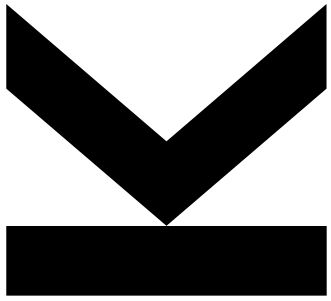
MOBILE COMPUTING (2H, KV)
WS 2019/20

MOBILE APPLICATION DEVELOPMENT



Mobile Web and Native Appliation Development

Karin Anna Hummel (karin_anna.hummel@jku.at)

OVERVIEW

- (Left-over) Positioning / GPS

- Mobile eco-system
- Mobile Web development (in brief)
- Mobile app development

- Break-out session: Towards a good design of a mobile app

karin_anna.hummel@jku.at

LITERATURE

Brian Fliring. Mobile Design and Development

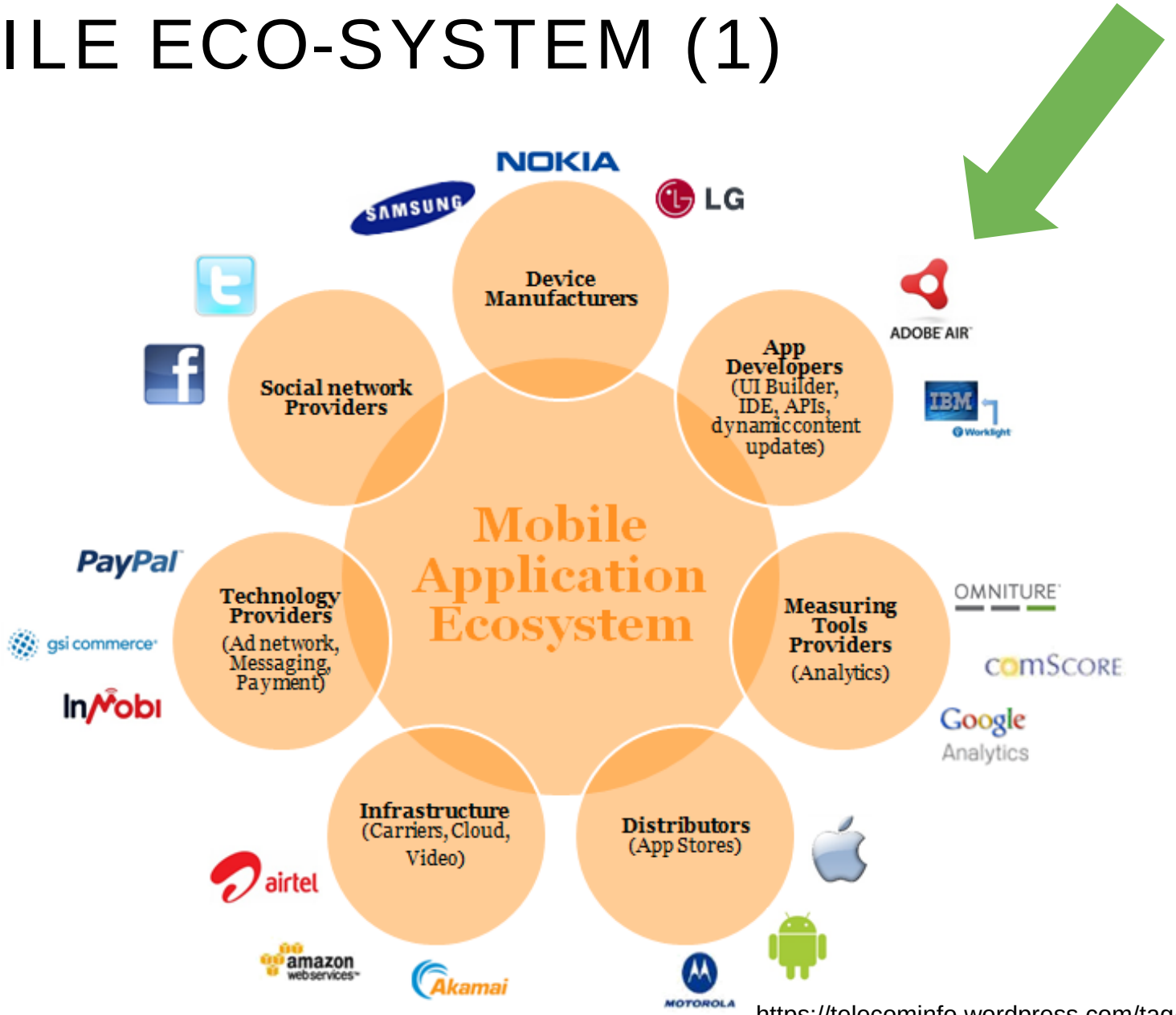
Maximilan Firtman. Programming the Mobile Web

CHARACTERISTICS OF MOBILE APPS

- **It is an all-in-one media center**
  - End-device for streaming and downloading (multi-media) content
  - End-device for generating content: videos, recordings, pictures
- **It is personal**
  - Should support/entertain a particular owner
- **It is ubiquitous and contextual**
  - Carried and used in different context: when, where, physical characteristics of the context
- **It is a primary interface to the Internet and becomes an interface to other appliances such as IoT**
  - E.g., interfacing to the smart home, e-meters, machines in a factory, etc. in addition to general information (e.g., Google maps info)

User interactions are characterized by **short, non-textual interactions**

MOBILE ECO-SYSTEM (1)



https://telecominfo.wordpress.com/tag/mobile-ecosystem/

karin_anna.hummel@jku.at

MOBILE ECO-SYSTEM (2)

- **Mobile device manufacturers, mobile operating system providers**
  - Smartphone, watch, glass – Samsung, Huawei, Apple, etc.
  - Apple/iOS, Google/Android, etc.
- **App developers, framework developers, App distributors**
  - Applications running on the mobile device – e.g., OEBB app
  - App stores
- **Wireless network operators, infrastructure providers**
  - 4G/5G, cloud providers (e.g., Amazon)
  - Analytics providers
- **Service and technology providers (distributed system)**
  - Integrated with a backend system, providing information and support – e.g., OEBB provides train schedules and ticket shop
  - **Social network providers**, **technology providers** (payment, statistics, e.g. Google Analytics)

EARLY MOBILE DEVICES

SMARTPHONES

karin_anna.hummel@jku.at

MOBILE DEVICES

- **Smartphones**
- **Smart watches**
- **Glasses**
- **Tablets**
- **(Notebooks)**

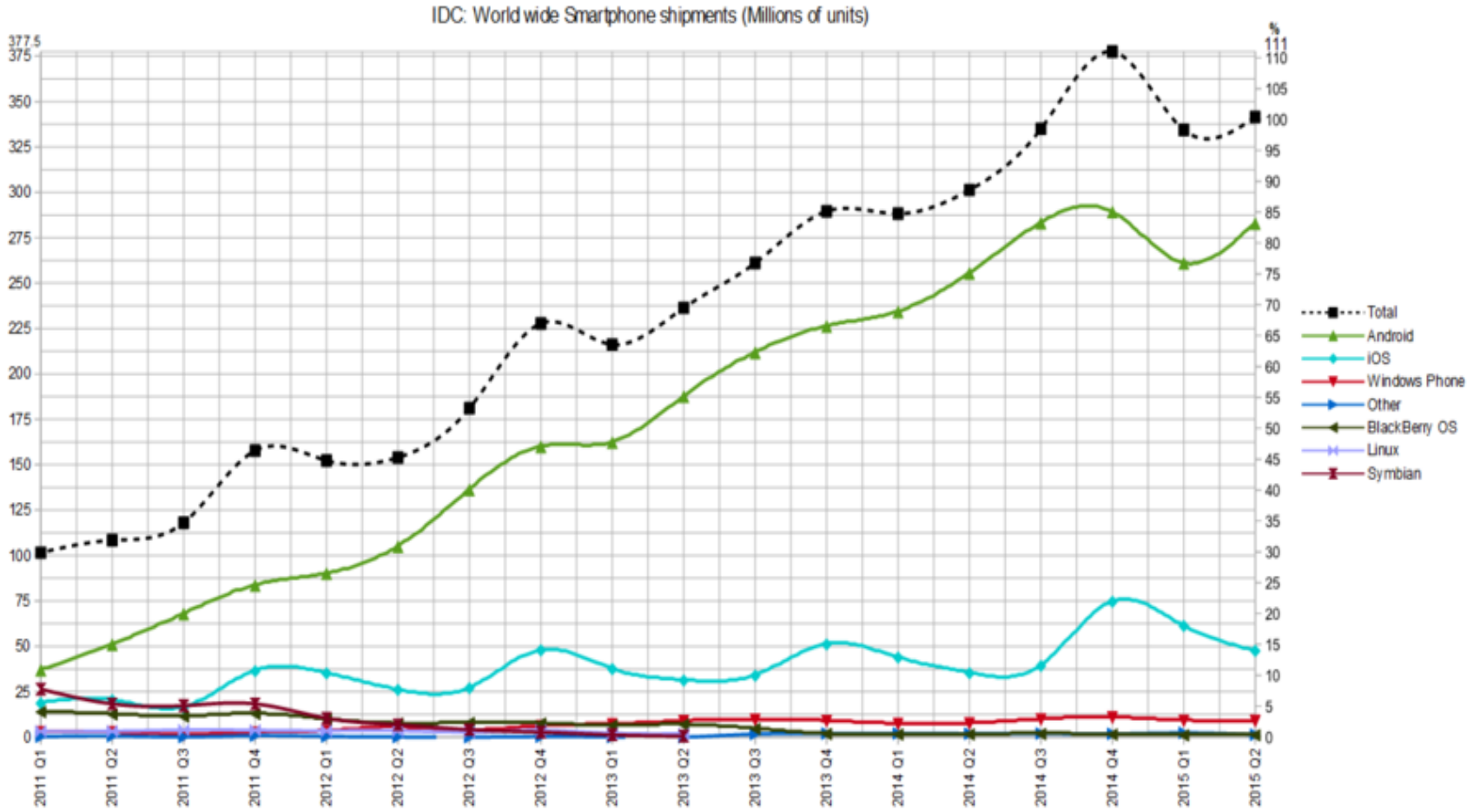
- But also: special **on-board computers in cars, drones, robots, autonomous sailing boats, etc.**

OPERATING SYSTEMS

- **Licensed platforms**
  - **Windows 10 OS (Windows Phone)**: based on MS Win32 API
  - *"The phone that works like your PC"* (Lumina 950, www.microsoft.com)
- **Proprietary platforms**
  - **Palm OS**: based on C/C++, low-end smartphones
  - **Blackberry OS**
  - **iOS:** adapted Mac OS X (based on Unix) for iPhone, iPod, iPad
- **Open source platforms**
  - **Android:** based on Linux kernel, Java-based programming
  - **Symbian**
  - **Linux** (Motorola)

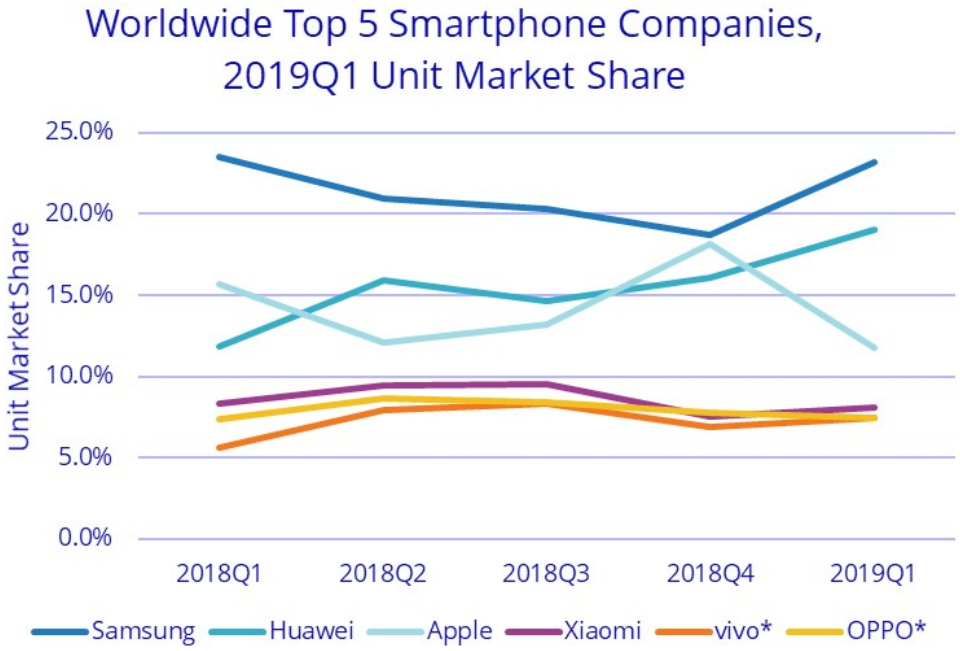
etc.

HISTORY OF MOBILE OS MARKET

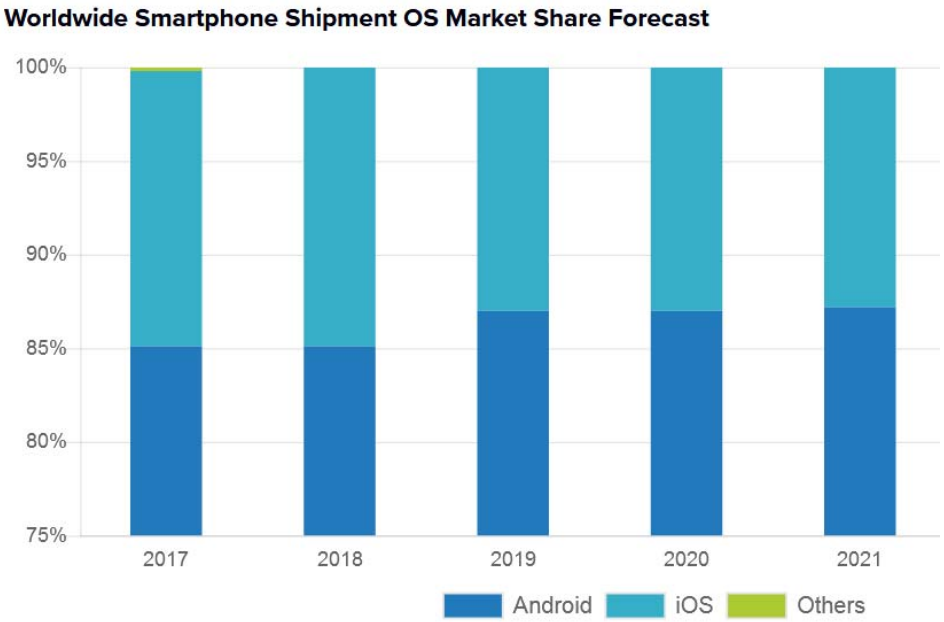


karin_anna.hummel@jku.at

DOMINATING MOBILE DEVICES & OS



Source: IDC 2019



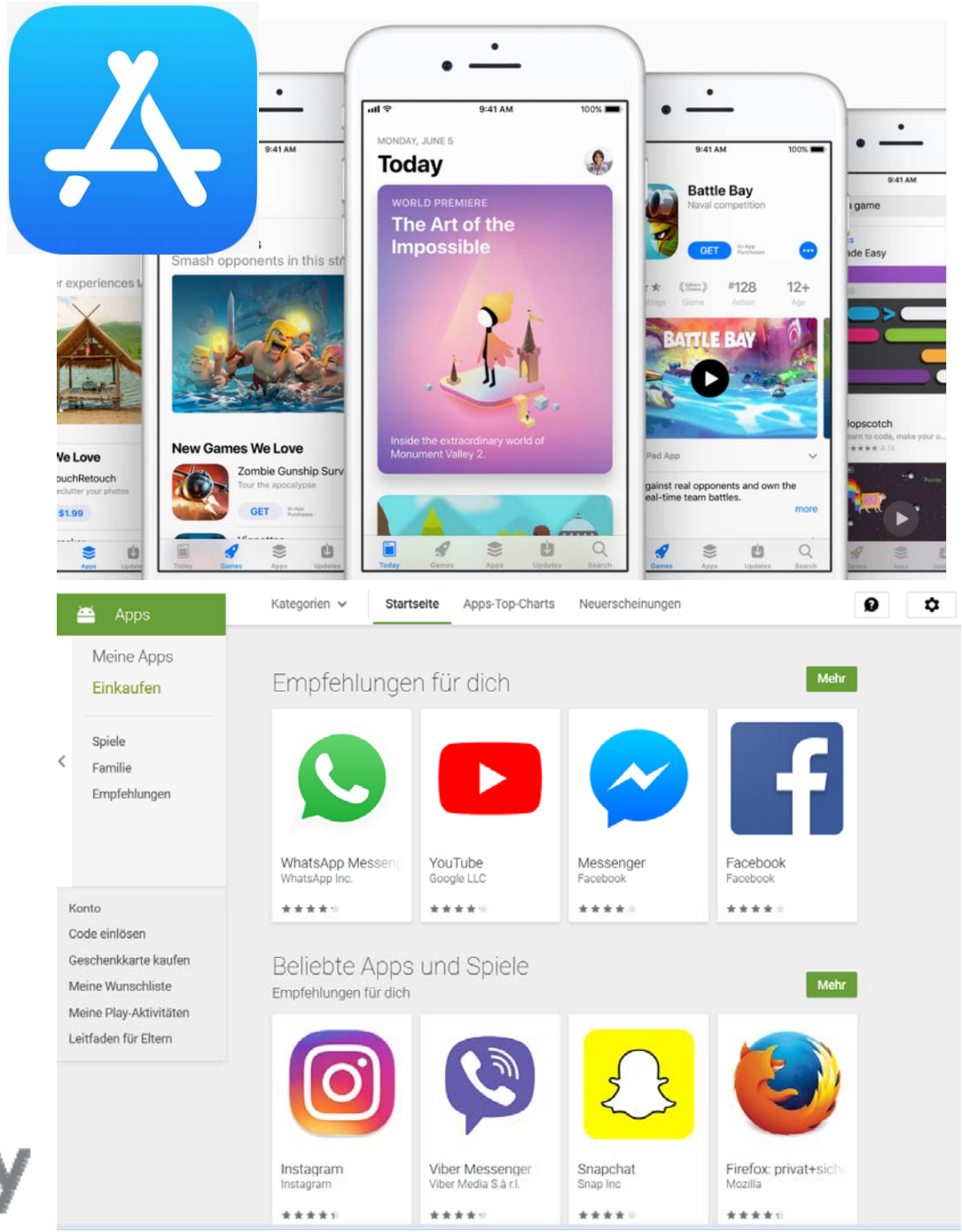
https://www.idc.com/getdoc.jsp?containerId=prUS45042319

karin_anna.hummel@jku.at

APP STORES

Two major platforms to sell / distributed apps

- **IOS: App Store**
- **Android: Google Play**



Google Play

karin_anna.hummel@jku.at

APP DEVELOPMENT OPTIONS

- **Native app development**
  - Apps developed for specific operating systems / platforms
  - (Visible in app stores – important for economic success)
- **Mobile Web development**
  - Web browser is the end tool, no different developments for different platforms – yet, maybe for different Web browsers!
  - Web browser cares about scaling (screen sizes)
- **Hybrid Web/native app development**
  - Partly native apps, partly Web apps
  - Live in app store, but are rendered in a browser, that is embedded in the app (wrappers of existing web page)

DEVELOPMENT – PROS / CONS

- **PROS of native apps**
  - Easy/more options to access device sensors and other functions (cameras, accelerometers)
  - Receiving push notifications (updates)
  - Sharing by using other apps (email, WhatsApp, FB, etc.)
  - Adds can be included (may be blocked in Web browsers)
  - Fast and responsive
  - UI and user experience in compliance with other apps

- **PROS of mobile Web apps**
  - Avoiding multiple code bases (iOS, Android, Windows Mobile) – faster development
  - (Note: possible to build geo-location services with Web apps)

CONTEXT-AWARENESS

- Applications are **used in context** (street, hiking, biking, in class, at home), thus their context changes
  - **User situation** with different characteristics concerning the way the user wants to interact with the device (e.g., when biking, you usually do not want to handle the touch screen)
  - **Physical context** parameters: location, ambient sound and light
- How to know about the context? → **Sensors and context-recognition** software
- Further: Knowing about the **media/technical context**: availability of sensors (camera – which resolution?), connectivity, screen size and orientation

TYPICAL ONBOARD SENSORS

Samsung Galaxy S 10

- Accelerometer
- Barometer, pressure, gyro, geomagnetic sensor
- Fingerprint, iris sensor
- Geomagnetic Sensor
- Proximity Sensor
- RGB Light Sensor
- Camera, microphone, GPS
- …

Apple Watch Series 5

- Accelerometer
- Gyroscope
- Heart rate
- Barometer
- GPS
- …

karin_anna.hummel@jku.at

(MOBILE) DISTRIBUTED COMPUTING

- **Message passing and space-based communication**
  - **Client-server style**: Request-answer communication pattern, e.g., HTTP (Web queries) or using sockets
  - **Peer-to-peer communication**: equal roles using sockets
  - Blackboard-based communication: write information to repository, read from this repository
  - Broadcasting information, information-centric message passing
- **Remote procedure call**
  - Remote invocation of a function on a remote machine, e.g., NFS (Network File System), Google Webtoolkit
- **Code on-demand, mobile agents**
  - Transfer code necessary to perform computation, e.g., Applets
  - Mobile agents: autonomous code migrates between computers

Mobility → intermittent connectivity has to be considered

MOBILE WEB DEVELOPMENT

In a Nutshell

karin_anna.hummel@jku.at

MOBILE WEB APPS

- **Web-apps designed for mobile devices**
  - Displayed by **browser**
  - Simple drill-down architecture, simple presentation of site navigation links, informational with limited interactive elements
  - Application-like experience, alter existing views instead of loading new pages (in contrast to traditional Web page)
  - Usually: single-page model

- **Mix of native and Web technologies**
  - E.g., use JavaScript **API** to take a photo, accessing native code

MOBILE WEB TECHNOLOGIES

- **Components:** mobile device, browser (navigation), content in form of markup information, presentation (look and formatting)

- **Major technologies used**
  - Markup languages: XHTML (HTML based on XML), HTML5
  - Cascaded Style Sheets (CSS)
  - Programming: JavaScript

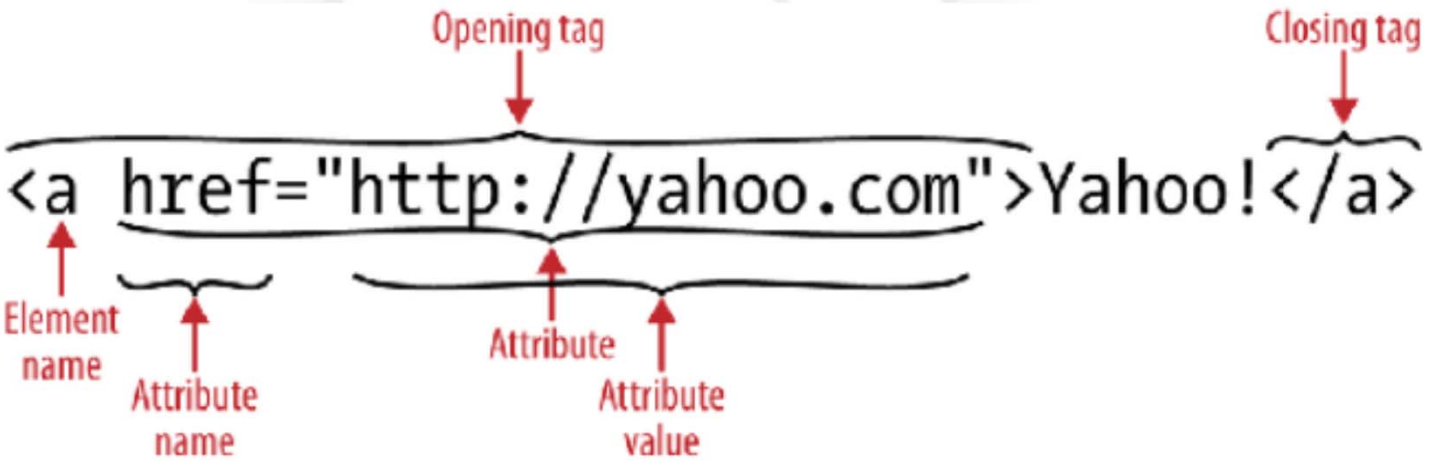
- Hybrid Web app development
  - E.g.: Apache Cordova (formerly PhoneGap): wrap native code to be able to deploy it on various platforms using different frameworks

EXAMPLE HTML

- Hypertext Markup Language

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
 <head>
  <title>sample</title>
 </head>
 <body>
  <p>Voluptatem accusantium
   totam rem aperiam.</p>
 </body>
</html>
```

HTML



karin_anna.hummel@jku.at

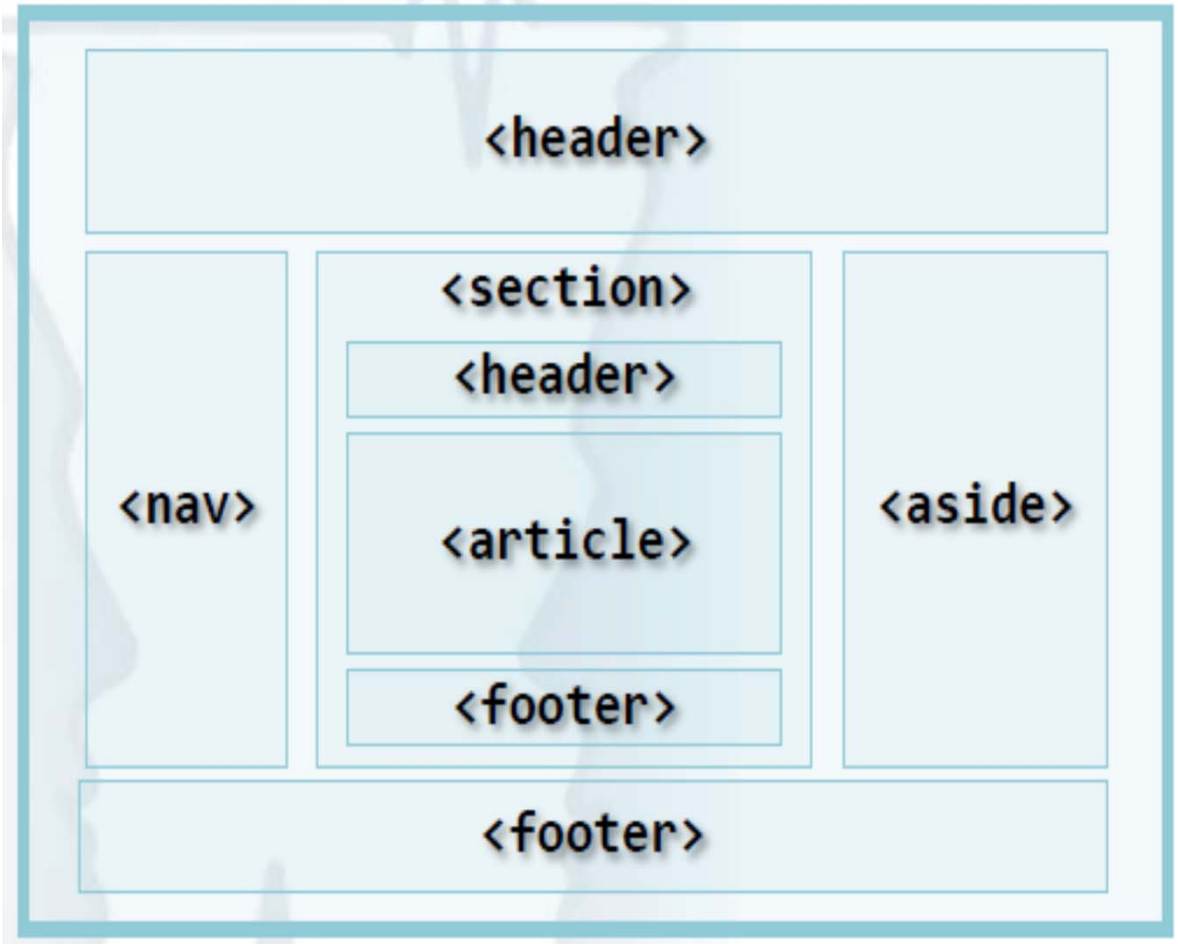
HTML5

https://www.w3.org/TR/2017/REC-html51-20171003/

(W3C recommendation, October 3, 2017)

- Family of Web technologies centered around HTML; reducing the need of plug-ins
- Means to specify:
  - (As with predecessors:) Document structure, text elements, links, images and objects, tables, frames, forms, etc.
  - New layout structure
  - New semantic tags, e.g., <nav> (section of navigation), <article> (text from a news article)
  - Input types (<input>), <canvas> element to draw shapes
  - Embedding of SVG (Scalable Vector Graphics)
  - New video and audio support (<audio> and <video>), Web forms and form validation, **geo-location API**, **offline content and storage**

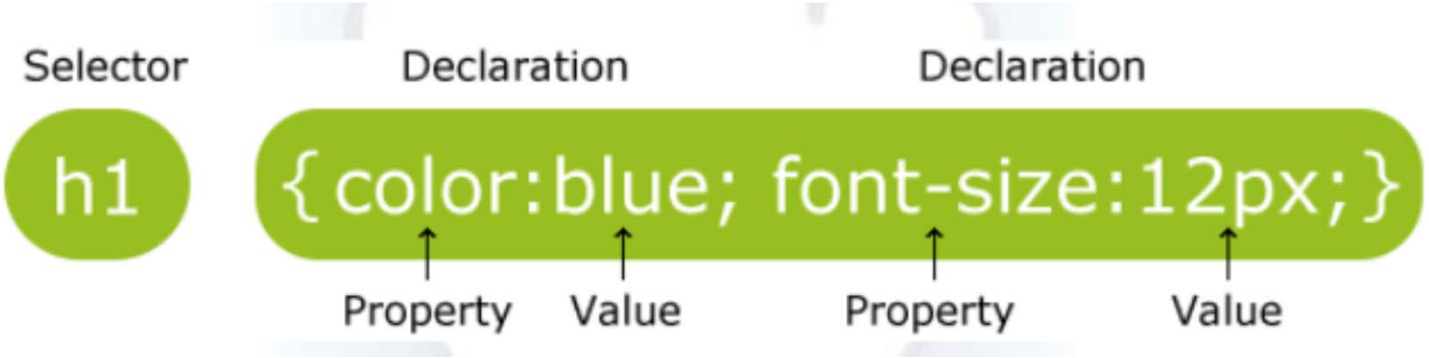
HTML5 – LAYOUT STRUCTURE



karin_anna.hummel@jku.at

CSS (CASCADING STYLE SHEETS)

- Style sheet language specifying the presentation of a document
  `Rule set:` consisting of Selector and all Declarations



Link to an external CSS file:
<link href="path/to/file.css"
rel="stylesheet" type="text/css">

```
h1 { color: white;
  background: orange;
  border: 1px solid bla
  padding: 0 0 0 0;
  font-weight: bold;
}
```

CSS 3

- Multiple backgrounds, background size
- Transitions and transforms (3D transforms), e.g. rotations
- Visual effects, gradients (smooth transitions), opacity, and text overflows
- Border radius, image
- Box and text shadow
- Animations

Etc.

```
background-image: url(sheep.png), url(grass.png);
```

karin_anna.hummel@jku.at

JAVASCRIPT

- Scripting language widely supported by browsers
  - Code runs on the client browser (not the server)
- Code parts can be included from external files or inline with HTML code
- **Code is run when events fire**
  - Page loaded
  - Timer events
  - User interaction (clicking, hovering)

- Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_head

karin_anna.hummel@jku.at

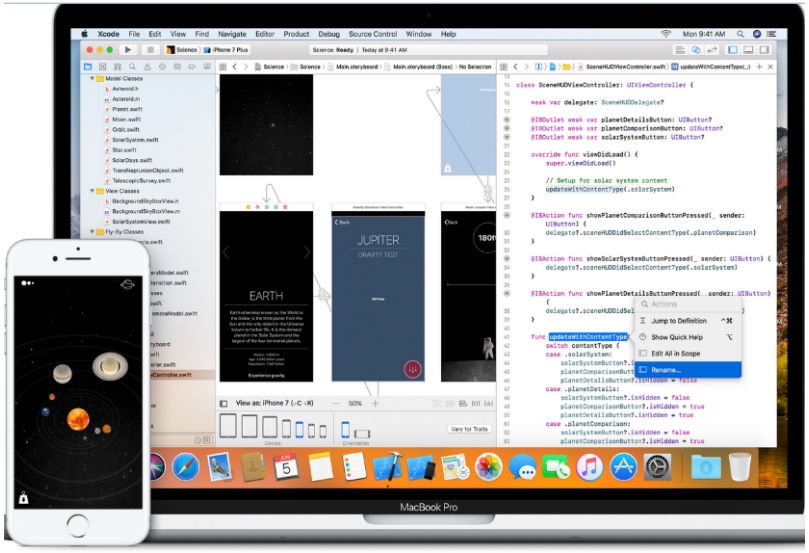
NATIVE APP DEVELOPMENT
Principles

karin_anna.hummel@jku.at

MOBILE NATIVE APPLICATIONS

Major platforms

- **IOS**
  - Apple developer program
    - Tools and resources for free, 99 USD pear year in case you enrol to the Apple Developer Program
  - **Xcode** (IDE), program in **Swift** (formerly Objective-C)
- **Android**
  - Google Android developers
    - Google developer account: 25 USD when registering
  - **Android Studio** (IDE), program in Java



karin_anna.hummel@jku.at

NATIVE APP DEVELOPMENT

- **Cross-platform development for each platform separately**
  - E.g., iOS and Android development
  - Typically, companies install development expert teams for each platform

- **Cross-platform development for multiple platforms**
  - Examples:
    - Xamarin (Microsoft): build programs in C# for iOS, Android, Windows Mobile
    - Titanium: JavaScript based SDK to build native iOS, Android apps (IDE: Eclipse)
    - React Native: JavaScript and React (declarative programming) to compile for iOS and Android
    - Ionic …

iOS

IOS

https://www.apple.com/ios/

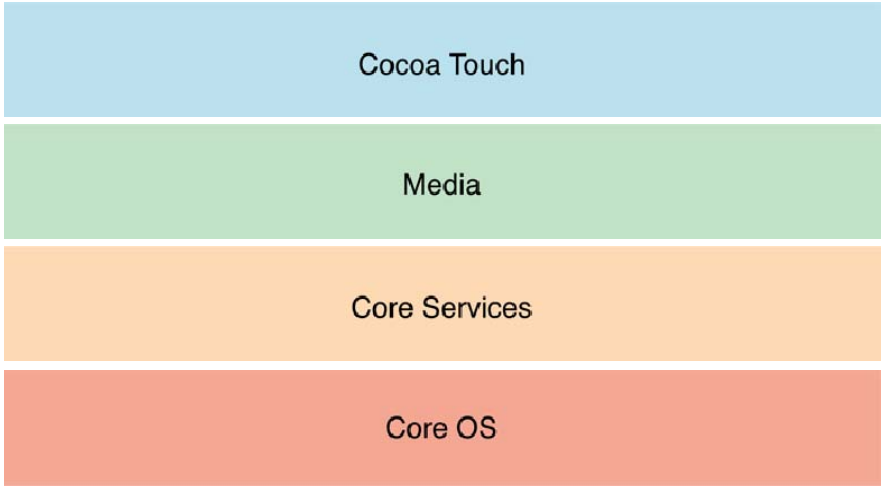
- **Unix-like operating system** (first release 2007, Darwin core), 32-bit and 64-bit ARM processor architecture, multitasking (with background activities enabled)
- Designed for iPad, iPhone, iPod
- User interface framework: Cocoa Touch
- Siri: voice input-driven assistant
- **Programming in: Swift**, (Objective-C, C/C++), IDE: Xcode
  - Swift: C-like language with automatic checks (memory, overflow, etc.), tokens
- Package manager: iTunes
- Application ecosystem: App Store

SWIFT program snippet:

```
var myVariable = 42
myVariable = 50

let myConstant = 42
```

IOS ARCHITECTURE



- **Cocoa Touch:** Abstraction layer for applications
  - **Multi-touch** events and controls, accelerometers, View hierarchy, localization, gesture recognizers, etc.
  - **Multitasking** support, printing support, data protection, Apple push notification service, local notifications, alerts, file sharing, peer-to-peer over Bluetooth, etc.
  - **Frameworks** for various purposes, e.g. `AddressBookUI.framework`
- **Media Layer:** Camera, photo library, remote control for headsets, etc.
- **Core Services Layer:** access to OS services, e.g., threading, network services, SQLite, address book, file access
- **Core OS:** OS X kernel, power management, file system, Bonjour, sockets, security, etc.

ANDROID

https://www.android.com/

- Aim: open and extendable operating system for mobile devices
- Developer: Google and Open Handset Alliance (originally Android Inc.)
- Based on **Linux kernel**, multitasking, multiuser platform, advanced power management, open source, **supports various architectures**: 32-bit/64-bit ARM, x86, etc.
- Processes have own virtual machine and can only access resources if granted
- Programming in: mainly Java, IDE: Android Studio
- Since 2008
- Application ecosystem: Google Play Store

ANDROID – HISTORY

							
Alpha, Beta	**Cupcake Android 1.5**	**Donut Android 1.6**	**Eclair Android 2.0/2.1**	**Froyo Android 2.2.x**	**Gingerbread Android 2.3.x**	**Honeycomb Android 3.x**	
v1.0,v1.1							
2008	04/2009	09/2009	10/2009	05/2010	12/2010	02/2011	
							
	Ice Cream Sandwich Android 4.0.x	**Jelly Bean Android 4.1.x**	**KitKat Android 4.4.x**	**Lollipop Android 5.0**	**Marshmallow android 6.0**	**Nougat android 7.0**	
	10/2011	07/2012	10/2013	11/2014	10/2015	08/2016	08/2017
API level	14-15	16-18	19	21-22	23	24-25	26-27

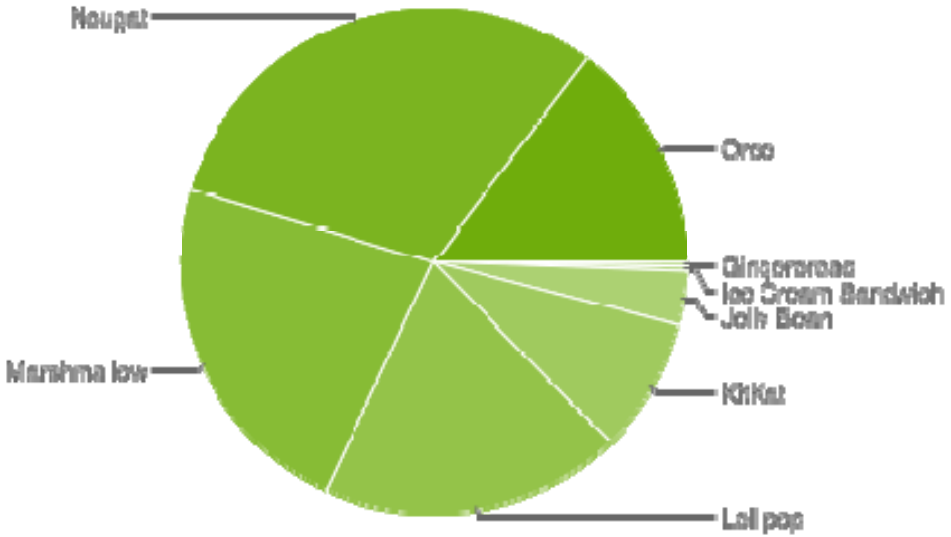
			
Oreo android 8.x	**Pie android 9.x**	**Queen Cake ? android 10**	
08/2017	08/2018	09/2019	
26-27	28	29	

karin_anna.hummel@jku.at

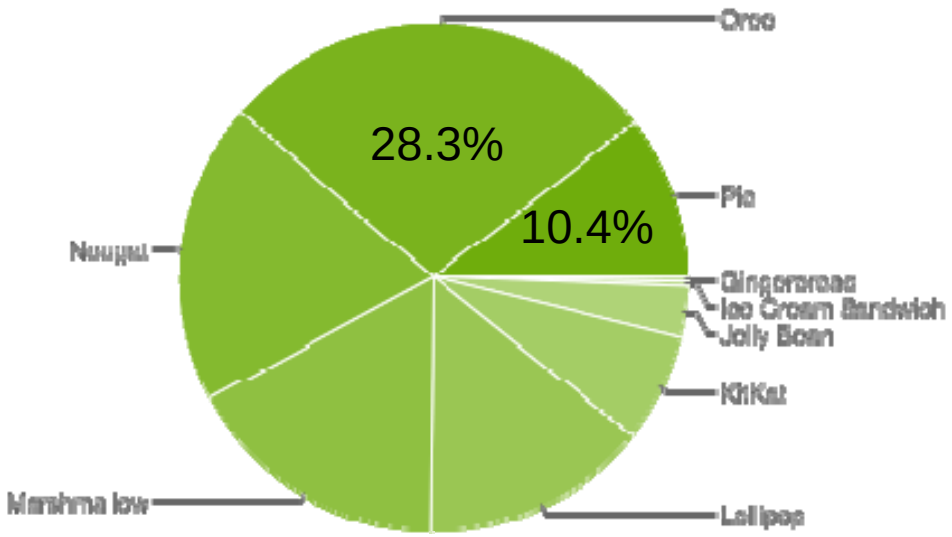
ANDROID - VERSIONS

- Fast innovation cycles

Supported platform versions, October 2018



Supported platform versions, October 2019

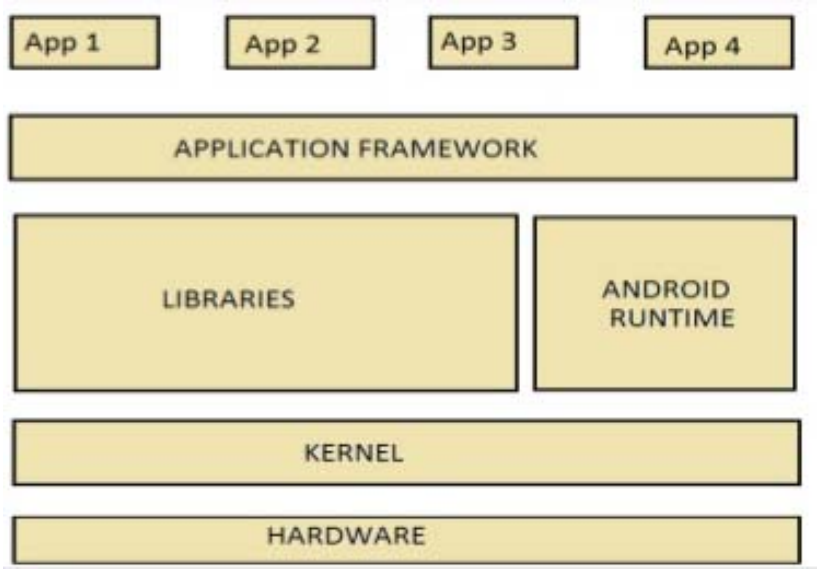


https://developer.android.com/about/dashboards/index.html

karin_anna.hummel@jku.at

ANDROID SYSTEM ARCHITECTURE

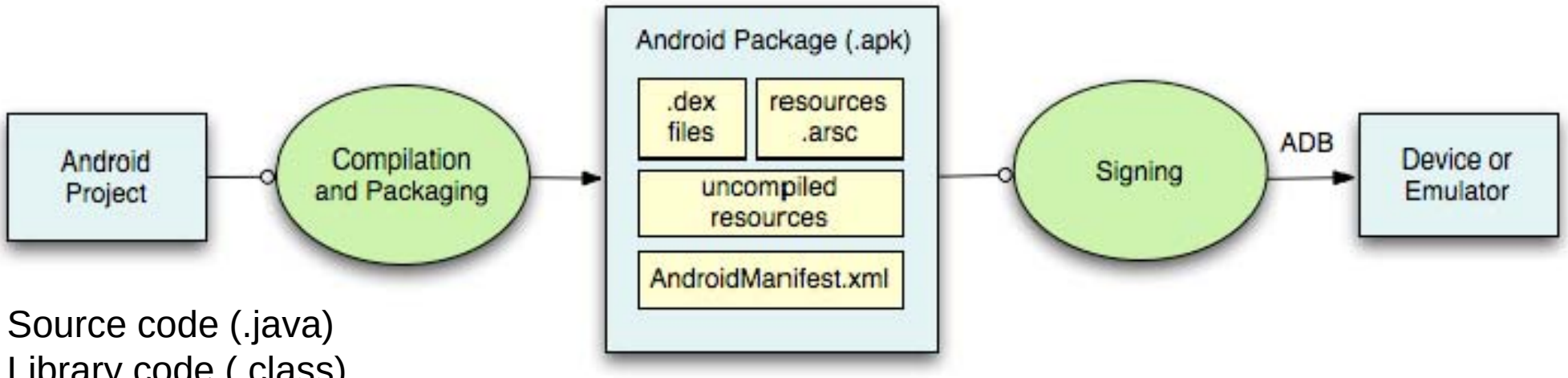
- **Apps**
  - Native Android and third party apps
- **Application framework**
  - Several “managers” such as: power manager, **activity** manager, etc.
- **Android Runtime**
  - JVM for Android (past: Dalvik VM, from 5.0 on: ART - Android Runtime)
- **Libraries** (with APIs)
  - libc (C library), SQLite (database), SSL (Internet security), etc.
- **Kernel**
  - Drivers: display, camera, BT/BLE, power management, Wi-Fi, etc.
  - Memory, process, device mgmt.



karin_anna.hummel@jku.at

ANDROID DEVELOPMENT

- **Cross-platform development**



Source code (.java)
Library code (.class)
Images

http://developer.android.com/guide/

ADB: Android Debug Bridge

CORE ANDROID CONCEPTS

- **Core concepts**
  - **Activities**
  - Services, broadcast receivers, content providers
  - All concepts have a “life cycle” (running in foreground, paused, etc.)
- **User Interface:** Basics (Button, TextView), Fragments, Views and View Groups, Menu – **event-driven programming**
- **Layout:** Layout XML files
- **Intents:** communicate between / start new Activities
- **Processes and threads**: asynchronous execution in background (AsyncTask)
- **Specific APIs:** Communication API, Location API
- **Specials:** Toast, logging, etc.

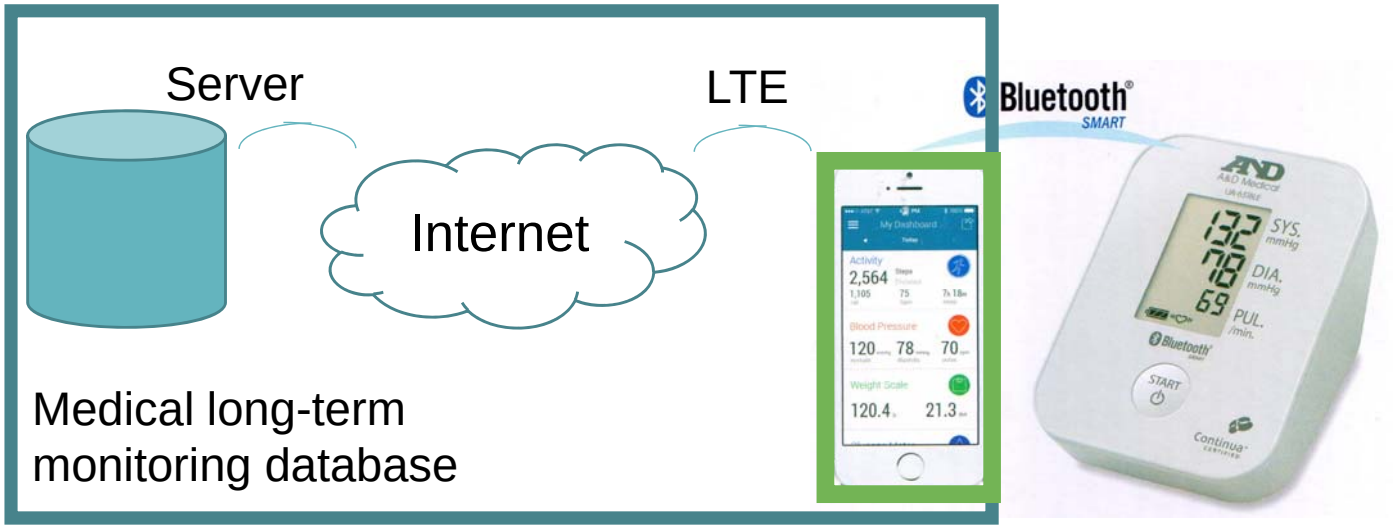
Where to get information? → http://developer.android.com/

WHEN DESIGNING AN APP

karin_anna.hummel@jku.at

TYPICAL MOBILE APP

- **Client (frontend): mobile app**
  - Mobile app opens an HTTP connection to the server
  - Exchanges data through this connection, e.g., using REST services
- **Server (backend): Web server, server in a cloud**
- Example: Blood pressure monitoring
  - Blood pressure monitor / **smartphone used to collect blood pressure data and upload it through REST services to a REST Web server**



karin_anna.hummel@jku.at

DESIGN-QUESTIONS (1)

BE PRECISE!

- **Who is the user?**
  - Business people, teenagers, mothers, sportive persons?
    - Categories: age, profession
  - Important, because the application will look significantly different depending on the user
- **In which situation will the app be used?**
  - On the move, with friends/alone, @home, @work, on the bus?
- **Why is the app used? What are the functional expectations?**
  - Retrieving information, guidance?
  - Increasing productivity (business)
- **I/O: How is the input received and output generated?**
  - User interface: at hand, landscape/portrait orientation, with tactile support (vibration), with sensory support (accelerometers, external sensors), interaction patterns?
  - Notifications?

DESIGN-QUESTIONS (2)

BE PRECISE!

- **App category? → Web or native development; comparison to related apps (novelty!)**
  - Information services: news, browsing, search, dictionary, library, encyclopedia etc.
  - Location services: map, navigation
  - Gaming and entertainment: music and video players, games
  - Social networks and IM (instant messaging)
  - Health services: measurements
  - Etc.

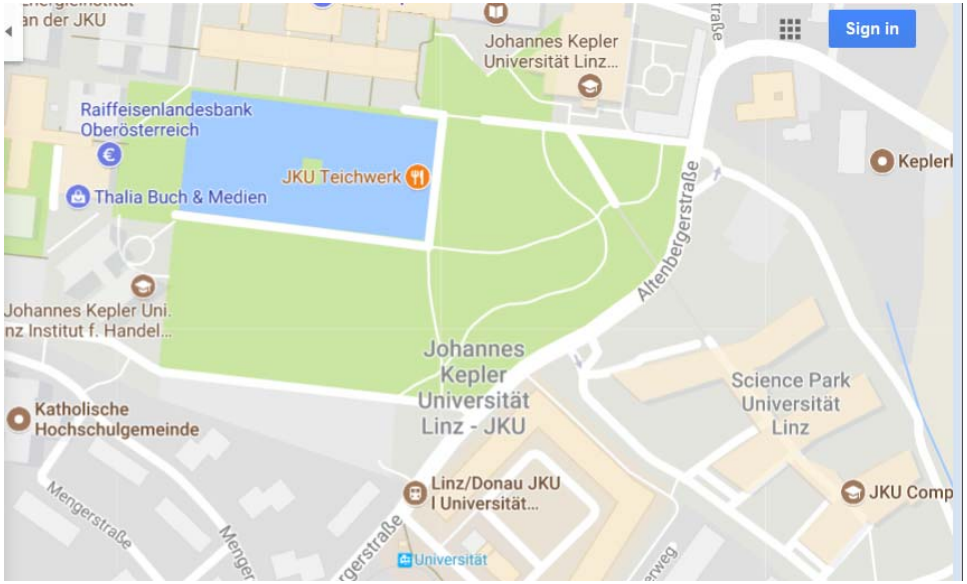
POSSIBLE COMMUNICATION

- **Connecting to an infrastructure**
  - E.g.: LTE, Wi-Fi campus network
- **Device-to-device communication**
  - E.g.: Wi-Fi direct (“ad-hoc mode on the smartphone”, Bluetooth)

CONTEXT AWARENESS

- **Geo-location context**
  - GPS
  - Network provided locations
  - Displayed on a **map**?
  - Including **navigation** of the user?
- **Time**
  - **Notifications**
  - Validity of an event
- **Noise/light**
  - App adaptations
- **Camera/vision**
- **Accelerometers/gyro**
  - User input (e.g., game)

etc.



karin_anna.hummel@jku.at

POSSIBLE USER INPUT

- **Numeric/alphanumeric keypad**
  - Physical and virtual (on screen)
- **Touch**
  - Single or multi-touch
- **Handwriting recognition, voice recognition (micro)**
- **Gestures, motion**
  - Point to some object (gyroscope)
  - Step counter, exercise recognition (accelerometers)
- **Camera**
  - Take a photo and analyse the picture (e.g., sight recognition), OCR (optical character recognition)
    e.g.: https://android-arsenal.com/details/3/6770
  - Scan a QR code

POSSIBLE OUTPUT

- **Visual on screen**
  - Change of color, text, diagram or picture being displayed, etc.
  - Pop-ups and dialogues
- **Vibration of the phone**
  - E.g., alarm is triggered by a timer
- **Sound played**

- Or: external displays (glasses), smartwatch, etc.

TECHNICAL SPECIFICATION

- **System overview**
  - Components (e.g., smartphone, backend server, external device)
  - Connectivity (e.g., Wi-Fi)

- **Design of mobile app**
  - **User interface**
  - App logic
  - Information extracted from on-board/external sensors
- **Design of the backend server and database**
  - Feasibility (where should the server run?)
- **Design of the communication protocol (sequence) between mobile app and server**
  - **Including error handling !**

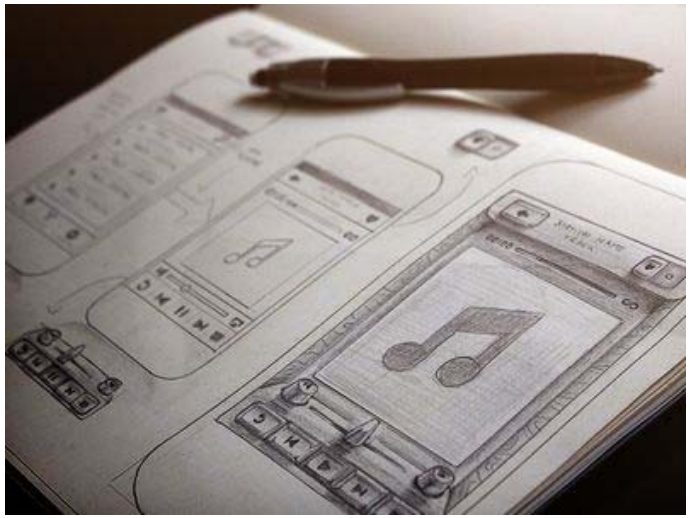
PROJECT SCHEDULE

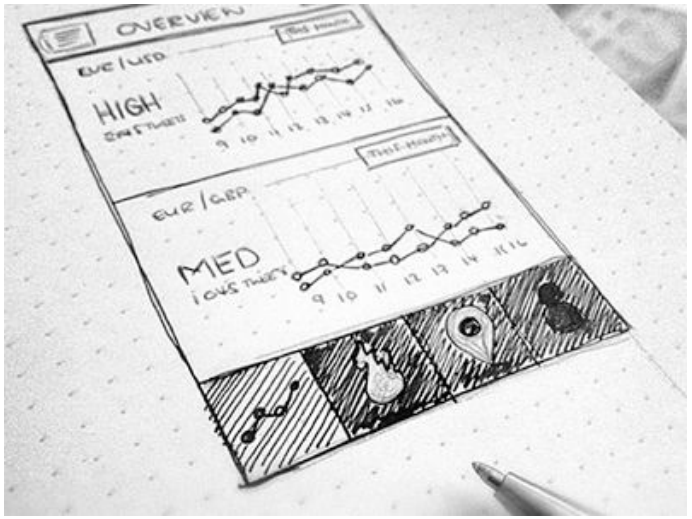
- **Start with the end-date**: Your final presentation

- **Rough time plan** (can be detailed by yourself on-demand) with deadlines for each of the following steps:
  - First **design** – 6th of Nov. (student presentations)
  - Basic training – familiarize yourself with the technologies and app development (e.g, our Android lectures)
  - **Technological feasibility** (most difficult parts should be tested first, i.e., whether they can be used as expected) – mid to end of Nov.
  - UI development (mock-ups) – mock-up student presentations
  - Develop/program specific functions
  - Integrate
  - Test and evaluate (e.g., with a small user study)
  - Upload project and present your results at the end of the lecture

WHERE TO START

1. **The story**
2. **The system architecture (including communication and sensor input, backend)**
3. **Sketch the UI (keep the number of windows low !!!)**





QUESTIONS?

karin_anna.hummel@jku.at