout.simple_list_item_2,
                              parent, false);
        return new ViewHolder(view, view.findViewById(android.R.id.text1),
                              view.findViewById(android.R.id.text2));
    }
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        User user = this.users.get(position);
        holder.text1.setText(user.name);
        holder.text2.setText(user.age + " Jahre");
    }
    @Override
    public int getItemCount() {
        return this.users.size();
    }
}
```

Intents

```
// Expliziter Intent
Intent secondActivityIntent = new Intent(
    this,
    SecondActivity.class);
startActivity(secondActivityIntent);
// Impliziter Intent
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.setType("text/plain");
sendIntent.putExtra(Intent.EXTRA_TEXT, "Hey!");
startActivity(sendIntent);
```

GUI-Aktualisierung

```
final Runnable updateUi = new Runnable() {
    @override public void run() {
        textOutput.setText("Updated!");
    }
};
Runnable background = new Runnable() {
    @Override public void run() {
        Thread.sleep(3000);
        activity.runOnUiThread(updateUi); // v1
        textOutput.post(updateUi);          // v2

        Looper looper = Looper.getMainLooper(); // v3
        Handler handler = new Handler(looper);
        handler.post(updateUi);
    }
}
Thread thread = new Thread(background); thread.start();
```

Fragments statisch

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); // Referenz auf Activity XML
    }
}

<LinearLayout xmlns:android="(…)"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Fragment android:name="(…).OutputFragment" // Referenz auf OutputFragment.java
        android:id="@+id/main_fragment_output"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>

public class OutputFragment extends Fragment {
    public OutputFragment {
        super(R.layout.fragment_output); // Referenz auf Fragment XML
    }
}

<LinearLayout xmlns:android="(...)"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
         android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello Fragment" />
</LinearLayout>
```

Fragments dynamisch

```
<!-- Platzhalter im XML -->
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/main_fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout="@layouts/fragment_infos" /> <!-- zur Hilfe im XML Designer -->
```

```java
// Activity verwendet FragmentManager
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
        FragmentManager mgr = getSupportFragmentManager();
        FragmentTransaction trans = mgr.beginTransaction();
        OutputFragment fragment = new OutputFragment();
        trans.add(R.id.main_fragment_container, fragment);
        trans.commit();
    }
}
```

Animation

```java
fragmentManager.beginTransaction()
    .setCustomAnimations(
        R.anim.slide_in, // Einblendung neues Fragment
        R.anim.fade_out, // Ausblendung altes Fragment
        R.anim.fade_in, // Einblendung altes Fragment (Pop)
        R.anim.slide_out) // Ausblendung neues Fragment (Pop)
    .replace(R.id.main_fragment_container, newFragment)
    .addToBackStack(null)
    .commit();
});
```

Permissions

```java
// Permission Check
int status = ContextCompat.checkSelfPermission(this, permission);
if (status != PackageManager.PERMISSION_GRANTED) {
    if (shouldShowRequestPermissionRationale(permission)) {
        // erste erklärung liefern
    }
    requestPermissions(new String[] { permission }, CALLBACK_CODE);
}
// Resultat in Lifecycle Methode
@Override public void onRequestPermissionResult(
    int requestCode, String[] permissions, int[] results) {
    if (requestCode != CALLBACK_CODE) return; // ID der expliziten Permissions-Abfrage.
    if (results.length == 0) return; // Anfrage abgebrochen
    if (results[0] == PackageManager.PERMISSION_GRANTED) {
        // Berechtigung erteilt
    } else { /* Berechtigung verweigert */ }
}
```

Filezugriff

```java
// Schreiben
File folder = getFilesDir();
File file = new File(folder, "my_file.txt");
String input = "MGE Beispiel";
FileOutputStream outputStream = new FileOutputStream(file);
outputStream.write(input.getBytes());
outputStream.close();
// Dateien anzeigen
for(File fileInFolder : folder.listFiles()) {
    Log.d("MGE.V05", "File: " + fileInFolder.getName());
}
// Lesen
int length = (int) file.length();
byte[] bytes = new byte[length];
FileInputStream inputStream = new FileInputStream(file);
inputStream.read(bytes);
inputStream.close();
String output = new String(bytes);
```

Preferences

```java
String file = "ch.ost.rj.mge.v05.myapplication.preferences";
String key1 = "my.key.1";
String key2 = "my.key.2";
String key3 = "my.key.3";
int mode = Context.MODE_PRIVATE;
// Objekt abholen
SharedPreferences preferences;
preferences = getSharedPreferences(file, mode);
// Schreiben
SharedPreferences.Editor editor = preferences.edit();
editor.putString(key1, "MGE Beispiel");
editor.putBoolean(key2, true);
editor.putInt(key3, 42);
editor.commit();
// Lesen
String value1 = preferences.getString(key1, "default");
boolean value2 = preferences.getBoolean(key2, false);
int value3 = preferences.getInt(key3, 0);
```

Datenbanken

Medien

```java
// Auslesen von Bildern sortiert nach Einfügedatum
String[] projection = new String[] {
    MediaStore.Images.Media.TITLE,
    MediaStore.Images.Media.DATE_ADDED
};
String order = MediaStore.Images.Media.DATE_ADDED + " DESC";
Cursor cursor = getContentResolver().query(
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
    projection, // projection
    null, // selection
    null, // selectionArgs
    order // sortOrder
    );
int ct = cursor.getColumnIndex(MediaStore.Images.Media.TITLE);
int cd = cursor.getColumnIndex(MediaStore.Images.Media.DATE_ADDED);
while (cursor.moveToNext()) {
    String title = cursor.getString(ct);
    long added = cursor.getLong(cd);
    // … hier die Werte verwenden …
}
cursor.close();
```

Dokumente

```java
private static final int CREATE_DOCUMENT_CODE = 1;
private static final int OPEN_DOCUMENT_CODE = 2;
private static final String FILE_NAME = "my_file.txt";
private static final String FILE_TYPE = "text/plain";
// Intent zum Schreiben eines Dokuments
Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);
intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.setType(FILE_TYPE);
intent.putExtra(Intent.EXTRA_TITLE, FILE_NAME);
startActivityForResult(intent, CREATE_DOCUMENT_CODE);
// Intent zum Öffnen eines Dokuments
Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.setType(FILE_TYPE);
startActivityForResult(intent, OPEN_DOCUMENT_CODE);

@Override
public void onActivityResult(int req, int res, Intent data) {
    super.onActivityResult(req, res, data);
    switch(req) {
        case CREATE_DOCUMENT_CODE:
            if (res == Activity.RESULT_OK) {
            Uri uri = data.getData();
            // Mit Content Resolver Uri verarbeiten
            }
            break;
        case OPEN_DOCUMENT_CODE:
            if (res == Activity.RESULT_OK) {
                Uri uri = data.getData();
                // Mit Content Resolver Uri verarbeiten
            }
    }
}
```

Netzwerkverbindung

```java
String service = Context.CONNECTIVITY_SERVICE;
ConnectivityManager manager;
manager = (ConnectivityManager) getSystemService(service);
// Aktive Verbindung prüfen
NetworkInfo activeNetwork = manager.getActiveNetworkInfo();
if (activeNetwork != null) {
    int type = activeNetwork.getType();
    Log.d(null, "Active connection: " + type);
}
// Verbindungen prüfen
for (Network network : manager.getAllNetworks()) {
    NetworkInfo info = manager.getNetworkInfo(network);
```

```java
        boolean state = info.isConnected();
        if (info.getType() == ConnectivityManager.TYPE_WIFI) {
            Log.d(null, "WiFi is connected: " + state);
        }
        if (info.getType() == ConnectivityManager.TYPE_MOBILE) {
            Log.d(null, "Mobile is connected: " + state);
        }
    }
```

Positionsbestimmung

```java
// Kriterien für Provider definieren
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setSpeedRequired(false);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(false);
criteria.setPowerRequirement(Criteria.POWER_MEDIUM);
// Provider abholen
String service = Context.LOCATION_SERVICE;
LocationManager manager;
manager = (LocationManager) getSystemService(service);
String provider = manager.getBestProvider(criteria, true);
// Updates abonnieren
manager.requestLocationUpdates(provider, 5000L, 0, this);
@Override
public void onLocationChanged(Location loc) {
    Log.d(null, loc.getLatitude() + "|" + loc.getLongitude());
}
```

Broadcast

```java
public class MyBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(LOG_TAG, "Broadcast | Action: " + intent.getAction());

        if (intent.getExtras() != null
                && intent.getExtras().containsKey(MyStartedService.SERVICE_RESULT_KEY)) {
            Log.d(LOG_TAG, "Broadcast | Service Result: " + intent.getIntExtra(
                MyStartedService.SERVICE_RESULT_KEY, 0));
        }
    }
}
public class MainActivity extends AppCompatActivity {
        private void registerBroadcastReceiver() {
        receiver = new MyBroadcastReceiver();

        IntentFilter filter = new IntentFilter();
        filter.addAction(ConnectivityManager.CONNECTIVITY_ACTION);
        filter.addAction(BROADCAST_ACTION);

        registerReceiver(receiver, filter); // Dynamische Registrierung, vs. statisch im Manifest
    }
    private void unregisterBroadcastReceiver() {
        unregisterReceiver(receiver);
        receiver = null;
    }
    private void sendImplicitBroadcast() {
        // Can be simulated using:
        // adb shell am broadcast -a ch.ost.rj.mge.v06.myapplication.MY_INTENT
        Intent intent = new Intent();
        intent.setAction(BROADCAST_ACTION);
        intent.putExtra("data","example");
        sendBroadcast(intent);
    }
    private void sendExplicitBroadcast() {
        // Can be simulated using:
        // adb shell am broadcast -a ch.ost.rj.mge.v06.myapplication.MY_INTENT -n
            // ch.ost.rj.mge.v06.myapplication/.MyBroadcastReceiver
        Intent intent = new Intent(this, MyBroadcastReceiver.class);
        intent.setAction(BROADCAST_ACTION);
        intent.putExtra("data","example");
        sendBroadcast(intent);
    }
}
```

Started Service

```java
public class MainActivity extends AppCompatActivity {
    private void runStartedService() {
        Intent broadcastIntent = new Intent(this, MyBroadcastReceiver.class);
        broadcastIntent.setAction(BROADCAST_ACTION);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(
            this, 0, broadcastIntent, PendingIntent.FLAG_MUTABLE);

        Intent intent = new Intent(this, MyStartedService.class);
        intent.putExtra(MyStartedService.SERVICE_PI_KEY, pendingIntent);
        startService(intent);
    }
    private void stopStartedService() {
        Intent intent = new Intent(this, MyStartedService.class);
        stopService(intent);
    }

}
public class MyStartedService extends Service {
    private boolean stopRequested;
    @Override
    public void onCreate() { /* ... */ }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        new Thread(() => {
            /* do some things */
            if (stopRequested) { stopSelf(startId); }
        }).start();
    }
    @Override
    public void onDestroy() {
        logMessage("onDestroy");
        stopRequested = true;
        super.onDestroy();
    }
}
```

Bound Service

```java
public class MainActivity extends AppCompatActivity implements ServiceConnection {
    private void connectToBoundService() {
        Intent intent = new Intent(this, MyBoundService.class);
        bindService(intent, this, Context.BIND_AUTO_CREATE);
    }
    private void interactWithBoundService() {
        boundService.showToast("Hallo aus der MainActivity");
    }
    private void disconnectFromBoundService() {
        unbindService(this);
        onServiceDisconnected(null);
    }
}
@Override
public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
    // Call UI Related Logic
    MyBoundService.MyBinder binder = (MyBoundService.MyBinder) iBinder;
    boundService = binder.getService();
}

@Override
public void onServiceDisconnected(ComponentName componentName) {
    // only called on problems, not normal unbind!
    // Call UI Related Logic
    boundService = null;
}

public class MyBoundService extends Service {
    public class MyBinder extends Binder {
        public MyBoundService getService() {
            return MyBoundService.this;
        }
    }
    private final IBinder binder = new MyBinder();

    @Override public IBinder onBind(Intent intent) {
        logMessage("onBind"); return binder;
    }
    @Override public boolean onUnbind(Intent intent) {
        logMessage("onUnbind"); return super.onUnbind(intent);
    }
    @Override public void onDestroy() {
        logMessage("onDestroy"); super.onDestroy();
    }
}
```

Data Binding: Observables

Observable Fields

```
public class User {
    public final ObservableField<String> name = new ObservableField<>();
    public final ObservableInt age = new ObservableInt();

    public User(String name, int age) {
        this.firstName.set(name);
        this.age.set(age);
    }
}
```

Observable Classes

```
public class User extends BaseObservable {
    // Fields and Constructor...
    @Bindable   // This Annotation should be applied to any getter of Observable Fields.
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
        notifyPropertyChanged(BR.name); // BR ist eine automatisch generierte Klasse mit
    }                        // Properties, die fürs Binding verwendet werden.
}
```

Lifecycle Aware Components

```
public class MyActivity {
    private MyLocationListener myLocationListener;
    @Override protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUserBinding.inflate(...);
        binding.setVm(viewModel);
        binding.setLifecycleOwner(this);         // Lifecycle Owner für Data Bindings im XML
        myLocationListener = new LocationListener(
            getLifecycle(),     // eigener Lifecycle übergeben
            (location) -> { /* UI Update bei Positionsänderung */ });
        Util.check(result -> {
            if (result) { myLocationListener.enable(); }
        });
    }
}
// Listener Implementation
public class MyLocationListener implements LifecycleObserver {
    private final Lifecycle lifecycle;
    private boolean enabled = false;

    public MyLocationListener(Lifecycle lifecycle, Consumer<Location> callback) {
        this.lifecycle = lifecycle;
        this.lifecycle.addObserver(this);
    }
    @OnLifeCycleEvent(ON_START)
    void start() { /* ... */ }
    @OnLifeCycleEvent(ON_STOP)
    void stop() { /* ... */ }
    public void enable() {
        enabled = true;
        if (lifecycle.getCurrentState().isAtLeast(STARTED)) {
            start();
        }
    }
}
```

ViewModel

```
public class UserActivity extends AppCompatActivity {
    private ActivityUserBinding binding;
    @Override protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        User user = new User("Thomas", "Kälin", 36);
        UserViewModelFactory factory = new UserViewModelFactory(user);
        UserViewModel viewModel = new ViewModelProvider(this, factory) // Factory nur
            .get(UserViewModel.class);                      // benötigt wegen user-Parameter
    }           // this ist Referenz auf Lifecycle-Owner ^
}
```