



Don't Panic

MOBILE DEVELOPER'S GUIDE TO THE GALAXY



17th Edition October 2017

This Developer Guide is licensed under the
Creative Commons BY 2.5 License.

Please send your feedback,
questions or sponsorship requests to:
mobiledevguide@open-xchange.com
Follow us on Twitter: [@MobileDevGuide](https://twitter.com/ MobileDevGuide)

Art Direction and Design by

Cornelius Kwiecniak
Mladenka Vrdoljak

Editors:

Marco Tabor
Mladenka Vrdoljak

www.mobiledevelopersguide.com

Mobile Developer's Guide

Contents

- 1 Prologue
- 4 The Galaxy of Mobile: Past, Present and Future
by Robert Virkus
- 16 From Idea to Prototype
by Andrej Balaz
- 44 Android
by Vikram Kriplaney, André Schmidt & Daniel Böhrs
- 64 iOS
by Alex Repty
- 78 Going Cross-Platform
by Robert Virkus
- 86 Mobile Web
by Ruadhan O'Donoghue
- 126 Enterprise Apps
by Ian Thain & Davoc Bradley
- 140 Mobile Gaming
by Oscar Clark
- 168 The Internet of Things (IoT)
by Alex Jonsson & Aaron Ardiri
- 178 Artificially Intelligent Apps
by Robert Virkus

- 
- 192 Security & Privacy
by Dean Churchill & Neil Cook
 - 210 Accessibility
by Sally Cain
 - 234 Testing
by Julian Harty & Marc van't Veer
 - 260 Mobile Analytics
by Julian Harty
 - 274 Collecting & Understanding User Feedback
by Julian Harty
 - 288 Monetization
by Michel Shuqair
 - 306 Epilogue
 - 308 About the Book

Prologue

Wow, it is the 17th edition already. Time flies. Long-time followers and fans of this book might have recognized that with this edition, the logo of Enough Software as publisher has disappeared from the cover. That is because our team has fully joined Open-Xchange last year - we look forward to improving the world of open source communication and collaboration tools. And we are happy that OX recognizes the value of this book project as a source of knowledge for anyone willing to improve his or her apps or planning to enter the apps market.

As usual, you will find all the basics along with advanced topics about mobile app and web development in this edition. All chapters have been reviewed, updated and often extended once again to make sure it contains all today's relevant stuff. With this release we have added another 50 pages of content: The mobile web chapter has been completely replaced with fresh content and we included two new chapters including one about artificial intelligence and the role it plays within the mobile ecosystem.

Of course this does not mean that there is no room for even more enhancement, so please let us know what content you are missing. Or even better: Get involved and share your knowledge with the community by becoming a contributor for the next edition. You know where to find us :)



Cheers,
Robert + Marco / Open-Xchange
Bremen, September 2017

PS: Please follow us on Twitter [@MobileDevGuide](#) and visit mobiledevelopersguide.com to obtain the electronic edition of this booklet.

PPS: In case you know a company interested in sponsoring the printing of upcoming editions, please let us know.

ART





The Galaxy of Mobile: Past, Present and Future

In the good old days we had lots of choice in the mobile space - feature phones & smart phones, Symbian, BlackBerry, webOS, bada, Firefox OS, Ubuntu Touch, Windows Phone, Sailfish OS, Tizen and of course Android and iOS, etc.

Today, we have a duopoly of Android and iOS. While Sailfish and Tizen and few others still do exist, they play a niche role - same for Windows after Microsoft retreated from the mobile market. Retrospectively the same has happened like in the home computer industry of the 1980's - one player - in this case the IBM-compatible PC - took the majority share and left only pieces to the other competitors.

In the mobile industry Google's Android operating system is the major player, with Apple following with a comparatively small but very lucrative share. Without any paradigm shift - like perhaps voice user interface represent - we expect this situation not to change for the foreseeable future.

Stand Out in a Crowded Market

With the increasing competition in the apps space there are various aspects that are worth considering:

- Experiences can carry across a variety of form factors - may it be in-car-systems, TVs, PCs, game console, augmented reality or voice-enabled smart home systems. As mobile technology moved to many systems, you can use your existing app development skills to reach these form factors. But make sure to adapt to each platform in

the best possible way, do not limit yourself to the least common denominator!

- Users seem to be less likely to try and install new apps, therefore existing apps will increase their features - moving from a single-purpose to a multi-purpose app world.
- With multi-purpose apps, extensions play an important role. Instead of creating and maintaining your own app you can also extend existing apps such as WeChat, Facebook Messenger, Microsoft Office, or even system extensions like the iOS 11 FileProvider or the Today app extensions. Search for "zero UI" and "atomized apps" to learn more.
- Take notifications seriously and make sure to add interaction options to your notifications.
- Think about creating extension points in your app to allow others to get their services into your app.
- Take your target regions into account. In many regions feature phones are still the dominant platform, making good old Java MIDlets¹, USSD² and STK³ or SMS⁴-based services options worth considering.
- Driving engagement is - as always - critical. One of the biggest driver for apps is communication and socializing. Instead of creating your own solutions you can consider adding support for *matrix.org* or *mastodon.social*, for example.
- If you consider adding artificial intelligence to your app - and you should - please read our new chapter about AI apps in this edition.

¹ en.wikipedia.org/wiki/MIDlet

² en.wikipedia.org/wiki/USSD

³ en.wikipedia.org/wiki/SIM_Application_Toolkit

⁴ en.wikipedia.org/wiki/SMS

The focus of this book is on developing mobile apps, which encompasses a number of phases including: planning and specification, prototyping and design, implementation, internal testing and deployment, deployment to an app store, discovery by users, installation, use and feedback. Ultimately, we want our users to enjoy using our apps and to give us positive ratings to encourage other users to do likewise.

Keep reading to learn how to develop apps for the major platforms. Should this be the first time that you have considered getting involved, do not delay; mobile has become the predominant form of computing in many areas already. At a global scale, mobile web usage overtook desktops⁵. Same applies for games: Mobile is generating more revenue than any other gaming market today⁶. And at least in the U.S., time spent on mobile app usage even surpassed the good old TV⁷.

While developing mobile apps shares many common features with developing other software, it has specific characteristics. We will cover some of these next.

⁵ www.telegraph.co.uk/technology/2016/11/01/mobile-web-usage-overtakes-desktop-for-first-time

⁶ www.superdataresearch.com/market-data/market-brief-year-in-review

⁷ dminc.com/blog/mobile-app-usage-surpasses-tv

How to Service Mobile Devices

There are several ways to realize a mobile service:

Native Apps

A native app is programmed in a platform specific language with platform specific APIs. It is typically purchased, downloaded and upgraded through the platform specific central app store. Native apps usually offer the best performance, the deepest integration and the best overall user experience compared to other options. However, native development is often also the most complex development option. When starting new apps you should consider using Kotlin for Android and Swift for iOS, rather than Java and Objective-C. Find further information on how to get started in the dedicated Android and iOS chapters.

Websites & Web Apps

Websites or - increasingly - so called single page applications are written in a variety of languages and use HTML and CSS for rendering. Consider using progressive web apps⁸ and configure them for iOS desktop pinning⁹.

Web apps run without app store, so you are independent of app stores which is both good because you are not limited by the app store and bad because it is harder for your users to find you.

Of course, you also find a dedicated chapter on mobile web development in this book.

⁸ developers.google.com/web/progressive-web-apps

⁹ developer.apple.com/library/content/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html

Cross-Platform Apps

There is a multitude of cross-platform services around that provide of write-once run-everywhere scenarios. Even when dealing with only two dominant platforms, cross-platform tools can help you to update and maintain your services with less effort. Read the cross-platform chapter to understand your options in this regard.

SMS, USSD and STK

Simple services can be realized with SMS, USSD or STK. Everyone knows how SMS (Short Message Service) text messaging works and every phone supports SMS, but you need to convince your users to remember textual commands for more complex services. In the UK in many towns parking charges can be paid using SMS messages.

USSD (Unstructured Supplementary Service Data) is a GSM protocol used for pushing simple text based menus, the capabilities depend on the carrier and the device. In Sri Lanka, visitors can receive a free SIM card which is registered using USSD menus.

STK (SIM Application Toolkit) enables the implementation of low-level, interactive apps directly on the SIM card of a phone. STK may appear irrelevant when so much focus is on smartphone apps; however, for example, M-Pesa is an STK app which is transforming life and financial transactions in Kenya and other countries.¹⁰

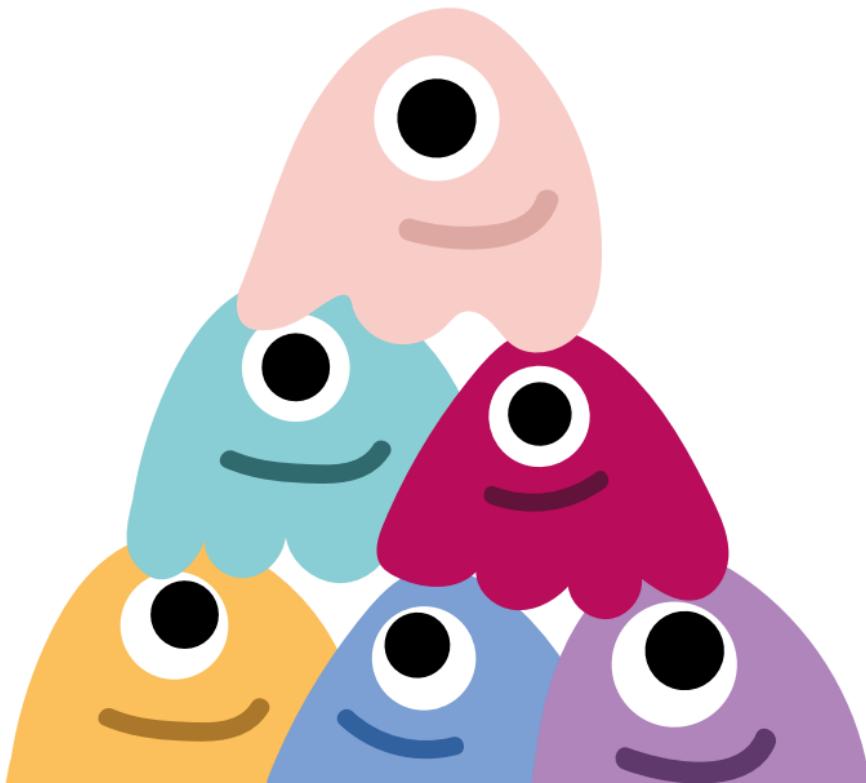
¹⁰ mpesa.in

Duopoly

The duopoly of Android and iOS has been further cemented in 2017 with Android and iOS controlling 99.8% of the worldwide market share in smartphone sales.

Platform	Market Share Q1 2015	Market Share Q1 2016	Market Share Q1 2017
Android	78.9%	84.1%	86.1%
iOS (Apple)	17.9%	14.8%	13.7%
Other	3.3%	1.1%	0.2%

(Source: gartner.com/newsroom/id/3725117 and gartner.com/newsroom/id/3061917)



Niche Players

Why bother serving niche players? Because those niches will protect you from competition, because a mobile niche may be dominant or important on other form-factors - such as Windows on PC or Tizen on TV systems and smartwatches -, because they offer a truly open ecosystem - like Sailfish, and because they provide a better foundation for the future without all that baggage from the past - such as Fuchsia.

The regional market share of each platform varies significantly. In a world where localized content is increasing in importance, it is vital to know the details and characteristics of your target market. For example, China is the largest smartphone market today, but Chinese Android handsets are typically based on the Android Open Source Platform (AOSP) and come without the Google Play Store or the Google Mobile Services. At the same time, Apple is especially strong in the U.S.: In March 2017, 44.5 percent of U.S. smartphone subscribers were using an iOS device¹¹.

To find out about market share in your target region, check out online resources such as Netmarketshare¹², comscore¹³,

¹¹ Nevertheless, Android is leading in the US as well holding 53.4 percent of the market, see statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states

¹² www.netmarketshare.com

¹³ www.comscore.com/Insights/Data-Mine

StatCounter¹⁴, VisionMobile¹⁵, Gartner¹⁶, Statista¹⁷ or Kantar Mobile World Panel¹⁸.

Windows

Windows Phone has given important impulses to the mobile ecosystem. Their "flat design" patterns have been adapted on iOS and Android as well and for some years it seems as if a Windows-based operating system on smartphones might become the third attractive platform for app developers. But in spite of Microsofts and Nokias enormous and cost-intensive efforts, its world wide market share decreased again in the past years and never reached a level where masses of developers considered it as a viable alternative. So Microsoft tried to change the rules with Windows 10 - now you can develop the very same app for both PC and Mobile (and for IoT, HoloLens and Xbox, too). However, Windows in the mobile ecosystem is only relevant for tablets today. Microsoft realized this some time ago and sold the Nokia brand to the Chinese phone manufacturer Foxconn in 2016 who are now releasing Android-based Nokia phones. New impulses are expected to come from Windows on ARM and a rumored ultra-mobile PC generation.

Tizen

Tizen¹⁹ has enjoyed quite a success in the smartwatch market, however none of Samsung's four Tizen-based phones has found its way to the European or US market.

¹⁴ gs.statcounter.com

¹⁵ visionmobile.com

¹⁶ gartner.com

¹⁷ statista.com/markets/418

¹⁸ kantarworldpanel.com/global/smartphone-os-market-share

¹⁹ tizen.org

Seemingly gently yet continuously pushed forward by Samsung and Intel, Tizen aims to power also TVs, tablets, netbooks and in-vehicle infotainment systems.

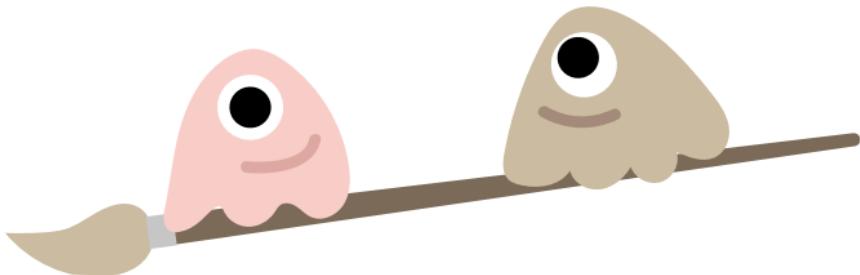
Typical Tizen apps are web based, but you can also create native C-based apps. Start your Tizen journey on *developer.tizen.org* and *developer.samsung.com/gear*. For latest news and rumors, visit *tizenexperts.com*.

Sailfish OS

Although Jolla²⁰ - the company behind the Sailfish OS²¹ - stopped producing devices themselves, the open OS is still being pushed forward. Start developing for Sailfish OS by visiting *sailfishos.org/develop*. One easy path to Sailfish is to use your existing Android app, but you can also create native Apps using Qt/C++ or even Python.

Fuchsia OS

Google is researching a new real-time operating system called Fuchsia²² that is not based on Linux but rather on a custom microkernel. Interestingly enough, Google released a cross-platform development framework called Flutter²³ that allows you to build for Android, iOS and Fuchsia at the same time.



²⁰ jolla.com

²¹ sailfishos.org

²² en.wikipedia.org/wiki/Google_Fuchsia

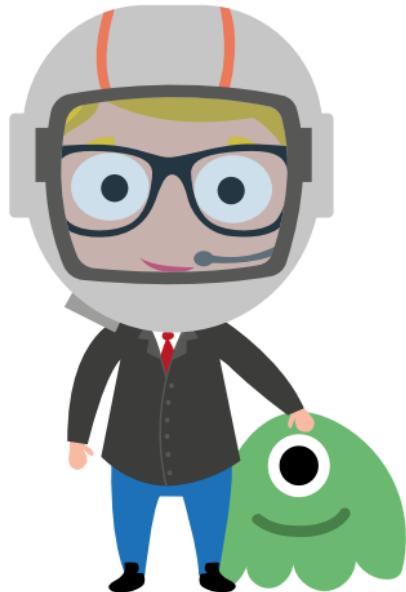
²³ flutter.io

Feature Phone Platforms

While smartphones generally get the most news coverage, in some parts of the world feature phones are still pretty relevant. Even on a global level 22% of all phones sold have still been feature phones in Q1 2016²⁴, with an install base much higher than that. However, Android is increasingly taken over the low-cost handset market so the future of this platform looks dim.

The big players in the feature phone market also had to realize this: Nokia shut down their feature phone app store in 2015.

While you can develop native apps for feature phones when you have close relationship with the vendor, you typically develop apps using Java ME or BREW for these phones.

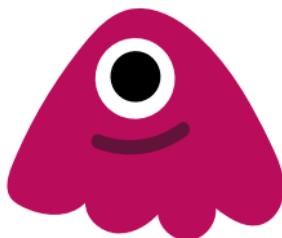


²⁴ www.statista.com/statistics/617945/feature-phone-smartphone-share-of-global-mobile-phone-sales

Flamewars

As developers, we tend to have a passion for our chosen darlings. However, let us not forget that these technologies are just that - technologies that are relevant at a given time and in a given space, but not more. Yes, flamewars are fun but in retrospect, they are always silly. Hands up those who fought about Atari versus Amiga back in the good ol' 80s! Probably not many of you but, surely, you get the point. Initiatives such as FairPhone²⁵, ShiftPhone²⁶ or the GuardianProject²⁷ may prove more important than the OS or vendor of your choice in the future.

If you are lost in the vast space of mobile development, do not worry, stay calm and keep on reading. Go through the options and take the problem that you want to solve, your target audience and your know-how into account. Put a lot of effort into designing the experience of your service, concentrate on the problem at hand and keep it simple. It is better to do one thing well rather than doing 'everything' only so-so. Invest in the design and usability of your solution. Last but not least, finding the right niche is often better than trying to copy something that is already successful. This guide will help you make an informed decision!



²⁵ fairphone.com

²⁶ shiftphones.com

²⁷ guardianproject.info



From Idea to Prototype

What makes a service successful? Why do so many start-ups fail, often despite their technological superiority? Why do so many products seemingly get to a point where they are stuffed with features that have little use, yet eat up considerable development and maintenance time?

One of the biggest pitfalls in product development is failing to understand what your customers use your product for in their lives. Not knowing what progress your customers are trying to make can lead your team to build the wrong or over-engineer the correctly scoped product. This might seem paradoxical at first. It would be completely reasonable to expect that companies would not invest into features that will be used sparsely or not at all.

Unfortunately, start-ups and established companies alike lack a process to analyze the reasons why customers hire their product and decide what to build next based on nothing more than some sort of “best-practice-remixing” of the competitors’ successful features. Failing fast is misinterpreted as a goal in itself, forgetting that recovery from failure can be difficult and pivoting needs to be learned.

It is in every team’s interest to do their best at their first shot and so it is helpful to explore methods that allow you to uncover the most about your customers’ contexts, motivations and struggles with as little investment as possible before you start building.

In this chapter we will explore:

- Why people hire new products
- How to conduct pragmatic customer research to frame your

product's value proposition and discover what to build and what not to build.

- How to formulate insights and align your team around the progress that users are trying to make.
- How to prototype and test your ideas by involving your customers at every step

At the end of the chapter you will be equipped with a powerful perspective and a set of tools that will help you on your next endeavor to start building a service that customers will value.

Why do people buy products?

People do not buy products, they buy better versions of themselves.

In order to explain what this means, consider this example:

Remember the last time you purchased wine to serve it to your friends. How much did you know and care about the wine's properties and features such as vintage, grape, price etc.? If you are not a wine expert, I am sure the context of the upcoming meeting with your friends played a more important role in your decision than the individual features of the wine. The price and grape might have come up in your mind but the region and vintage probably did not influence your decision much. You might have simply selected a dry wine because you know that is what adults drink. Depending on the strength of your desire to impress your friends, you might have selected wine in a wine store to make sure you are serving something respectable to your friends. Or you might have visited the supermarket to simply get more of it—you know, to get the conversation rolling.

No matter what the case, you selected the wine based on your desire to be a better host in that given situation. To speak in a software analogy, you wanted to update the “version” of the host you were before. The attributes of the wine only played a role if you were able to connect them to this purpose in your mind. The wine helped you to “progress”, you hired it for the job of making you a better host.

Accordingly, when you interview a customer about a product, they will rarely talk about the product itself but what they do with it. They hire the product to make progress in their lives, not because of its features or attributes. And this urge to make progress arises when they encounter a situation where their current state is not good enough. We describe this condition and desire for progress as a Job to be Done¹.

Progress is not only functional.

It is important to note that when describing customer progress and the jobs they are hiring products for, it is not sufficient to think only in functional terms (What I can do with this). The wine from our previous example certainly needed to meet a certain standard to help you stand out as a good host, yet being a better host describes a mostly emotional and social struggle. You tried to impress your friends and feel appreciated. Because most human decision-making and motivation is ultimately emotional, framing how your product helps your customers to improve always includes an emotional dimension (How I want to feel and how others should think of me).

¹ To learn more about the Jobs To Be Done innovation perspective, read this free ebook by Alan Klement: www.whencoffeeandkalecompete.com

Customers do not see competition in the same way many organizations do.

When customers experience that something is not good enough and start looking for something better, they consider a variety of solutions that might deliver the desired progress. The term solutions may refer to products and services, but it also includes behaviors, workarounds, a combination of the latter—sometimes even—doing nothing. Following on our wine example, a party host who is interested in entertaining his guests might consider hiring wine, telling friends to bring their own favorite drinks or not hiring drinks at all and hiring a DJ instead.

Notice how the customers consider solutions outside of a particular product category (e.g. alcoholic beverages) and how this perspective differs from a traditional way to look at competition. When challenged with innovation, companies often focus on their closest competitors (beer, soft drinks, other wine producers) and end up remixing the competitors' best features into one-size-fits-all solutions. Failing to acknowledge that this approach often leads to over-engineering good enough solutions, companies become blind to disruptive efforts from other markets.

By embracing the view that customers are hiring solutions to progress and get a job done and not because of their features and functionality, you enable your team to innovate more freely and learn to categorize the market in the way your customers do. As a wine producer, you might incorporate your primary product (wine) into a service ecosystem that helps your customers to become knowledgeable and fun party hosts.

Why do people buy a certain product?

We have established that it is desirable to concentrate on helping your customers to become better versions of themselves when building new products. But how do you find out how this better version looks like? How do you find out which job your customers are hiring your product for?

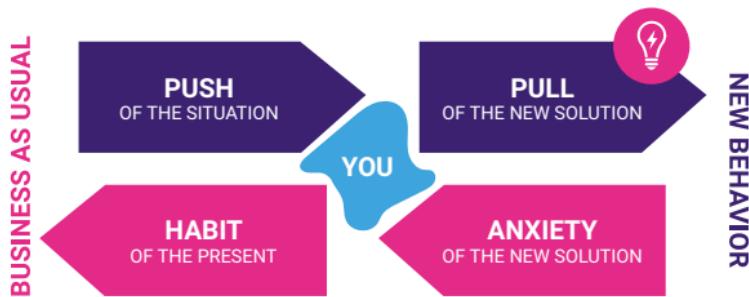
The most pragmatic way to uncover what motivates people to progress is to research their shopping and decision-making behavior. As customers switch from one product to another, they shape what is important to them. Their emotional, physical and social circumstances influence what tradeoffs they are willing to make, what features they consider important, what solutions they consider and what that better life with a new solution might look like. Understanding your customers' switching journey provides an excellent foundation for your product strategy:

Think about your last larger purchase, for example a mattress². I am sure that you took some time thinking about which model or type you should buy and that various situations, people and emotions influenced your journey to a new mattress. You might have seen the same mattress at your friend's house or have one day woken up with a terrible back pain that did not go away for the whole day.

² <http://jobstobedone.org/radio/the-mattress-interview-part-one/>

With every new purchase or behavior change, people go through a complex process of decision-making that is not only influenced by functional requirements but is also surrounded by plenty of emotional energy.

We can describe this energy with the help of a simple tool called the “Forces of Demand Diagram” that you can see below.



There are two forces that move the customer from the old solution to hiring a new product, the **PUSH of the current situation** and the **PULL of the new solution**. These forces are accompanied by two progress-blocking forces at the bottom of the diagram, the **ANXIETY of the new solution** and the **HABIT of the present**.

These forces can be explained in the following way:

Push: The push describes the need to make my current situation better. The leading thought here is: “What I have is not good enough.” It describes all the struggles and situations which stir people up to look for change.

In the mattress journey, this might be the terrible back pain caused by the old mattress or the purchase of a new, wider bed frame that you need to fill.

Pull: The other driver of progress, the pull, is created by

the new solution that your customer imagines. This new idea, as symbolized by the light bulb in the diagram, is a fantasy created in the mind of your customer about how much better his or her life might look after the purchase of the new product or service. It is created by your product's marketing, the customer's social environment etc.

The customer might need to see and touch the mattress before buying it or quite the opposite, she might just buy it based on a recommendation without ever touching or lying on the mattress at all.

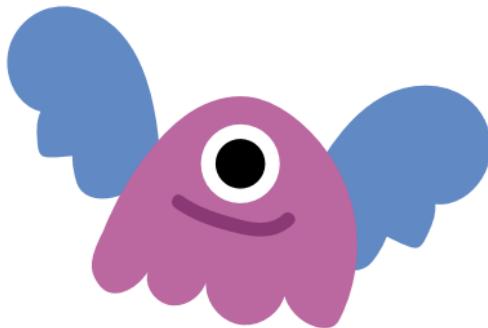
The pull is led by the thought: "Can this help me to make progress and get the job done better?"

Anxiety: Every consideration and fantasy of a new way (pull) is accompanied by progress-blocking anxieties. The purchase might be too expensive, the handling too difficult, the barriers of purchase too high.

You might have purchased a mattress a few months back, which will probably block you from investing a large sum again.

We might think: "Is this too expensive?" or "Will my partner support it?"

Habit: The "What I have right now, is good enough"



thought is probably the strongest hurdle that your product or service must overcome. If people are familiar with their solution or if the necessity to change is not large enough, people will not switch.

Understanding the anxieties experienced by your customers is a very important aspect of user research. Lowering them in consequence remains one of the most effective ways to increase demand.

Similarly, you could invest in letting your customers know about the negative aspects of their current situation and thus increase the push forces (insurances do this all the time).

Whatever your approach, once the moment is reached when the progressive forces are stronger than the progress-blocking forces, the customers will switch. The trade-offs are made in the mind of your customer, the positives overweight the negatives and money is put down on the table.

Switch interviews

Talking to customers who have switched to your product is an effective way to uncover why your customers hired your product. In this chapter, we will look at how to conduct these interviews. Mastering this technique will help your team to frame the design problem for your service and spot valuable innovation opportunities.

To understand how trade-offs were made and what forces were in play you need to understand what events happened in the purchase journey of your customer that led him to his decision. Recording the story that people went through on their way to your solution will almost universally reveal

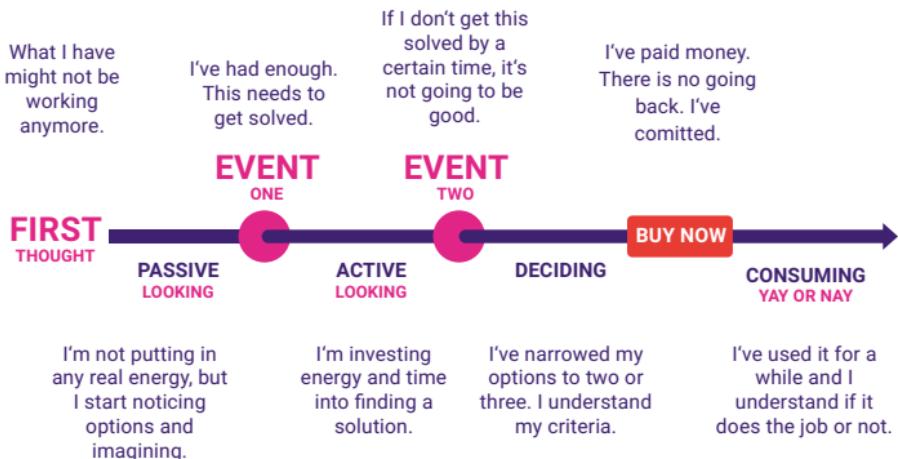
unexpected information, especially because no decisions are entirely rational³.

Switch interviewing is all about getting the story of your customers' progress on record, understanding all the causes and moments that led to purchase.

You start your interview by asking questions about the moment the customer purchased your product (The Switch). During the interview you retrace the story from the switch to the first thought, when your customer thought that what she has was not good enough anymore.

On the way you will often discover how complicated and unexpected some buying decisions were, broadening your understanding of the progress your customers are trying to make and their behavior.

To help you with this retrospective interviewing process, we will use a second tool: The Timeline.



³ <http://5by5.tv/criticalpath/146>

The First Thought: Once your customer wakes up with the realization: “What I have is not good enough. I might need to make progress.” the decision-making process begins. During the interview, it is one of your goals to try to find out, when, and most importantly, under what circumstances this thought occurred. Sometimes, negative experiences might have triggered these thoughts, sometimes your customer might experience your product at a friend’s house and be pulled to it. Either the case, from this moment on your customer starts to be sensitive to new solutions.

Passive Looking: In this phase your customers are increasingly sensitive to possible options that might help them to get their jobs done better. They do not make trade-offs yet, nor do they narrow down their selection strongly. They do not put any real energy into the looking process yet, they remain in fantasy land. Considering the mattress again, you might think: “If I ran across a great, new mattress for little money, and they would deliver it to my doorstep, I’d get it right away.”

Event One: Usually at some point, something happens that makes you start looking more actively: “This needs to get solved. I cannot continue like this.”

In my personal mattress journey, a friend of mine has got a slipped disk. This definitely made me to look for new alternatives to my own, not so good mattress.

Active Looking: Once your customer is investing some real energy into looking for solutions, for example by browsing websites, visiting stores or asking friends for recommendation or information, the trade-offs are formed and the most important decision criteria crystallize. This is when your customers’ value proposition is formed.

Event Two: More often than not, as you dig deeper in your interview, you might discover a pretty discrete and decisive moment, when the customer suddenly feels pushed to action

again. "If I do not get this solved by a certain time, it is not going to be good". It might be a conflict with a partner about the back-pain-fueled-grumpiness in the mattress scenario or a sudden drop in prices (Black Friday scenario).

Deciding: When the criteria are clear and trade-offs of possible alternatives are understood, your customer usually narrows down their options to two or three options. As mentioned earlier, finding out about this so-called consideration set provides great insights about what your product is actually competing with.

Buying: "I've decided and paid money. I will have to live with my decision." The buying process itself might be very short or pretty long, depending on the nature of your service or product. However, at this point most anxieties are overcome or weakened and your customer gives your solution a real chance. It is important to note that this kind of commitment is usually not based on significant experience with the product. Therefore, the customers' understanding of whether the new will replace the old and whether your solution fulfills the job is only partial. True value is not created yet.

Consuming: In the weeks following the purchase, the customers are no longer in love with their newly acquired solution, have gathered some significant experience with it and can more clearly judge whether the job is fulfilled in a more satisfactory way or not. "I've used it and understand if it is better or not. "It is now that true value is created in the mind of your customer.

Satisfaction: During consumption the customer comes to realize if he or she is actually satisfied with the new product. If not, this frustration might act as a forward-pushing force in the consequent search for a better solution in near the future.

Who to interview

It is important that you talk to people who have already bought and used your product who have a somewhat fresh memory of all the important tradeoffs and circumstances that shaped their decision. Because people are prone to rationalize their (ultimately emotional) decisions, avoid talking to customers who are only thinking to buy but did not commit yet. Similarly, talking to people who got a product gifted will not reveal any valuable insights—they simply did not spend any energy thinking about it. You will use the story of the shopping and decision-making process as a tool to identify customers whose motivations have undergone the test of reality, getting a clearer view on their actions as opposed to their aspirations and beliefs.

As a guide: Interview customers who switched to your product in the last 90–120 days. They will not be in love with the brand-new product anymore and will have had time to evaluate if they were able to reach the desired progress or not.

What if my product does not have customers yet?

It does not matter whether your product is already on the market or not, or if it is a product, service, app or a feature. While interviewing current customers is easier for beginners, what you are after is the story of change. Once customers leave their old way of doing things in order to progress (for example starting to pay for premium features), they can be interviewed. In case your product is not released yet, you can simply interview people who switched to a competitor's product. In case you cannot think of any competitor or your app is free, talk to customers who exhibit a new behavior that is in line with the progress you are trying enable.

To give you an example: To create a new service that helps people to become better hosts, you could interview people who

hired wine, DJs, a cooking class or use a new recipe app and see what motivated them. Be creative and sensitive to what people do today to improve in a similar way or situation.

For further tips, please read this article on Medium:
jtbd.info/uncovering-the-jobs-that-customers-hire-products-and-services-to-do-834269006f50.

Define the problem: Job Statement

After you have conducted 6–12 switch interviews, you should be able to spot patterns emerging in the forces that shaped the customers' desire for progress. For further development it is helpful to summarize these insights in a structured and concise manner. This process will improve the communication with your team and will help you to see which parts of your product are essential and which you can abandon.

The job statement helps your team to establish a high level vision for your project. It expresses what your customer is struggling with and what job they desire to get done. It opens up space for ideation by capturing insights into the customers current state and motivation to progress. And it helps to shape your product's value proposition and aligning your company's strategy around a customer-centric purpose.

A job statement has two parts: 1. The situation in which a desire for progress arises and the future, 2. Desired state once the job is done and the progress has been made.

For a grocery shopping application, the job statement might be: "Free me of the stress of looking for healthy ingredients (STRUGGLE), so I can rekindle the fun of cooking for my family (STRUGGLE IS RESOLVED)."

A job statement should not include any solutions or features, otherwise it will limit your team's creative potential.

Consider this job statement that ignores this rule: “Deliver groceries to my home, so I spare time shopping.”

The latter statement describes the functionality of a grocery shopping app well, but it does not provide room for broader thinking. Solutions, such as ready-made inspirational boxes sold at the supermarket or recipe collections would not be captured by it. Observe further how the goal of the second statement is creatively limiting (“so I spare time shopping”). “Sparing time” does not describe a desired future state, a “new me”. It is an effect that might alleviate a problem people experience that blocks them from having fun while cooking (shopping takes time and is stressful), but it does not describe how the life the customer would improve.

For further tips how to formulate your job statements, consult the free book “When Coffee and Kale Compete” by Alan Klement⁴.

Define Situations: Job stories

After having formulated a high level vision of your product using the job statement, you can now use Job Stories to describe concrete struggling moments in your customers' lives. Job stories provide your team with a way to capture concrete situations in which the customer starts to look for something new. They capture all the different struggling moments, contexts, anxieties, desires and emotions that you heard throughout your interviews. You can use job stories to frame smaller design problems to kick off the ideation process for an individual product within the scope of a service or a particular feature.

Job stories explain how a particular group of customers acted in a particular situation, what they were doing, what

⁴ whencoffeeandkalecompete.com

effects a solution should have in their mind and in which direction they wanted to progress. Let us have a look at an example job story for a grocery shopping application:

When I feel bad about shopping ingredients for my children that are not organic, because visiting multiple stores with my small children is very exhausting,

I want a way to avoid having to shop at multiple locations, **so that I can** serve my family healthy meals without trading convenience for quality.

When... describes the situation in which a problem arises and the person is looking for change. What happened?

I want to... captures the expectations that customers have regarding the effects of a solution when they use it. The solution itself or its features are not mentioned. What effects does or does not the solution have in their mind?

So I can... describes what happens when the struggle is solved. It describes how life will be when the solution is effective in getting the job done.

If you are familiar with user stories, you will recognize that job stories are fairly similar in their structure. The important difference is that job stories attempt to tell a story of progress instead of describing the needs of a particular persona towards reaching a goal. The reason for this is the observation that very diverse people can act similarly when they find themselves in the same situation. Consider this example:

A wealthy business man has about 25 minutes to pass security at the airport. He is hungry and sees that the line at the checkpoint is pretty short. He figures that he can grab a Snickers quickly to recharge before entering the nasty procedure. He enters a kiosk, grabs the chocolate bar and pays. He knows that a Snickers is not the best choice and goes against the advice of his nutritionist, but he enjoys it anyway.

Now try to answer this question: Which elements of the

story had stronger influence on his decision? That he was wealthy, a businessman and is concerned his nutrition or that he found himself in an airport with limited offerings and time and an appetite for a quick bite? People quickly override their stated preferences and attitudes when they are confronted with a situation or social context that makes alternative behavior more favorable. This is the main reason, why asking people what they would do or what they want is such a risky activity. People adapt. It is not who they are, it is where, when and with whom they are that causes their behavior.

However, many software development teams work with "personas", fictional characters representing certain segments of their target group.

But as we discussed before, it is not who the customers are in terms of their demographic and psychographic group, what they state they want, how they look, where they live or what brands they prefer which causes them to switch. While these data can correlate with certain behavior, they do not cause it. Context does. It is therefore to frame your customer insights around situations and the behaviors and tradeoffs they cause, not around personas.

That is why job stories offer an improved way to frame insights into customer motivation and can be used as a foundation in agile development processes. While going in-depth on how to turn these insights into a complete product roadmap would extend beyond the boundaries of this book, let us have a look at a few helpful tools that help your team to frame what you have learned and start building.

- If your team is used to working with personas, reduce the amount of correlative data and focus on customer motivation⁵.
- Intercom offers free ebooks⁶ on topics such as product management, jobs-to-be-done and starting-up. They also provide a format called Intermission⁷ that can help you kickoff a project. It is a project brief that tells a story of a customer wanting to progress, a couple job stories and ways to measure your team's progress towards solving the customer's problem. To see how this works in real-life, read their blog⁸.
- Eric Ries and his Lean Startup methodology⁹ provide you with a great toolkit and perspective in order to avoid building features that customers will not use and ensure that you do not miss opportunities for improvement as you build. Ries recommends a scientific approach of formulating customer-centric hypotheses, building solutions and prototypes as soon as possible, testing them quickly and analyzing how customers react with the help of clear metrics.



⁵ www.slideshare.net/AndrejBalaz/improving-personas-with-jobs-to-be-done

⁶ intercom.com/books

⁷ blog.intercom.com/accidentally-invented-job-stories

⁸ blog.intercom.com/how-we-build-software/

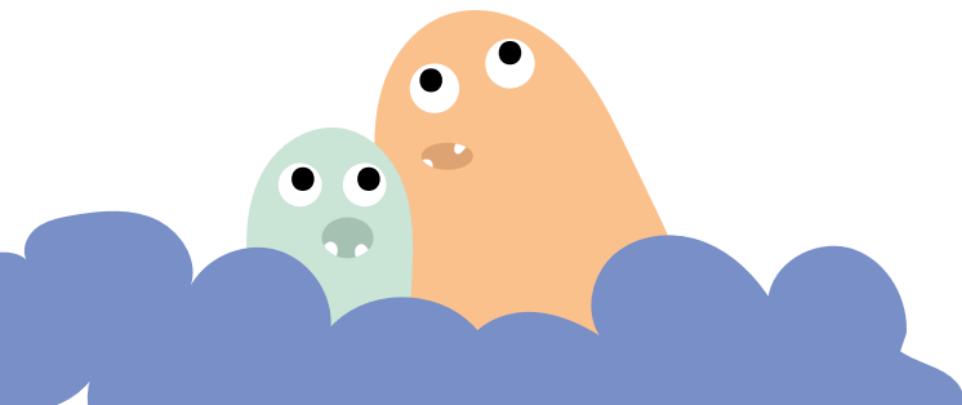
⁹ theleanstartup.com/principles

Define the Use Case: Customer Journey

Before you dive into building complicated prototypes or drawing wireframes digitally, it is essential that you sit down with a couple of sheets of folded paper or a block of long, screen-sized post-its and attempt to sketch how your customer will use your app.

Create a storyboard that captures the user's physical surroundings, who they are with and what they are interacting with. If your service has more than one touchpoint, for example if it enables customer to share a ride and both the riders and drivers have an app, you need to think about the interactions between these two people. Your goal is to get the story and different interactions right. In this way you will quickly notice and improve inconsistencies and get a feeling of which portions of the customer's journey are most challenging.

Pay special attention to what happens outside of the screen's boundaries. In some circumstances a customer cannot hold their phone in a hand, for example when moving around at airports with lots of baggage in tow. This will enable you to open up your imagination to other forms of interaction. Think about a bus service. It combines physical, as well as digital touchpoints that together help reduce the customers' anxieties and guide them through the experience.



Create the User Interface: Prototypes and Visual Design

When you understand how individual journeys fit together in the bigger picture you can move into more defined design.

An interactive prototype is the best way to visualize and evaluate your app's interactions. It is usable enough to communicate the design, so you do not need to provide as much documentation as you would need to annotate static images. Your prototype can have visual design applied and look exactly as it will after the implementation, or you can stay on the wireframe level and focus on functionality and content.

It does not matter whether you have a big budget or are working on a personal project over the weekend, having a fairly complete prototype of your app is the best way to communicate your concept and discuss it with others. The non-linear narration of your apps should be self-explanatory at this stage. Many prototyping tools allow you to experience your concept on an actual device. Take the advantage of it.

Prototypes are usually developed before you spend time on implementing code and pixel perfect design. An agreed clickable walkthrough is a useful reference that teams can work towards without risking going too much off track. It is also great to user test prototypes and get external feedback on.

In terms of putting a prototype together there is no single best solution. You can use whatever technique works for you. From paper prototyping to using one of the specialized tools or other applications that have the functionality to put clickable journeys together. If you have coding skills, building an HTML prototype is another good way to go. You can also rapidly prototype on the existing app, it all depends on what approach works for a specific project setup. In that sense everything can be seen as a prototype until it is released.

Google Ventures Design Sprint

The design sprint methodology by Google¹⁰ helps you to develop a tested prototype for your service within 5 days. It does require user research before you start, but if you followed our suggestions so far, you should be equipped to start it right away. In the beginning of the week you decide on scope and the experience you want to design, then you progress to ideation, sketching and building of a testable prototype. By the end of the week you will have tested your idea with real users, gathering invaluable feedback about your idea.

The Google Design Sprint will get you a good overview of various helpful methods that you can apply to prototyping and sketching. In the following section let us highlight a few methods that will help you to transform your idea into a more coherent concept.

Prototyping & Wire Framing tools

While paper should remain a truthful companion to your design process all along the way, teams around the world have spawned great prototyping tools that help you test your ideas directly on the device and share it with others. They allow you to further elaborate your ideas, test concepts quickly and work on your devices. Below you will find a few tools that work well together.

Quick prototyping with Sketch and InVision+Craft or Marvel

Sketch is a design tool for Mac that has become the de-facto standard for designing user interfaces. Its community offers hundreds of useful plug-ins and the Sketch keeps improving the tool at a rapid pace.

Sketch is easy to learn and integrates well with tools such

¹⁰ www.gv.com/sprint

as InVision or Marvel. The latter allow you to build lightweight clickable prototypes that can be previewed directly on your device, speed up your design process (especially thanks to the many helpful real-data import features from Craft by InVision).

The prototypes are great for communication with your client (if you have any) and early usability testing. Personally, I recommend to make your designs interactive early on to spot problems with the interaction flow. Viewing your designs on your device is the only way to take the right visual design decisions regarding type sizes, whitespace and many other usability considerations.

- **Sketch:** sketchapp.com
- **InVision:** invisionapp.com
- **Marvel:** marvelapp.com

High-fidelity prototyping with Framer, Principle

If you are looking into prototyping animated flows and transitions that feel like the real thing, Framer will be your first choice. While having a steeper learning curve, the CoffeeScript-like programming language is as approachable as it can be. If you prefer to leave code to other team members, Principle provides you with a simple interface to make interactive prototypes reality. It has its limitations when your prototype requires many states, so be prepared to “fake” things a lot. Both tools are well-documented and integrate well with Sketch.

- **Framer:** framer.com
- **Principle:** principleformac.com

Keeping specs and assets in order with Zeplin and InVision

Craft

When transferring your designs to development, exporting of specifications and UI assets have been a painstakingly time-consuming process. Zeplin, as well as InVision help in this regard, allowing you to view information about margins, color values, code snippets at one glance interactively. While InVision Craft eliminates the need to spec designs, Zeplin goes further by simplifying exporting of assets. Both tools allow you to organize assets in helpful libraries that greatly improve cross-team collaboration.

- **Zeplin:** zeplin.io
- **InVision:** invisionapp.com/craft

Windows: Affinity Designer, Adobe Experience Design

If you are on a Windows computer, your options are a bit more limited, but you are not left alone. Affinity Designer is an excellent program for graphic design which allows you to create user interfaces as well. It is fast, cheap and supports reusable components. If you are familiar with Adobe Illustrator and Photoshop, you should feel right at home.

If you have the Creative Cloud subscription, you might give Adobe XD a try. While still relatively young, it is faster than any other Adobe application in creating and exporting clickable UI prototypes while sporting an easy-to-use interface. It has made great strides in catching up to Sketch, but it lacks its community and plugin ecosystem.

- **Affinity:** affinity.serif.com
- **Adobe Experience Design:** www.adobe.com/products/experience-design.html

Visual Design

Unless you are building an app that uses a non-visual interface, your app's UI will rely on graphics. You probably clarified conceptual aspects of user interface design in the sketching and wireframing phase. Depending on the level of detail you applied to your prototypes, you might also already have thought about visual design details and implemented them. If you have not: Do it now. As good as your idea might be: You will probably not succeed if your app's visual appearance is not attractive to users. Sometimes it is actually only the visual appearance that makes an app successful.

But a well-polished visual design will not only improve your UI's aesthetic appeal, a well-executed branding enhance your app's functionality and reduces the learning curve for users by providing visual cues.

Style consistency through the flow helps users make sense of your UI and learn interactions faster. For example, if your main action button changes color from screen to screen, consider the impact on the users. Will they be confused? Will they understand the reason behind the change? If the style alterations are intentional make sure you are doing them for usability reasons.

Similar to designing layouts and interactions on the prototyping level, certain styling decisions might be informed by specific platform guidelines. Your app can look very different depending on which platform it was defined for. Make sure that your design follows the recommended practices for font use, standard icons and layout conventions. Again, see the platform-related chapters of this guide to find more information and links to specific resources.

Company branding in the UI can be applied in a non-obstructive way so that users can concentrate on interacting with your app. Use the background, controls color and maybe

certain images or layout choices to achieve the brand's look and feel.

Finally, the launch icon is the first impression visual element that your app will be identified by and judged on. Make it look good and easy to recognize.

User Testing

The best way to validate your interface concept is to show it to users as soon as your work is representative enough to prompt feedback. You do not have to wait until you have a finished and polished product. Testing early can save you a lot of time in the long term. It will expose concepts that do not work early in the process. The more time you invest into developing your designs, the harder it gets to let go of them and start over. It is more difficult to accept feedback on something that you considered almost done that on a clickable prototype that you can update quickly.

Test your assumptions regarding interaction, visual design and content as often as you can. It will help you to streamline your design process, uncover problems quickly and generate new ideas.

The best way to test your designs is to invite some users over and watch them perform your testing tasks. You will quickly see where they struggle and will be able to invite stakeholders to sit in or watch a live recording of the test to help your team members empathize with your customers. Typical user testing session is about an hour long. During that time users that are unfamiliar with the product are asked to perform certain tasks, usually around core functionality.

Here are some helpful tools that will greatly improve your process:

1. Build a testable prototype
2. Set up your hardware for recording. Use Reflector 2¹¹ and Quicktime or Lookback¹² to record your screens and cameras. Try to record the device and the user's face for later analysis
3. Print a short introduction for your participant. Imagine a description similar in length and style to those in the App Store. (Value proposition, short)
4. Print every task on a separate sheet for your participants to read
5. Recruit participants and invite them. When searching for people to interview it is good to refer to the original personas descriptions and look for users that match those profiles. Offering rewards makes it easier to find appropriate candidates, but be careful when choosing them: You do not want people to show up because they get money, you need people that fit your target group and give you honest feedback. So consider offering rewards that fit your target group, e.g. a photo printing voucher if you are testing a photo app.
6. Introduce yourself, clarify that there are no right or wrong answers and that the person should think aloud. Let them know that you want to hear everything they think when they do something with your prototype. Mention that the prototype is not complete and there might be some unfinished parts. If they assume that the person that is running the session is the author of the design, they might feel cautious of giving critical feedback. Reassure them that they are free to express their honest opinions. After

¹¹ www.airsquirrels.com/reflector/

¹² lookback.io

- all, the only reason you arranged the testing session was to get an independent feedback.
- 7. Let your participant read the introduction and then perform all tasks
 - 8. Do not lead users to any conclusions. Do not help them out by revealing how things work (unless they cannot figure it out and you cannot proceed with the session) and word your questions in a non-interruptive way.
 - 9. Give a reward to your participant
 - 10. Discuss key insights directly after the test with your other team members. What went well, what went bad, what was surprising? As you progress through the tests with 4–8 participants, patterns will start to emerge.

For a more robust approach, use Lookback which allows your users to participate in your test by simply downloading an app to their device. Not only can you record what your users are doing, you can choose to perform in-house tests, let users perform the test on their own (unmoderated testing) or jump into a live session with them (moderated testing). The Lookback tools are thorough and easy to use and can spare you lot of time and fiddling with your own setup.

If you want to spare time with recruiting and recording your participants, you can use platforms such as *userTesting.com* to facilitate your testing process. You will receive recordings of your users as they are confronted with your prototypes and tasks. The biggest advantage of *userTesting.com* is their large pool of potential users, allowing you to recruit people from different countries and demographics. While the recruiting criteria do not allow you to select people based on their situations, struggles or switching behavior, being able to let your tests run over the weekend is a great time-saver. The downside is that your prototypes need to be more refined than in an in-person test where you can moderate if something breaks.

When you get feedback, you can reiterate your design and improve the parts that were not quite complete or if the feedback was good move on to the development phase.

If you are still exploring new areas and your own prototype is not quite ready, you can run testing sessions on other apps that have been already released. It can surprise you how much others notice about the application that you might never have thought of.

Keep Iterating and Learning

As you build your product, it is important that you test your assumptions frequently. Not only will you be able to uncover what works and what does not before committing large sums of money and time to building the wrong features, you will build empathy for your customers within your team.

Users, their motivations, struggles and circumstances change continuously, so it is crucial to develop processes within your company to gather feedback at various levels of the creation process.

Successful innovation does not have to be guesswork, but it will always remain a study of complexity. The sooner you develop your own process of learning and understanding, the sooner will you be able to qualify your customers' feedback, take better scoping decisions and find customers that will value your efforts. On to the challenge!



Android

The Ecosystem

The Android platform is developed by the Open Handset Alliance led by Google and has been publicly available since 2007. Its use by the majority of hardware manufacturers has made it the fastest growing smartphone operating system ever which today dominates the market: More than 86% of all smartphones sold in Q3 2016 worldwide were based on Android¹, 80% of all professional mobile developers are targeting Android². The number of Android apps on Google Play has surpassed 3 million in July 2017³. Over 2 billion Android devices have been activated so far⁴ which also includes wearables, tablets, media players, set-top boxes, TVs, phones and car entertainment systems.

Android is an operating system, a collection of pre-installed applications and an application framework supported by a comprehensive set of tools. The platform continues to evolve rapidly, with the regular addition of new features every 6 months or so. The latest release is Android 8.0 Oreo, which was just released. Android Oreo introduced a lot of interesting new features (see below), some of which are introduced to tackle one of the most discussed issues when developing for Android: The system's fragmentation.

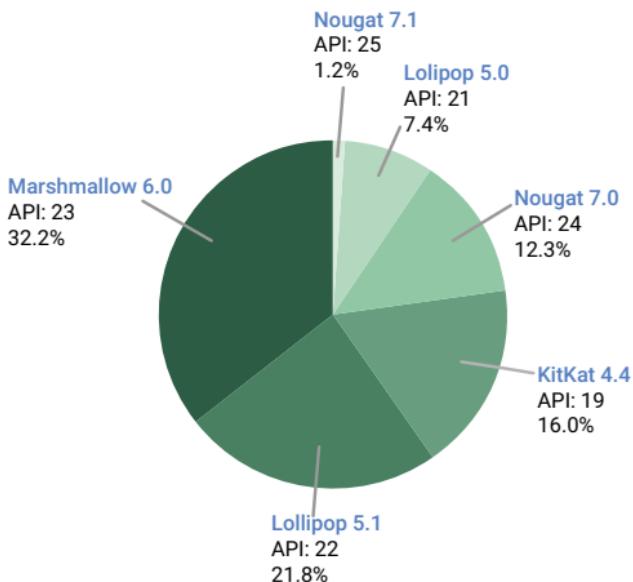
¹ idc.com/prodserv/smartphone-os-market-share.jsp

² sometimes among others, see the Developer Economics Report 2016, p7 available via developereconomics.com

³ www.appbrain.com/stats/number-of-android-apps

⁴ www.macrumors.com/2017/05/17/2-billion-active-android-devices

The multitude of different devices by various manufacturers and the fast progress of the platform itself leads to uncertainty over whether or not your Android application will run everywhere. In addition, the adaption of the latest OS version is slower compared to other mobile platforms. However, today, you will reach over 90% of the installation base if you decide to target Android 4.4 or above⁵.



Data collected during a 7-day period ending on August 8, 2017.

Any versions with less than 0.1% distribution are not shown and devices without Google Apps (like those from Amazon and many China-based manufacturers) are not counted. All versions prior Android 4.4 were also excluded due to their outdated nature.

⁵ developer.android.com/about/dashboards

Android Nougat

Android 7 "Nougat" was officially released in August 2016, with the Nexus 6, 5X, 6P, 9, Nexus Player and Pixel C as flagship devices.

Nougat introduced a split-screen mode for phones, as well as a hidden, experimental multi-window mode. The notification system was redesigned, the shade now being able to bundle multiple notifications from the same app.

Nougat extends device battery life with an enhanced "Doze" power saving mechanism. It also introduced "seamless" updates on newer devices, switching between two partitions and applying updates in the background (while the device is still in use).

Google also added platform support for Vulkan, a high-performance, low-level 3D-rendering API to supplement OpenGL.

In version 7.1, Nougat added support for Google's Daydream virtual reality (VR) platform, for use with its Google Daydream View virtual reality headset.

Android Oreo

Android 8 "Oreo" is the most recent OS release. After four developer previews, it got released and initially delivered to Google's Pixel phones and the latest Nexus devices. This version allows notifications to get snoozed and grouped into channels. Also, more customization regarding sounds and alarms are possible. There is a new picture-in-picture mode which can e.g. be used for watching videos while using the phone. The ability to utilize Neighborhood Aware Networking (NAN) for e.g. data sharing, should increase the connectivity possibilities within Android.

The updated Android Runtime (ART) provides an improved performance and a more fine grain tuning of background activities of apps should provide a better battery life.

A new hardware abstraction layer called the “vendor interface” separates low-level, hardware-specific code from the Android OS framework – an important architectural change. Due to this change vendors should be able to reuse most of their code changes if a new version of Android gets released. This is hopefully a huge step in the direction of faster and more reliable updates.

Developers will be able to use a multi process WebView, which will be guarded by Google Safe Browsing. This should secure apps that use web content within their flows.

To reduce the workload for users, developers are able to use the new Autofill Framework⁶ which will also support passwords and credit card information in a secure manner.

Android Go

Android Go is basically a lightweight version of Android O, designed to run on smartphones that have 1 GB or less of RAM. Even apps on the Play Store will be optimized for these lower-end devices mostly targeting emerging markets like India or China.

Kotlin: It is Official

At the Google I/O 2017 keynote, the Android team announced first-class support for the programming language Kotlin⁷. Android Studio 3.0 now ships with Kotlin out of the box. Named after the Russian island, Kotlin was initially developed by JetBrains in Saint Petersburg in 2011. It is a concise, safe and welcome modern alternative to Java. Compare it to Swift, for example:

⁶ developer.android.com/preview/features/autofill.html

⁷ kotlinlang.org

Kotlin

```
var myVariable: Int = 41
val myConstant = 42 // type inference
fun greeting(name: String?, age: Int): String {
    val name = name?.capitalize() ?: "Stranger" // safe navigation and null coalescing operators
    return "Hello $name. You're $age!" // string interpolation
}
```

Swift

```
var myVariable: Int = 41
let myConstant = 42 // type inference
func greeting(name: String?, age: Int) -> String {
    let name = name?.capitalized ?? "Stranger" // safe navigation and null coalescing operators
    return "Hello \(name). You're \(age)!" // string interpolation
}
```

Material Design

During the Google I/O conference in June 2014, Google unveiled their new design language based on paper and ink, named Material Design. Originally codenamed Quantum Paper, Material Design extends the "card" concepts first seen on Google Now. Says designer Matías Duarte: "unlike real paper, our digital material can expand and reform intelligently. The material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch."

Perhaps for the first time, Material Design brought a strong, consistent visual identity to the Android ecosystem, parallel but distinct from iOS's flat design and Windows' Metro design. To encourage a solid user experience and consistent appearance of Android apps, Google provides a comprehensive documenta-

tion for the design language⁸ and a design guide for developers⁹. Going into the importance of color schemes, design patterns, and the new Material design, the guide provides a great orientation when building apps for the Android ecosystem.

Also in the latest version of Android is Material Design utilized to build the UI and the system apps. Additionally, a variety of libraries brought support for material design to different other platforms, like the web. Third party developers started to adapt the design concept within their own solutions.

Android Wear

Android Wear¹⁰, launched in 2014, is basically the Android OS ported to smartwatches and other wearable devices, which can pair with phones running Android version 4.3 or newer (there is some limited support for pairing with an iPhone). Wear devices integrate Google Now and the Google Play Store.

A key feature is the Google Fit ecosystem of apps that support run and ride tracking, heart activity, step-counting, etc. Users can use their watch to control their phone – music, for example. Notifications via the vibration engine are another key element. Those can be used for notifications from Google Now like flight reminders, traffic warnings, meeting reminders, etc.

Android TV

Android TV¹¹ is a successor to Google's previous smart TV initiative, Google TV. Android TV is designed to be built into TVs as well as standalone digital media players. Several TV manufacturers, including Sony, Sharp, and Philips, have integrated

⁸ google.com/design/spec/material-design

⁹ developer.android.com/design

¹⁰ developer.android.com/training/building-wearables.html

¹¹ android.com/tv

Android TV into their screens today. Users can download apps and games from the integrated Play Store, including media apps like YouTube, Hulu, and Netflix. Some devices also include direct support for the Chrome Cast Receiver. Users can mirror their screen or stream content directly from their phone or the web to the device.

Android TV apps use the same structure as those for phones and tablets. Developers can thus leverage their existing apps and knowledge to target the TV platform. See developer.android.com/tv to learn how.

Getting Started

The main programming language for Android is based on Java. But beware, only a subset of the Java libraries and packages are supported and there are many platform specific APIs that will not work with Android. You can find answers to your "What and Why" questions online in Android's Dev Guide¹² and your "How" questions in the reference documentation¹³. Furthermore, Google introduced a section in their documentation called "Android Training"¹⁴ that helps new developers learn about various best practices. This is where you can learn about basics such as navigation and inter-app communication, as well as more advanced features such as intelligent Bitmap downloads and optimizing your app for better battery life. Experienced developers are able to acquire the Associate Android Developer Certification by Google¹⁵ since February 2017.

¹² developer.android.com/guide

¹³ developer.android.com/reference

¹⁴ developer.android.com/training/index.html

¹⁵ developers.google.com/training/certification/

To get started, you need the Android SDK¹⁶, which is available for Windows, Mac OS X, and Linux. It contains the tools needed to build, test, debug and analyze apps. Development is done within an adapted version of the IntelliJ Idea¹⁷ IDE. This Tool is called Android Studio¹⁸ and allows beside developing, also automatic building, syntax checking and testing.

IDE support

Android Studio is the official IDE for Android and comes directly with Gradle Support and many features explicitly tailored to Android development. It is available as pre packed download including the Android SDK. An extended feature list¹⁹ as well as an end user documentation²⁰ can be found on the official Android Studio website. Android Studio itself comes with example code and provides code documentation for all system classes and methods available.

Native development

The Android NDK²¹ enables native components to be written for your apps by leveraging both JNI for invocations of native methods and using native subclasses that offer callbacks to its non-native pendants. This is important for game developers and anyone who needs to rely on efficient processing.

¹⁶ developer.android.com/sdk

¹⁷ jetbrains.com/idea

¹⁸ developer.android.com/studio/index.html

¹⁹ developer.android.com/studio/index.html#features

²⁰ developer.android.com/studio/intro/index.html

²¹ developer.android.com/tools/sdk/ndk

Implementation

App Architecture

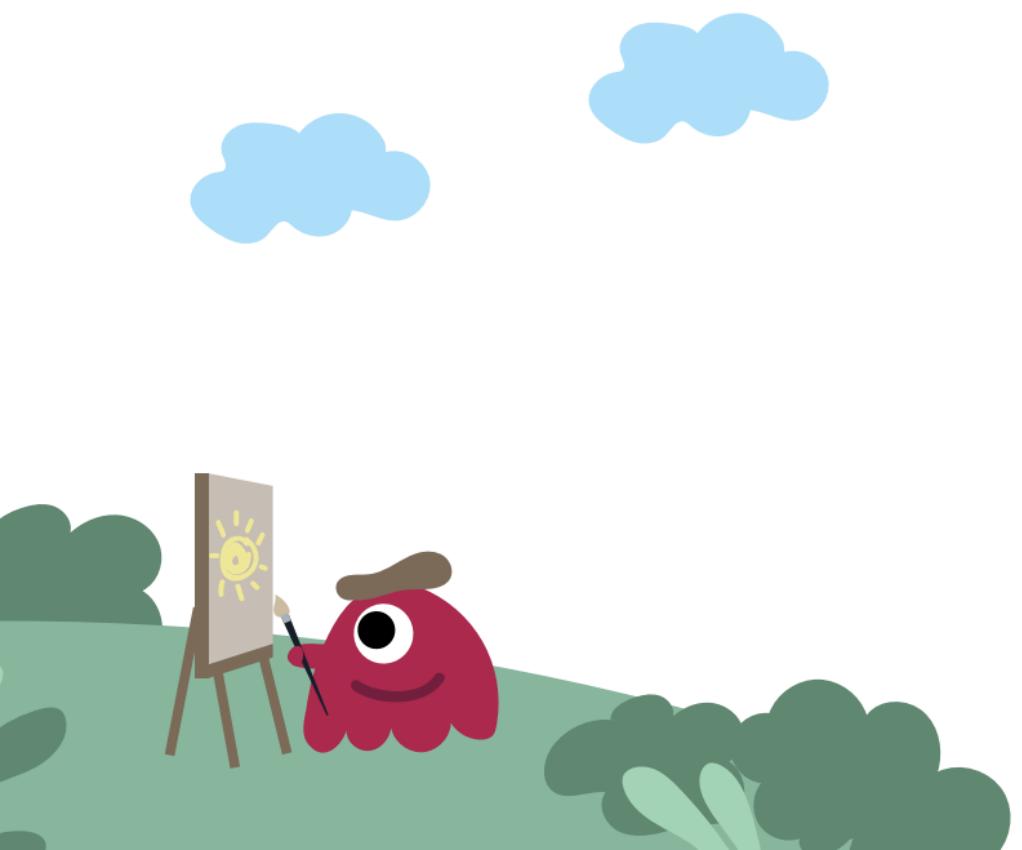
Android apps usually include a mix of Activities, Services, BroadcastReceivers and ContentProviders; these all need to be declared in the application's manifest. The manifest also includes the metadata of an application, like the title, version and its required permissions.

An Activity is a piece of functionality with an attached user interface. A Service is used for tasks that run in the background and, therefore, are not tied directly to a visual representation. A BroadcastReceiver handles messages broadcast by the system, your own or other apps. A ContentProvider is an interface to the content of an application that abstracts from the underlying storage mechanisms, e.g. SQLite.

An application may consist of several of these components, for instance, an Activity for the UI and a Service for long-running tasks. Communication between the components is achieved by Intents or remote procedure calls handled by Android Interface Definition Language (AIDL).

Intents bundle data, such as the user's location or a URL, with an action. These intents trigger behaviors in the platform and can be used as a messaging system in your app. For instance, the Intent of showing a web page will open the browser. A powerful aspect of this building block philosophy is that any functionality can be replaced by another application, as the Android system always uses the preferred application for a specific Intent. For example, the Intent of sharing a web page triggered by a news reader app can open an email client or a text messaging app depending on the apps installed and the user's preference: Any app that declares the sharing Intent as their interface may be used.

The user interface of an app is separated from the code in Android-specific XML layout files. Different layouts can be created for different screen sizes, country locales and device features without touching the Java code. To this end, localized strings and images are organized in separate resource folders. Of course, you are also able to define and design layouts in code or make use of both strategies to enable dynamic UI updates.



The SDK and Plug-Ins

To aid development, you have many tools at your disposal in the SDK, the most important ones are:

- **android:** To create a project or manage virtual devices and versions of the SDK.
- **adb:** To query devices, connect and interact with them (and virtual devices) by moving files, installing apps and alike.
- **emulator:** To emulate the defined features of a virtual device. It takes a while to start, so do it once and not for every build.
- **ddms:** To look inside your device or emulator, watch log messages, and control emulator features such as network latency and GPS position. It can also be used to view memory consumption and kill processes. If this tool is running, you can also connect the Eclipse debugger to a process running in the emulator. Beyond that, ddms is the only way (without root-access) to create screenshots in Android versions below 4.0.

These four tools along with many others, including tools to analyze method trace logs, inspect layouts and test apps with random events, can be found in the tools directory of the SDK. If you are facing issues, such as exceptions being thrown, be sure to check the ddms log or use the logcat mechanism.

If you are using features such as Fragments²² for large screens, be sure to add the corresponding Support Library packages from Google. They are available through the SDK + AVD Manager or since latest changes also via the Google

²² developer.android.com/guide/topics/fundamentals/fragments.html

Maven repository²³. They enable developers to deploy modern features on older Android Versions without compatibility issues. Be sure to use the v4 packages in your apps to provide maximum backward support. There is also a version for Android 2.1 and above called v7 appcompat library that introduces a way to implement the ActionBar pattern and more as documented online²⁴.

Developing your application against Android 3.1+ will enable you to make homescreen widgets resizable, and connect via USB to other devices, such as digital cameras, gamepads and many others. Android 4.X releases introduced further interesting features such as expandable notifications, lockscreen widgets, and a camera with face detection. The Material Design UI Toolkit was introduced with Android 5.0 and introduces new widgets and more to use in phones, wearables and other platforms. The native computing framework, Renderscript no longer provides direct graphic rendering capabilities but may now be used for heavy processing instead.

There are also support libraries dedicated to specific areas of Android. The Design Support Library provides navigation enhancements and the floating action button for older versions and the Vector Drawable Library + Animated Vector Drawable Library add support for different vector formats.

To provide some backward compatibility for devices with older Android versions, Google began to use the Google Play Services framework²⁵ which gets updated via the Play Store and adds libraries such as the latest Google Maps. If you are interested in authenticating users, you might want to have a look at the Google+ Sign capabilities that bring the benefit of

²³ developer.android.com/topic/libraries/support-library/setup.html

²⁴ developer.android.com/tools/support-library/features.html

²⁵ developer.android.com/google/play-services/

real user data to your app. The functionality is managed via OAuth 2.0 tokens that allow the use of the Google account on the user's behalf.

Testing

The first step in testing an app is to run it on the emulator or a device. You can then debug it, if necessary, through the ddms tool.

All versions of the Android OS are built to run on devices without modification, however, some hardware manufacturers may have changed pieces of the platform. Therefore, testing on a mix of devices is essential. To get an idea of which devices are most popular, refer to [AppBrain's list²⁶](#).

To automate testing, the Android SDK comes with some capable and useful testing instrumentation²⁷ tools. Tests can be written using the standard JUnit format, using the Android mock objects that are contained in the SDK.

The Instrumentation classes can monitor the UI and send system events such as key presses. Your tests can then check the status of your app after these events have occurred.

MonkeyRunner²⁸ is a powerful and extensible test automation tool for testing the entire app. These tests can be run on both virtual and physical devices.

In revision 21 of the SDK, Google finally introduced a more efficient UI automation testing framework²⁹ which allows functional UI testing on Android Jelly Bean and above. The

²⁶ www.appbrain.com/stats/top-android-phones

²⁷ developer.android.com/guide/topics/testing/testing_android.html

²⁸ developer.android.com/guide/developing/tools/monkeyrunner_concepts.html

²⁹ android-developers.blogspot.de/2012/11/android-sdk-tools-revision-21.html

tool itself can be executed from your shell with the command `uiautomatorviewer` and will present you the captured interface including some information about the views presented. Executing the tests is relatively easy: After you have written your test, it is then built via ANT as a JAR-file. This file has to be pushed onto your device and then executed via the command `adb shell uiautomator runtest`.

Since July 2017, the Android Testing Support Library is available in version 1.0. This release provides new features like multiprocess Espresso and support for the Android Test Orchestrator. Multiple stability and performance improvements are also worth noting.

Espresso³⁰ provides a very lean API that helps to quickly write procedural tests for your UI.

Open source testing frameworks, such as Robotium³¹, can complement your other automated tests. Robotium can even be used to test binary apk files if the app's source is not available. Robolectric³² is another great tool which runs the tests directly in your IDE in your standard/desktop JVM.

Your automated tests can be run on continuous integration servers such as Jenkins or Hudson. Robolectric runs in a standard JVM and does not need an Android run-time environment. Most other automated testing frameworks, including Robotium, are based on Android's Instrumentation framework and will need to run in the respective JVM. Plugins such as the Android Emulator Plugin³³ enable these tests to be configured and run in Hudson and Jenkins.

³⁰ developer.android.com/training/testing/espresso/

³¹ code.google.com/p/robotium

³² robolectric.org/

³³ wiki.hudson-ci.org/display/HUDSON/Android+Emulator+Plugin

Building

Aside from building your app directly in the IDE of your choice, there are also more flexible ways to build Android apps. Gradle³⁴ is now the officially supported build automation tool for Android. There is also a Maven plugin³⁵ which is well supported by the community. Both tools can use dependencies from different Maven repositories, for example, the Maven Central Repository³⁶.

Google ships libraries for Gradle as Android Archive (.aar) files that can be obtained using the Android SDK Manager. You are also able to package your own libraries or SDKs utilizing the android-library plugin for Gradle. A great source for finding Gradle-friendly Android libraries is "Gradle, please"³⁷.

Signing

Your apps are always signed by the build process, either with a debug or release signature. You can use a self-signing mechanism, which avoids signing fees (and security).

The same signature must be used for updates to your app - so make sure to not lose the keystore file or the password. Remember: you can use the same key for all your apps or create a new one for every app.

Google also provides a centralized solution for signing. Google Play App Signing³⁸ hands over a lot of the work to

³⁴ tools.android.com/tech-docs/new-build-system

³⁵ code.google.com/p/maven-android-plugin/

³⁶ www.maven.org

³⁷ gradleplease.appspot.com

³⁸ developer.android.com/studio/publish/app-signing.html

Google's infrastructure and could reduce problems with managing and securing the needed keys.

Distribution

After you have created the next killer application and tested it, you should upload it to Android's appstore called "Play" at play.google.com/apps/publish.

You are required to register with the service using your Google Checkout Account and pay a \$25 registration fee. Once your registration is approved, you can upload your app, add screenshots and descriptions, then publish it.

Make sure that you have defined a `versionName`, `versionCode`, an icon and a label in your `AndroidManifest.xml`. Furthermore, the declared features in the manifest (uses-feature nodes) are used to filter apps for different devices.

One of the recent additions to the Google Play Store is alpha and beta testing plus staged rollouts. This allows you to do some friendly user testing before publishing the app to all users. Furthermore, you can target specific countries and devices by setting the right flags in the Developer Console and export detailed statistics that help in understanding your userbase. Using the inbuilt localization service, you can easily add new languages to your app by paying a fee - make sure to check the Localization Checklist³⁹ for detailed information about the importance of this topic.

As there are lots of competing applications in Android Play, you might want to use alternative application stores⁴⁰. They provide different payment methods and may target

³⁹ developer.android.com/distribute/googleplay/publish/localizing.html

⁴⁰ onepf.org/appstores

specific consumer groups. One of those markets is the Amazon Appstore which comes pre-installed on the Kindle Fire tablet family. But you should keep in mind that alternative play stores force the user to enable unknown sources for app installation, which is always a potential security risk.

Adaptation

As the adaptation of Android increases, the vendor specific ecosystem also been growing. That involves their own SDKs, fully-customized Android versions and tools around topics such as alpha and beta testing. This has both upsides, such as a very tight integration that allows an amazing experience for users, and downsides, such as increased fragmentation of ecosystem. Vendor specific marketplaces often prohibit the upload of generic apps that utilize utilities other than their own.

One example is Amazon's Kindle Fire ecosystem which is basically a customized fork of Android and represents the Android tablet with the biggest market share: Instead of using Google's Play Services for enabling in-app purchases or maps, you have to use Amazon's own libraries that offer similar functionality. The reasoning behind it is pretty simple: Kindle devices are not delivered with the required libraries to run Google's services. Amazon also offers their own advertisement and gaming services (comparable to Google Play Games) that help to target your audience. Offering emulators for their different device models, Amazon helps perfect your app by providing a realistic environment. On top of the testing that Amazon offers for their developer community, they also review any app that gets uploaded to their Appstore.

Here is a little overview that can help you find the right resources

Vendor	Documentation
Amazon	developer.amazon.com/fire-tablets
HTC	htcdev.com
LG	mobile.developer.lge.com
Samsung	developer.samsung.com
Sony	developer.sonymobile.com

Interestingly enough more and more vendors (e.g. Samsung and HTC) have also started to offer vanilla Android versions of their devices called "Google Play Edition". These devices use the same hardware as the regular models but do not come with any software customization. These devices are directly distributed through Google's Play Store and offer bleeding edge devices to users that want to stick to Google's experience.

In August 2017, Lenovo went a step further and announced that they will discontinue their Android customization called "Vibe Pure" and ship their devices with stock Android⁴¹.

Additional to the versions of the major manufacturers, the Android Open Source Project (AOSP)⁴² offers an open source version of the Android stack to create custom ROMs and port devices to the Android Platform. Independent manufacturers like Fairphone⁴³ use AOSP to create their version of the Android platform. The downside of this approach is missing Google services like the Google Play Store as normally available on mainstream Android devices.

⁴¹ gadgets.ndtv.com/mobiles/news/lenovo-k8-note-vibe-pure-ui-stock-android-1733110

⁴² source.android.com

⁴³ fairphone.com

Monetization

Google Play is the main distribution channel and of course the most popular platform for Android app distribution. Google charges you 25USD for the registration and a transaction fee of 30% of your earnings.

But the Play Store is not your only option- see the monetization chapter in this guide to learn more about the app store landscape and its opportunities.

For the vendor specific ecosystems, such as Samsung Apps or Amazon's Appstore, you should consider using their SDKs to enjoy the benefits of optimized monetization.

In addition to selling an app in one of the many app stores available, there are several different ways of monetizing an Android app. One suitable way is by using advertising, which may either be click- or view-based and can provide a steady income. Other than that, there are different In-App Billing possibilities such as Google's own service⁴⁴ that utilizes the Google Play Store or PayPal's Mobile SDK⁴⁵ and Mobile Payments Library⁴⁶. Most services differ in transaction-based fees and the possibilities they offer for example subscriptions, parallel payments or pre-approved payments. If you are looking to bring extra cool functionality to your app, you should consider implementing card.io's SDK⁴⁷ for camera-enabled credit card scanning.

Be sure to check that the payment method of your choice is in harmony with the terms and conditions of the different markets you want to publish your app to. Those particularly for digital downloads, for which different rules exist, are worth checking out.

⁴⁴ developer.android.com/google/play/billing/

⁴⁵ github.com/paypal/PayPal-Android-SDK

⁴⁶ developer.paypal.com/webapps/developer/docs/classic/mobile/gs_MPL/



iOS

Ten years after the launch of the first iPhone, the side project that rode on the success of the iPod when it first launched has developed into one of the biggest successes in the history of the tech industry with more than a billion units sold. When the iPad was launched a few years later, in 2010, iOS became a multi-device operating system and a major force in mobile computing.

The iPhone upset the smartphone market when it was released by introducing some unusual paradigms which exist to this date, such as a multitouch screen without a physical keyboard and direct interaction using gestures.

One of the driving forces behind the success of the iPhone was the early adoption of apps by third-party developers. With the release of iPhone OS 2.0 in July 2008, Apple opened up the App Store with 500 apps - and the rest is history.

The Ecosystem Today

The popularity of developing for iOS is not losing any of the momentum it established over the course of the lifetime of the App Store. Total developer earnings from Apple's AppStore have exceeded \$70 billion by June 2017.¹ Interest in Apple's yearly Worldwide Developer Conference (WWDC) is at an all-time high and a lottery is in place to decide who is assigned a ticket.

¹ www.apple.com/newsroom/2017/06/developer-earnings-from-the-app-store-top-70-billion/

iOS Install Base

In addition to selling over one billion iOS devices, a plus in Apple's favor is the high adoption rate of each iOS version soon after release. This allows developers to focus on the latest version as a development target and not worry about supporting a lot of devices on older versions, which has been a challenge for Android developers. In fact, less than two months after the launch of iOS 10 Mixpanel already reported an adoption rate of around 72% of all iOS devices² with 19% still on iOS 9, leaving only 9% of devices running an older iOS version. Contrast this with Android's OS version 7.0/7.1 Nougat, which after ten months has not managed to capture even 10% of the installed base³. Android developers have to target a much lower API level to capture a significant audience, which translates into a much faster adoption of critical new features on iOS.

Devices Running iOS

One point that plays into the high adoption rate is that iOS generally supports devices ranging back a few years - iOS 11, for instance, still supports the iPhone 5s originally released in 2013. Usually, this means that developers need to have a few devices available to test on different screen sizes and hardware generations.

A detailed list of iOS devices, their capabilities and supported iOS versions can be found on Wikipedia⁴.

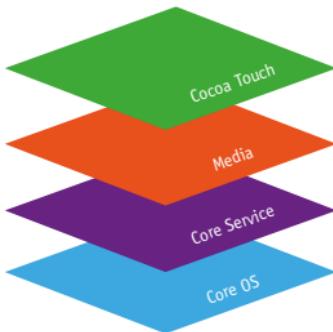
² mixpanel.com/trends/#report/ios_10

³ developer.android.com/about/dashboards

⁴ en.wikipedia.org/wiki/List_of_iOS_devices

The Architecture

Like most operating systems the iOS Architecture is defined by layers of technologies to allow your application to run on a device without communicating directly at the hardware level. These technologies can be thought of layers or interfaces that are packaged as frameworks that the developer imports into their iOS projects to leverage. The primary framework developers interact with, is called Cocoa Touch.



Cocoa Touch

While macOS and iOS are different operating systems, they share a lot in common in terms of frameworks, developer tools, and design patterns.

Apple leveraged and extended the main framework for developing macOS apps, Cocoa, and added support for unique features in iOS such as Touch gestures and named it Cocoa Touch. Included in Cocoa Touch are frameworks to build GUI, access device sensors like the accelerometer and perform networking and data management tasks.



Getting Started with iOS Development

Along with the SDK to develop for iOS, Apple also provides an Integrated Development Environment (IDE) called Xcode to create both iOS and OSX applications. As Xcode has evolved, Apple has strived to provide all the needed tools to write, test, monitor performance and deploy apps to the App Store all from inside Xcode.

Development tools and paradigms change extremely quickly on iOS, which is partially owed to the internal pace at which Apple introduces new technologies, such as Swift and partially owed to community interest. Several iOS adopters have released comprehensive tools and frameworks targeted at developers. For instance, as of 2017, one of the most pervasive trends is to use tools like React Native which use JavaScript and other web technologies to build an abstraction over the native UI code.

Xcode

Apple released Xcode in 2003 for writing applications in OS X. Version 3 of Xcode supported the first iPhone SDK in 2008 and the most recent version is Xcode 9, just recently released with iOS 11. Xcode is an integrated development environment used during the whole application development life-cycle. Interface Builder is a visual design tool used to design and wire together views of the application without writing code and is integrated with Xcode. Also provided is an iOS Simulator to allow developers to test their apps on all current devices without having to always install apps on physical devices.

Interface Builder

A lot of discussion in iOS developer circles is whether it is better to use Interface Builder to visually design the UI and application flow or to undertake it all manually with code. In the past this may have been a personal preference but with new devices and screen sizes like the Apple Watch and iPhone X, the case can be made that Interface Builder is becoming more essential. One of the primary differences between iOS and Android development was not having to develop for several device types and screen sizes. However, this line is becoming more blurry with iOS 10 and 11 supporting eight different screen sizes. Instead of supporting all of them separately in your applications, Interface Builder uses concepts such as Auto-Layout and Adaptive Layout to aid the developer in supporting all screen sizes more easily. With each new version of Xcode, Interface Builder has seen improvement and advancement so it is apparent that Apple prefers developers to take advantage of it. Something a new iOS developer should consider.

Objective C

Objective C has its roots in the NeXTSTEP operating system developed in the 1980s from where OSX and iOS are derived. It is an object-oriented programming language that adds messaging to the C Programming language⁵. In fact C and C++ can be written alongside Objective C and some of the iOS frameworks only provide a C level API to access. However, it has been criticized for having a quirky syntax with a plethora of asterisks, '@' signs, and square brackets which leads to a higher learning curve for developers coming from modern languages such as Java or C# while providing improved legibility through named

⁵ <http://en.wikipedia.org/wiki/Objective-C>

parameters and verbose class and method names. Incremental improvements have been added over the years including dot notation of object properties, blocks, collection literals and memory management via Automatic Reference Counting (ARC). But the remaining need to use pointers, header files and remain tightly coupled to the limitations and risks of the C language has left Apple to conclude a new modern language is needed.

Swift

In July 2010 Chris Lattner, Senior Director and Architect in the Developer Tools Department at Apple began implementing the basic language structure of a new programming language whose existence only a few people knew of. It became a major focus for the Apple Developer Tools group in July 2013 and almost a year later at Apple's World Wide Developer Conference (WWDC) Apple announced a new programming language for iOS and OSX called Swift. Lattner stated Swift is influenced by other languages such as C#, Ruby, Haskell, Python and countless others⁶.

Apple felt the reason to create Swift was the need for modern language syntax that is more concise and easier to learn for new iOS developers, including modern features like inferred data types, data structure declarations, tuples, closures, optional semicolons and no pointers. It has been suggested Apple's support for Swift is to ensure iOS developers stay interested in Apple's tools and do not look at other platforms with modern language support for iOS development.

In early December 2015, Apple open-sourced Swift⁷ along with a bunch of related tools, frameworks and examples. Apple

⁶ nondot.org/sabre

⁷ github.com/apple/swift

is actively engaging the community in the future development of the language by soliciting feedback, proposals for new language features and pull requests. Less than a week after it was first open-sourced, Swift is already the #1 open source programming language on GitHub⁸, overtaking other popular languages like Ruby or PHP. In March 2017, Swift cracked into the top 10 of the TIOBE index⁹.

Together with Xcode 9.0, Apple introduced Swift version 4.0. While the pace of development has settled a bit and the direction of the language becoming more clear, different language versions still neither have source nor binary compatibility.

Objective-C vs Swift

The question whether to start new projects using Objective-C or Swift can be extremely tricky to answer, since there are valid reasons to use either of the two languages at the current time.

In favor of Objective-C, there is the argument that the language is extremely mature, all of Apple's APIs still feel like they were designed to work with the language in mind - or at the very least with C, which generally works really well in conjunction with Objective-C. As opposed to Swift, Objective-C offers excellent source and binary compatibility, so far that projects from years ago still compile without any changes, or only with minimal changes. On Swift, on the other hand, there is a lot of cost involved in migrating the code to new versions of the language. Debugging is also still much nicer in Objective-C with fewer bugs in the necessary tools. Many of Swift's features, such as generics, were ported back to Objective-C, so Apple is not really neglecting the language in any way.

⁸ github.com/showcases/programming-languages

⁹ www.cultofmac.com/471301/swift-is-already-of-the-worlds-most-popular-programming-languages/

On the other hand, Apple is clearly positioning Swift as the future of development on their platforms, including iOS. APIs are consistently updated to work better with the language and feel less alien - for instance in/out parameters, such as for error handling. Some areas that still feel very bolted on in Swift, for example NSCoder support for archiving, get cleaned up in Swift 4 and even the required time for migrating between Swift versions gets less with every passing release, partially owned to better tooling support in Xcode.

Performance Tools and Testing

In addition to providing the tools to develop iOS applications, Xcode also comes with tools for performance monitoring and testing.

Instruments

Instruments allows developers to collect data about the performance and behavior of their iOS apps over time. Some of the common templates offered allow developers to track memory leaks, or detect application "hot spots" using the profiler instrument. The Automation instrument is used to automate user interface tests in your iOS app through test scripts written by the developer. These scripts run outside of the app and simulate user interaction by calling the UI Automation API. It can be run on a device or simulator.

XC Test Framework

XCTest is the test framework integrated with Xcode to provide extensive testing in an organized and efficient way. By default, new projects created in Xcode using one of the application templates will add a Test target to the project. This allows the developer to write their own unit test classes, execute them and analyze the results using the Test Navigator, all from inside Xcode.

Setting Up the Dev Environment

After registering for a free developer account at *developer.apple.com* access is granted to download Xcode, sample code, videos, and documentation. Requirements to run all Xcode tools is a Mac computer running OS X 10.10 (Yosemite) along with the iOS SDK. This setup will allow for the creation and testing of iOS apps to run in the iOS Simulator. To submit apps to the App Store you must upgrade the developer account at a cost of \$99 a year which also gives access to betas of future versions of Xcode and iOS as they are released.

Distribution

The primary method for deploying apps to consumers is through the App Store. Each app submitted is reviewed by the Apple review team to ensure it meets the requirements and standards set by Apple. This is a major difference from the Google Play store for Android apps where Google does not review apps but ensures they are code signed.

Apple is very strict on how 3rd party applications run on iOS and uses the Sandbox technique to ensure application security and tries to prevent nefarious or buggy code that could compromise the OS, other applications or the device. Think of a sandbox as a virtual barrier around the application that sets the rules of what resources the app can access. For example an application does not have access to another app's file directory or system resources not accessed by the SDK frameworks. Apple has given more control to the user to grant access to their data (i.e. contacts, calendars, photos) or GPS location. Developers must prepare for cases where the user has denied these type of requests.

Learning Resources

With the popularity of Apple's developer ecosystem comes a multitude of learning resources in different formats to help a new developer start coding for iOS, and a lot of them are free. By taking advantage of these resources and others like them the learning curve of mastering iOS development will lessen considerably.

Websites and Blogs

- **Developer.apple.com** contains complete reference and programming guides for developers to learn how to develop iOS apps and class reference of all classes in their public frameworks. The library website is organized by Resource Types, Topics, and Frameworks plus the ability to search. One important document to read before designing the first app to be submitted to the app store is the iOS Human Interface Guidelines¹⁰. It offers developers recommendations on Apple approved ways to design apps to ensure a positive user experience. Violation of these recommendations will most likely lead to apps being rejected by the App Store during review for submission.
- **Swift.org**, the Swift community's official home
- **RayWenderlich.com** has become an essential site for free iOS tutorials written by his community of developers with the goal being "to take the coolest and most challenging topics and make them easy for everyone to learn - so we can all make amazing apps." The site has expanded into offering programming books and Video tutorials (with a

¹⁰ developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG

paid membership). Subscribe to their weekly Podcast for the latest news relevant to developers and interviews with leaders in the iOS developer community.

- **iOS.devtools.me** is a website created by Adam Swinden that he updates daily with the best iOS developer tools and back-end services to help in developing apps. Content is organized by categories (i.e Design, Graphics, Debugging), most popular, and recently added. Also provided is a weekly newsletter on the latest additions to the site.
- **iOSDevWeekly.com** is a weekly round up of the best iOS development links every week. Dave Verwer operates the site and they offer a weekly email newsletter published every Friday.
- **Galloway.me.uk**, a blog by London based iOS developer/author Matt Galloway. His book "Effective Objective-C 2.0"¹¹ is highly recommended when ready to start learning advanced features and tips on the language.
- **Merowing.info** is a blog from developer/trainer/speaker Krzysztof Zablocki who offers tutorials and insights into iOS development from his experience as a consultant. He also is active in the open source community creating tool and libraries for iOS developers.
- **AshFurrow.com** is another popular iOS blogger/developer who proudly states the purpose of this blog is for "Exploring the Pain Points of iOS." He has authored multiple iOS development books, is an active speaker and involved in the Open Source Community.
- **This Week in Swift** is a weekly newsletter involving the most interesting Swift-related news, developments, tutorials and general tidbits related to iOS development.

¹¹ available via <http://www.amazon.com/Effective-Objective-C-2-0-Specific-Development/dp/0321917014>

Online Video Training

As a member of Apple's Developer program you get free access to all of Apple's World Wide Developer Conference videos, source code, and presentation files are available to download and stream via the website or WWDC iOS app from the past several years. Apple usually makes the videos available the day after the presentation whereas previously it would take weeks to be available after each year's conference.

Lynda.com currently offers over 70 video courses with paid membership subscription for beginning iOS Development. Source code for the projects are available to download depending on the membership level chosen. They also have a free app in the App Store to watch videos on iOS devices.

One popular free resource for video training is offered by iTunes University of a full semester course taught at Stanford University on beginning iOS development. The lectures dive deep into the Objective-C language and iOS Frameworks. Viewers can even download the coding assignments. Videos are viewed through iTunes or the iTunesU app for iOS devices.

Finally, YouTube has quite a few free videos for Learning iOS development including a channel created by Mohammad Azam¹² that lists several screencast tutorials for iOS.

¹² www.youtube.com/user/azamsharp



Final Thoughts

It is an exciting time to be part of the iOS Development community and hopefully this chapter will prove helpful in finding a starting point. To say things change quickly is an understatement with all the new devices, frameworks and services that have been launched in the past few years. But do not get intimidated by the speed at which technology moves inside Apple's ecosystem. Most of the basics in developing a standard App apply now as they did in the first few versions of iOS. Luckily there are countless resources available to get started and grow your iOS development skills and most of them are free.

Things to consider when getting started on your first iOS "Hello World" App and beyond.

- Does it make more sense to use Interface Builder for designing the UI layout or to do it in code?
- While using back-end services like CloudKit make development easier am I locking myself too much into Apple's architecture which would make developing an Android version accessing same back end not possible?
- What are the drawbacks of developing iOS apps outside of Xcode (using cross-platform tools)? Is the user community big enough to find answers to issues? How well do their products keep up with the latest releases to iOS?
- What is the environment like today for being a full-time iOS Indie Developer?

Answers to these questions are beyond the scope of the chapter but the resources above can help you navigate around the pitfalls in iOS development quicker on the way to being an experienced iOS Developer. Good luck and welcome to the club.



Going Cross-Platform

With only Android and iOS as the main players, why should you consider using a cross-platform development framework? In this way even a small team can cater both platforms. And you might even target other form-factors and media easily like PCs, game consoles and websites.

Key Differences Between Platforms

If you want to deliver your app across different platforms you have to overcome some obstacles. Some challenges are easier to overcome than others:

- **Programming Language:** Java & Kotlin for Android, Objective-C and Swift for iOS
- **UI and UX:** Style and interaction patterns differ on each platform.
- **Desktop/Launcher Integration:** On iOS you can only add a badge with a number to your app's icon, while on Android you can add a full-blown desktop widget that may display arbitrary data and use any visuals.
- **Lockscreen Integration:** Again, there are various ways to integrate on the lockscreen.
- **Multitasking:** Android supports full background services, to preserve battery. iOS only has limited support for running apps in the background.
- **Fragmentation:** The Android ecosystem is very fragmented, iOS is a lot more homogenous.
- **Platform Services** such as push, in-app-purchase and in-app-advertisement differ on each platform.

Cross-Platform Strategies

This section outlines some of the strategies you can employ to implement your apps on different platforms.

Direct Support

You can support several platforms by having a specialized team for each and every target platform. While this can be resource intensive, it will most likely give you the best integration and user experience on each system. An easy entry route is to start with one platform and then progress to further platforms once your application proves itself in the real world.

Component libraries can help you to speed up native development, there are many commercial and open source components available for all platforms.

Asset Sharing

When you maintain several teams for different platforms you can still save a lot of effort when you share some application constructs:

- **Concept and assets:** Mostly you will do this automatically: share the ideas and concepts of the application, the UI flow, the input and output and the design and design assets of the app (but be aware of the need to support platform specific UI constructs).
- **Data structures and algorithms:** Go one step further by sharing data structures and algorithms among platforms.
- **Code sharing of the business model:** Using cross platform compilers you can also share the business model between the platforms. Alternatively you can use an interpreter or a virtual machine and one common language across a variety of platforms.

- **Complete abstraction:** Some cross platform tools enable you to completely abstract the business model, view and control of your application for different platforms.

Player And Virtual Machines

Player concepts typically provide a common set of APIs across different platforms. Famous examples include Xamarin¹ and Lua². This approach makes development very easy. You are dependent, however, on the platform provider for new features and the challenge here is when those features are available on one platform only. Sometimes player concepts use a “least common denominator” approach to the offered features, to maintain commonality among implementations for various platforms.

Cross Language Compilation

Cross language compilation enables coding in one language that is then transformed into a different, platform specific language. In terms of performance this is often the best cross platform solution, however there might be performance differences when compared to native apps. This can be the case, for example, when certain programming constructs cannot be translated from the source to the target language optimally.

¹ xamarin.com

² lua.org

There are three common approaches to cross language compilation: direct source to source translation, indirectly by translating the source code into an intermediate abstract language and direct compilation into a platform's binary format. The indirect approach typically produces less readable code. This is a potential issue when you would like to continue the development on the target platform and use the translated source code as a starting point.

(Hybrid) Web Apps

Hybrid web development means to embed a webview within a native app. The standard for hybrid apps is the open source tool Apache Cordova³ (formerly known as PhoneGap). This approach allows you to access native functionality from within the web parts of your apps and you can also use native code for performance or user experience critical aspects of your app. Hybrid apps allow you to reuse the web development parts across your chosen platforms. Read the web chapter to learn more about mobile web development.

ANSI C

While HTML and web programming starts from a very high abstraction you can choose the opposite route using ANSI C. You can run ANSI C code on all important platforms like Android, iOS and Windows. The main problem with this approach is that you cannot access platform specific APIs or even UI controls from within ANSI C. Using C is mostly relevant for complex algorithms such as audio encoders. The corresponding libraries can then be used in each app project for a platform.

³ cordova.apache.org

Finding the Right Cross-Platform Framework

For a benchmark of the available frameworks refer to the research2guidance report available at research2guidance.com/cross-platform-tool-benchmarking-2014.

Popular and interesting frameworks include the before mentioned Cordova⁴ and Xamarin⁵ but also Corona⁶, Cocos2D⁷, Flutter⁸, Unity⁹, NativeScript¹⁰, Sencha¹¹ and Titanium¹².

Here are some questions that you should ask when evaluating cross platform tools. Not all of them might be relevant to you, so weight the options appropriately. First have a detailed look at your application idea, the content, your target audience and target platforms. You should also take the competition on the various platforms, your marketing budget and the know-how of your development team into account.

- How does your cross platform tool chain work? What programming language and what API can I use?
- Can I access platform specific functionality? If so, how?
- Can I use native UI components? If so, how?

⁴ cordova.apache.org

⁵ xamarin.com

⁶ coronalabs.com

⁷ cocos2d.org

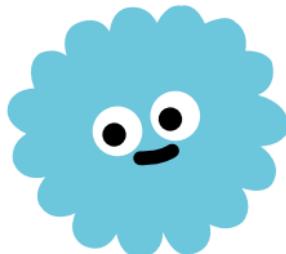
⁸ flutter.io

⁹ unity3d.com

¹⁰ nativescript.org

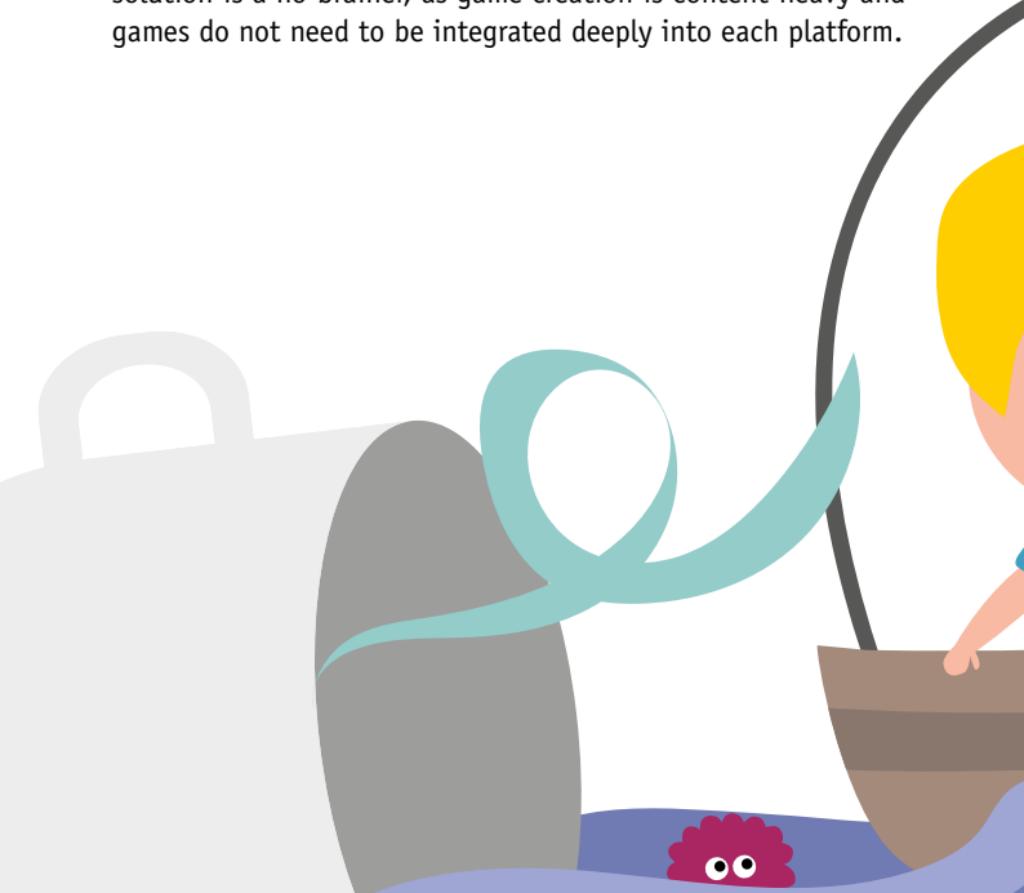
¹¹ sencha.com

¹² appcelerator.com



- Can I use a platform specific build as the basis for my own ongoing development? What does the translated/generated source code look like?
- Is there desktop integration available?
- Can I control multitasking? Are there background services?
- How does the solution work with push services?
- How can I use in app purchasing and in-app advertisement?
- How does the framework keep up with new OS releases?
- What's the performance of the solution?

Last but not least, for gaming using a cross-platform solution is a no-brainer, as game creation is content-heavy and games do not need to be integrated deeply into each platform.







Mobile Web

While the theme of this book is largely app-oriented, it would not be complete without talking about the mobile web. Indeed the line between apps and web is often blurred in an ecosystem where apps can be built entirely with web technologies, can pull their data and content in via web API requests, or can act as simple app shells for what is essentially a browser (WebView). It can be useful to think of a web-native continuum, with native at one end and web at the other, and various hybrid models in between.

The mobile web and native apps are often pitted against each other as competitors. In many ways they are; often either approach would be suitable to solve a particular problem. It is easy, however, to get lost in the arguments; there are emphatic and obsessive proponents on both sides.

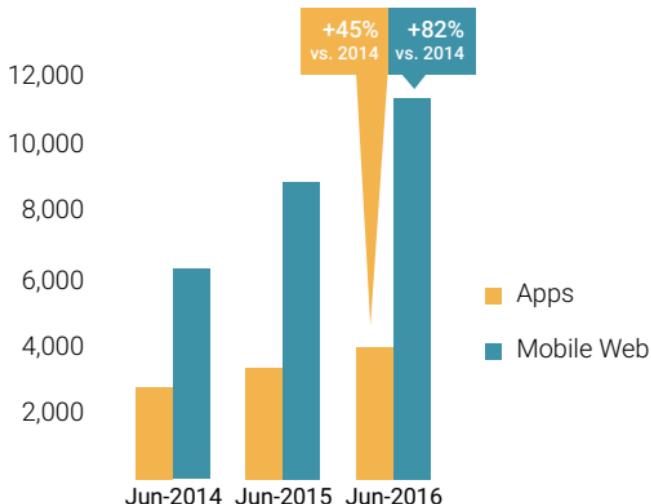
But while apps and web are competing platforms, it is also true that they are complementary platforms, each with its own set of strengths and weaknesses. We will not dwell on the app versus web argument here. Rather we will view them as complementary technologies, each with its own set of strengths and weaknesses, which often overlap.

That said, with modern web features such as Device APIs, push notifications, installable apps, 60 fps animations, discoverability, the mobile web is a platform both capable and formidable.

Mobile Web Usage

The world has already reached the tipping point where more time is spent on mobile than desktop. And while users spend far more time in apps than on the mobile web, it can be misleading to think that is the whole story. The mobile web has a far larger audience than native apps.

Average Monthly Audience: Top 1000 Mobile Apps vs. Top 1000 Mobile Web Properties



(Data from the U.S., users aged 18+, source: comscore.com/Insights/Presentations-and-Whitepapers/2016/The-2016-US-Mobile-App-Report)

Most digital strategies will touch the mobile web in at least some way, if not embrace it wholesale.

Devices, Browsers and Fragmentation

The web today is mostly experienced through a browser running on a desktop computer or mobile device. This is where things can get tricky for mobile developers: if you come from a desktop web development background and thought that developing and testing for all the various desktop browsers was an arduous task, then you would better sit down; things are considerably more complex on mobile.

There are at least as many mobile browsers as there are on desktop. But on mobile, in addition to the browser(s) on the device, we also have to consider the combination of devices and their properties and capabilities. The types of properties and capabilities that can impact web development include

- **Screen size properties** such as physical dimensions, aspect ratio, and pixel density
- **Input types** such as keypads, touchscreens, styli, microphones, and cameras
- **Spatial sensors** such as GPS, accelerometers, compasses, and gyroscopes
- **Network capabilities** such as WiFi, 3G, LTE

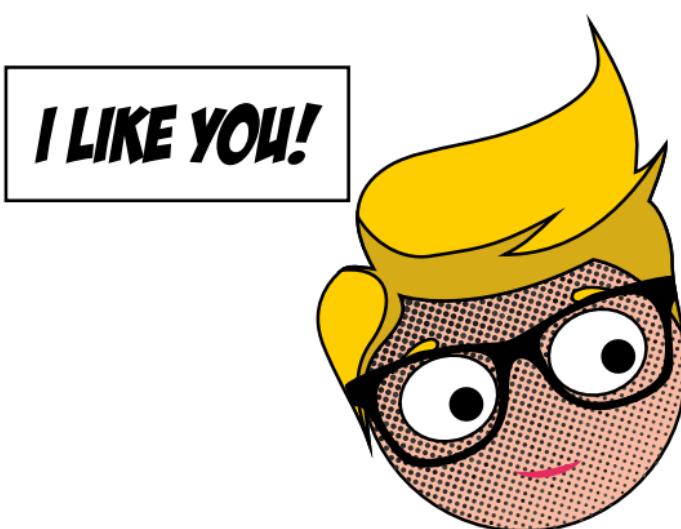
This results in a staggering number of device-capability and browser combinations that your visitors might be using. If you want to provide a good user experience then ideally you want your web pages to work on all device-browser combinations. This is the scale of fragmentation problem facing web developers today.

What is a Web Browser Anyway?

The web browser is a central part of the web platform. It is a complex piece of software with many roles. It orchestrates the underlying web technologies, combining them into functional web pages. It acts as a window and interface to the web for the user, interpreting the users actions and inputs, and rendering its response in real time.

On top of all this the major browsers come bundled with a set of complex developer tools, that provide deep insights into the inner workings, structure and performance on any web page that it renders. There are many developer tool features that help specifically with mobile development. We will see more about developer tools later in the Testing section.

When you build a web page, you are building something to be consumed by browsers, and so you must be aware of their capabilities, idiosyncrasies, and limits, especially on mobile.



Browsers and Rendering Engines

At the heart of every browser is a component that is responsible for laying out and rendering the content of a page. This is known as the rendering engine, or layout engine. Most modern web browsers are based on a small number of rendering engines.

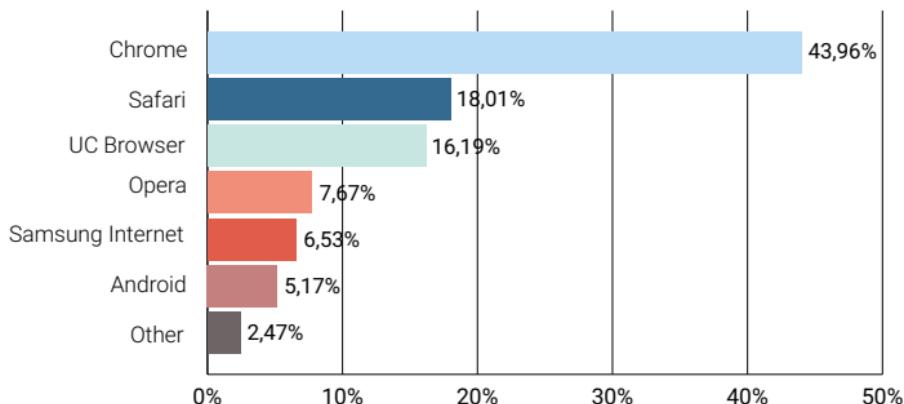
Knowing what rendering engine a browser is based on can help guide development and testing, since any browsers that share the same rendering engine will often behave similarly: they will generally support the same features, and at the same time fall foul of the same bugs.

- **WebKit:** the most widely used rendering engine today, it was built by Apple in 2001 and open-sourced in 2005
- **Blink:** in 2013 Google forked WebKit—which it had been using—to create Blink, which is now used in Chrome, Opera and Chromium based browsers
- **Gecko:** open source engine used by Mozilla's Firefox
- **Presto:** formerly used by Opera, and still used in Opera Mini
- **EdgeHTML:** used in Microsoft's Edge browser



What Browsers Should You Develop For?

One constant of the web is the ever-changing browser landscape. Browser popularity will vary from market to market and location to location. It is important that you have an idea of the browser market share in your target market, so that you can prioritize and optimize for these browsers. This said however, you should also try to maximize browser compatibility across the widest range of browsers where possible, since, except for in very limited or constrained circumstances, you will not know in advance what browser or device a user will use.



Worldwide mobile browser market share June 2016-June 2017 (source: gs.statcounter.com/browser-market-share/mobile/worldwide/#monthly-201606-201706-ba)

Reports such as DeviceAtlas' Mobile Web Intelligence Report¹ provide periodic snapshots of the device and browser landscape, highlighting interesting market share data such as the most popular OSes, manufacturers, and screen-

¹ deviceatlas.com/blog/most-used-mobile-browsers-q3-2016

sizes, and can help guide decisions relating to browsers and browser targeting.

HTML, CSS, and JavaScript: the Building Blocks of the Web

So far we have only covered devices and browsers. Now let us look at the technologies that are used to build web pages: HTML, CSS, and JavaScript.

HTML: Structuring Web Content

HTML is the markup language of the web. It is used to structure the content of a web page. Many variations have emerged over the years. Early mobile markup languages include WML, and XHTML Mobile Profile. The most recent iteration, HTML5, has matured sufficiently to encompass semantic markup, custom elements, and device APIs, and is well supported on mobile web browsers.

CSS: Styling Web Content

CSS stands for Cascading Stylesheets, and is used to style and lay out web content. CSS rules apply various properties, such as color, to elements by targeting those elements with a selector. Selectors allow you to pinpoint any element or group of elements in a page, so that you have full control: CSS can be used for small styling tasks such as setting the color of text, to large layout tasks that affect the entire page layout.

CSS has gone through multiple specifications, and is split into many modules, each with its own specification, and covering diverse aspects of CSS from selectors to 3D transformations and animations. CSS has become advanced enough that it can often deliver sufficiently interactive experiences without the need for JavaScript. Generally, for performance, you should

choose CSS over JavaScript for implementing interactions and animations where possible.

CSS preprocessors such as LESS and SASS are often used to extend CSS with operators and functions and other features that improve the development process and promote code reuse and maintainability.

CSS3 is well supported across the range of mobile browsers.

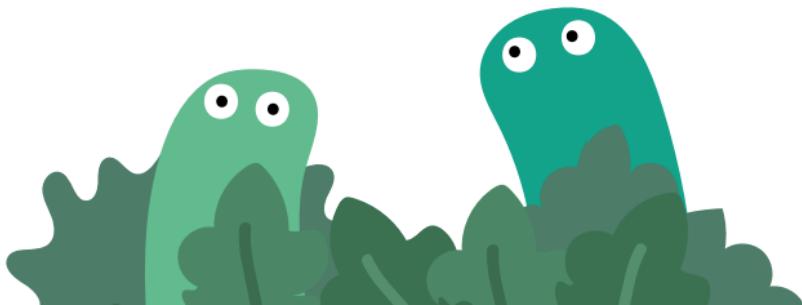
JavaScript: Client-side Scripting

JavaScript adds a programming layer to the web. It is used to bring more complex interactions, animations, and functionality to the web, and can be used to build web applications and games.

Originally only a client-side scripting language, JavaScript has grown and matured over the years as its underlying ECMAScript specification has evolved. It is now backed by a huge and vibrant community and a wealth of development tools, and has gained respectability as a programming language, and, through Node.js became available as a server-side language too.

Many thousands of JavaScript libraries have been developed, as well as entire front-end frameworks for building entire web apps, such as Angular.js, React.js, and Vue.js.

JavaScript libraries and frameworks can be a major source of page bloat, causing poor performance and bad UX, particularly on mobile. The mobile developer, therefore, should always be wary of adding JavaScript libraries to a project without good reason. We will see more about mobile web performance later.



HTML5

HTML5 is the most recent major version of HTML, and being well supported on mobile, brings many new features that enhance the mobile web user experience. But the term HTML5 refers to more than just the HTML markup language; it also refers to related web technology specifications that includes CSS3 and many useful JavaScript APIs.

Among the JavaScript APIs included in HTML5 are useful browser features such as the Canvas element, Touch Events, and Web Storage, to name a few. They also include device or hardware APIs that are particularly useful in the context of the mobile web. Device APIs give the browser access and control of specific device hardware features, such as the camera, the accelerometer, and GPS sensor.

HTML5 brings the browser closer to feature parity with native apps—things that were once only possible via native apps are now possible with the web. With location-based APIs web apps can know where the user's device is; through sensor APIs they can access camera images and compass bearings; with storage and caching APIs they can work offline. Websites can even send opt-in push notifications to users, even when the website is not open. With HTML5 the web experience got a whole lot richer.

Some of the more interesting HTML5 APIs from a mobile perspective are described below.

- **Geolocation API:** The Geolocation API gives access to the geolocation capabilities of a device, which can include precise location data based on the sensors of the device. Location can be based on the available sensors of the device, and includes GPS, A-GPS, Wi-Fi, and cell-based

triangulation methods. Latitude, longitude, altitude, heading, and speed data are exposed by the API.

- **Device Orientation API:** Like the Geolocation API this API returns information about a device's physical relationship with the world. While the Geolocation API is concerned with location in space, the Device Orientation API is concerned with orientation. The information is based on orientation sensors such as compasses, gyroscopes, and accelerometers. The data exposed includes orientation in three axes, acceleration, and rotation rate information.
- **Service Workers:** Service workers allow webpages to run scripts in the background. They can act as a proxy servers to web pages, intercepting requests and generating responses, and so they facilitate offline capabilities, background syncing, and push notifications.
- **Push API:** Allows web pages to receive push messages
- **Notifications API:** Allows a web application to display native-like notifications. Together with the Push API and service workers, web apps can send and receive asynchronous push notifications to a device.
- **Web Payments API:** Keying in credit card payment information has always been a laborious task on the web, but particularly so on mobile where input is more difficult. The Web Payments API aims to solve this problem, while at the same time removing the need to share your payment details with any third-party ecommerce site of unknown trustworthiness.
- **Touch Events API & Pointer Events API:** These are two APIs based around touch screen input, offering information about touches, and swipes etc.
- **Media Capture API:** Allows a web page to interact with a device's media sensors such as microphone and camera to capture audio and video.

Approaches to Modern Web Development

Modern web development practices must accommodate mobile devices and address the fragmentation issues facing mobile web developers. Several distinct approaches have emerged in recent years, each approaching the problem in a different way.

Responsive Web Design

Responsive web design (RWD) is an approach to web design that delivers resolution independent web pages—that is, flexible pages that will work well on most screen sizes, from desktop browsers to small screen mobile devices, and everything in between. It is based on three techniques:

1. **A flexible grid:** ensures the page layout scales with screen resolution, rather than using fixed dimensions
2. **Flexible images:** images that will scale within a flexible grid
3. **CSS media queries:** applies CSS rules to distinct ranges of resolutions or device classes, based on breakpoints

RWD became popular because a single HTML page designed this way could be expected to perform reasonably well on wide range of devices. It is basically a one-size-fits-all solution, which carries advantages and disadvantages:

Advantages

- Resolution independence can mean less time to implement and maintain
- No need to maintain and serve separate versions for different device classes
- Browser features can be detected by the client

Disadvantages

- Only achieves resolution independence, but not content adaptation, so content is not optimized for all devices
- More bytes than necessary can be sent to a device, which can impact performance
- Can perform poorly, or not at all, on low-end devices since the same content will be delivered to both desktop and mobile device.

Progressive Enhancement

Progressive enhancement is a technique that has been around for more than 15 years. The idea is that you start off with a minimal, base page to every device, along with JavaScript enhancement logic. A low-end device might ignore or fail to execute the enhancement, but will be still deliver a functional experience for the user. More capable smartphones, tablets and desktop browsers will execute the JavaScript enhancements progressively until the page is built up to an optimal level for the device.

This approach stands in contrast to the idea of graceful degradation, where rich functionality is built first, and exceptions are added afterwards. This requires additional work to ensure that a page is still functional in the absence of some feature.

In practice, you should consider Progressive Enhancement as a technique that can be used to smooth over differences in a range of mobile devices, rather than as an overall approach.

Advantages

- In theory no limit to the features that can be progressively added
- Can cater for full range of devices from low-end to high-end and desktop

Disadvantages

- Using a single base for all devices can be restrictive
- The actual progressive enhancement JavaScript takes time to execute, and can impact performance

Mobile-First Responsive Design

Mobile-first RWD follows the design principles of both RWD and progressive enhancement techniques. In this approach the design begins with a mobile-optimized version, and builds up from there.

Advantages

- Increases reach over pure RWD since it is more likely to work on lower-end devices
- Forces designers to focus on content and functionality, making it easier to define content hierarchy, with the most important content at the top

Disadvantages

- Suffers the same issues as RWD
- May require complete redesign of existing site, if existing site does not follow this approach

Adaptive Web Design (Server-Side Adaptation)

Server-side adaptation² uses a device detection solution on the server to map a device's request headers to a database of device capabilities. Once the device's capabilities are known, a page can be built to match the device capabilities resulting in highly optimized pages. Leading solutions include DeviceAtlas³ and ScientiaMobile⁴.

Sometimes referred to as "browser-sniffing", the effectiveness of this technique is evident in its adoption by most of the major internet brands that take their web presence seriously, including Google, Amazon, YouTube, Facebook, and Ebay.

Advantages

- Knowing the capabilities of a device means that highly optimized pages can be delivered for multiple device classes
- Extremely reliable and accurate: good solutions report over 99.5% device detection accuracy
- Excellent performance since pages can be fine-tuned for devices

² There are various definitions of adaptive web design; here we mean that there is some server-side adaptation taking place.

³ deviceatlas.com

⁴ scientiamobile.com

Disadvantages

- Involves developing and maintaining different page templates for different device classes
- Database of User-Agent capabilities must be updated with new devices
- Most solutions are commercial

RESS: A Hybrid Approach

One last approach to consider is RESS (REsponsive design with Server Side components). RESS combines the adaptive and responsive approaches to deliver a solution that combines the best of both. Using server-side adaptation an initial page is optimized for a range of devices or device category. Then, within each category, the content can be adapted further on the client-side using responsive techniques. This approach can be pushed further still, so that properties source from the browser can be fed back to the server to further tune the server adaptation.

Advantages

- Offer most flexibility and highest degree of optimization of all solutions
- Benefits of high-performance server-side adaptation, combined with ability to tweak with properties obtained on the client-side

Disadvantages

- Difficult to implement, requiring device database
- Full round-trip required to get the most benefit

Hybrid Apps

Another class of web application worth a mention is hybrid apps. These are often, but not necessarily, built with web technologies, HTML, JavaScript and CSS. Hybrid apps are compiled and packaged as native apps, and distributed in native app stores. They are installed like native apps, but are essentially web apps on the inside. Generally they consist of a full screen webview, or thin native app wrapper and a webview. The webview does the heavy lifting of rendering the web app, while the native library gives access to native APIs and hardware. This is an attractive approach to many since it is possible to leverage web development know-how for native apps without having to learn native platform development. Various hybrid app development frameworks exist to help the developer, including Apache Cordova⁵ and PhoneGap⁶, and React Native⁷.

It is possible to have an app that is built with web technologies, but that is compiled down to the native code for the platform, for example with React Native. Unless this app is using a webview it is not really a hybrid app, at least in the traditional sense. React Native is essentially pushing the boundaries of what is considered a hybrid app. With React Native apps are compiled into native code for each platform, iOS and Android, and widgets are rendered as native platform widgets, while Cordova/PhoneGap apps use use webviews to render web content within a native app shell.

Please see the chapter about Cross-platform apps for more information around hybrid and cross-platform apps.

⁵ cordova.apache.org

⁶ phonegap.com

⁷ reactnative.com

Advantages

- Cross-platform: A native app shell uses a webview to render content from a web based content backend
- Does not require intimate development knowledge of multiple native platforms
- Easier to develop and maintain than distinct native apps for each native platform
- Using a remote backend means the app can be updated without having to resubmit to app stores

Disadvantages

- Performance is not as good as a native app for demanding applications
- UX might not be as polished as a full native app, as compromises are made to satisfy cross-platform constraints

Progressive Web Apps

Progressive Web App (PWA) is a term used to describe web apps that make use of modern browser features to deliver rich app-like experiences. The term was first coined by Alex Russell⁸ in 2015 to describe web apps that exhibit the following criteria:

- **Progressive:** they work on all devices, and functionality is enhanced progressively
- **Responsive:** layout is flexible and can adjust with device form factor as appropriate

⁸ Find Alex' blog at infrequently.org

- **Connectivity-independent:** will function under poor network conditions, and even offline
- **App-like:** feels like an app, with an app-shell, and mostly without full page refreshes
- **Fresh:** pulls in new content whenever possible
- **Secure:** served over HTTPS
- **Discoverable:** identifiable as an app while also being indexable by search engines on the web
- **Re-engageable:** stimulate re-engagement with features like push notifications
- **Installable:** can be added to the home screen of a device

The reason PWAs have risen to prominence is because the web platform has grown sufficiently mature to deliver such experience. As we saw earlier, HTML5 has become advanced enough to support features like push notifications, geolocation, and offline experiences. These are all features that previously would have been found only in native apps. Several key HTML5 APIs make PWAs possible:

- **Service workers** for offline and network-challenged conditions, as well as handling push message notifications
- **Push Message API**, and **Notifications API** provide a mechanism for delivering asynchronous push messages
- **Web App Manifests** provide a mechanism for meta description of a web app, that helps with indexing and adding to the home screen of devices

Support for PWAs relies on support for the underlying browser features. Of the main mobile browsers, PWAs are supported by Chrome, Firefox, Opera, and Samsung Internet browser, and is coming soon to Edge.

For a long time it was uncertain whether Safari would sup-

port PWAs. However, implementation of service workers, a key PWA component, began in WebKit early in July 2017. WebKit, once the poster boy of the browser engine world is already being referred to as the new Internet Explorer as it lags behind the rest in the implementation of new features and standards, causing headaches for developers.

Accelerated Mobile Pages (AMP)

Google's Accelerated Mobile Pages (AMP) project⁹ is a publishing format based on an open source web components framework with an emphasis on performance. AMP was originally conceived as a response to Facebook's Instant Articles and Apple's News projects, and so initially the emphasis was on news and blog style content.

However, since its launch in 2015 it has evolved to support a much wider range of content, with interactive features such as carousels, image galleries, interactive menus, and a programming model that supports complex ecommerce features.

AMP is an answer to poor performance of mobile pages; it is designed to have <1sec page downloads, and is often a lot faster.

⁹ ampproject.org

What is AMP exactly?

There are three parts that make up AMP:

1. **AMP-HTML:** A flavor of HTML5, which both restricts the tags you can use, as well as adding some new ones
2. **AMP-JS:** A JavaScript library that functions as the AMP runtime, which orchestrates the optimized loading and rendering of AMP pages
3. **AMP-CACHE:** A special cache for AMP pages that enables AMP pages to be rendered instantly in some cases

Getting started with AMP is not too difficult since AMP pages are basically HTML. Every AMP page starts off with some standard boilerplate code that includes the AMP-JS runtime. When the AMP runtime parses the AMP-HTML page and comes across any AMP components, it will inject generated markup into the DOM to replace the AMP-HTML markup; this is what gets rendered in the browser.

Canonical AMP Pages

All valid AMP pages must include a canonical link tag. This tag should point to the equivalent, non-AMP version of the page if it exists. If there is no non-AMP equivalent, then the canonical link must point to itself. This is known as a canonical AMP page—a standalone AMP page that serves as both the mobile and the desktop web page.

AMP supports responsive design and media queries, and so, despite its name (Accelerated Mobile Pages) it is not a mobile-only technology. It could be better described as mobile-first: optimized for mobile but can scale responsively to support larger viewports.

The Canonical AMP approach is being promoted by the AMP team as a web strategy suitable for a wide variety of business. To this end the AMP team launched a site, ampstart.com, dedicated to publishing free to use and modify responsive canonical page templates along with reusable AMP UI components. These components include styled buttons, forms, image carousels, navigation and other common UI components.

Combining AMP and PWAs

AMP and PWAs have complementary strengths and different weaknesses. It is therefore natural to consider combinations of the two. Several different patterns have been identified that combine the speed of AMP with the richness of PWAs:

AMP as PWA

In this pattern, the AMP page is the PWA. It uses the AMP library, so that a valid AMP page can be served from the AMP Cache, resulting in lightning fast pages. When links are followed however, the user is brought to the original server, where a service worker can now be used.

AMP bootstraps PWA (aka AMP up aka AMP to PWA)

In this model, the AMP page uses a special component, `<amp-install-serviceworker>`, to install a service worker in the background on the users device. The service worker can then bootstrap the PWA by downloading and caching initial parts of the PWA, so that when the user follows a link to the full PWA, it is already downloaded and ready to display.

AMP data embedded in PWA (aka AMP down aka AMP in PWA)

In this pattern AMP pages are used as the content backend, within a PWA shell.

The Physical Web

The Physical Web¹⁰, is an open source project that aims to enable quick and seamless interactions with physical objects and locations, via Bluetooth beacons.

Apple has its own, proprietary Bluetooth beacon technology called iBeacon. iBeacon differs from the Physical Web in that iBeacons trigger apps, while Physical Web beacons trigger web URLs. The biggest advantage of the Physical Web over iBeacons is that no app is needed, while an app must be installed prior to interacting with an iBeacon. This significantly lowers the barrier to use of the Physical Web, and increases its reach, without any prior setup, to billions of Bluetooth enabled devices.

Getting started with the Physical Web is remarkably easy. Just get a beacon, set it up to point to a URL, that's it! Users with compatible Bluetooth enabled devices will then receive notifications where they are nearby. This is a simple idea that enables a wide variety of applications, like smart vending machines and targeted marketing in physical spaces.

Google also has a more complex beacon platform¹¹ with remote cloud management of beacons, which can offer scalability and other benefits for large beacon deployments.

¹⁰ google.github.io/physical-web

¹¹ mobiforge.com/design-development/googles-beacon-platform-and-the-physical-web

Web Performance and Why it Matters

Most developers intuitively know that web performance is important; nobody likes to wait for a page to load. But there is plenty of empirical data to prove that performance is crucial, especially on mobile:

- Walmart experiments found that for each 1 second faster page load time there was a 2% increase in conversions¹²
- Amazon research found that a 100ms latency increase resulted in 1% fewer sales¹³
- A Google study reported that 53% of visitors will leave before a page has loaded if it takes longer than 3 seconds¹⁴

Depending on which report you read, if your site is slow, you will lose over half your visitors; they just will no wait for it to load, and you will not have a chance to show them what you have to offer. You have a tough audience to please, and the only way to succeed is to deliver a good user experience; and to do this, you will need to deliver a fast site.

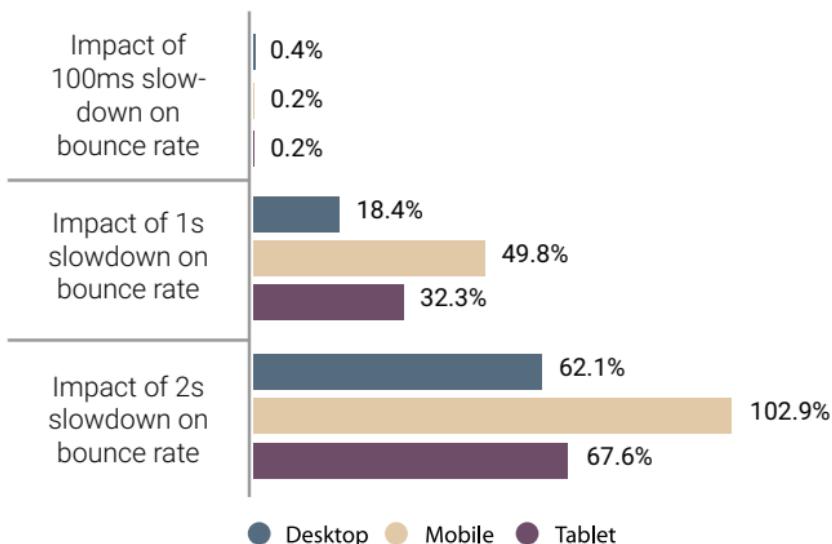
To illustrate this, here are some statistics on page slowdowns on bounce rates:

¹² webperformancetoday.com/2014/04/09/web-page-speed-affect-conversions-infographic/

¹³ blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

¹⁴ doubleclickbygoogle.com/articles/mobile-speed-matters

Impact of page slowdowns on bounce rates (by device type)



(Source: soasta.com/wp-content/uploads/2017/04/State-of-Online-Retail-Performance-Spring-2017.pdf)

Performance Goals

The idea of a performance budget has been around for a few years; during planning a "budget" is set on different aspects of how a page should perform, and you try to stick to this budget during development. The specific dimensions of a performance budget might include restrictions on page weight, number of HTTP requests, page load time, time to initial interactivity and so on. If you can not meet the budget, then you need to consider the assets or features that are blowing the budget. Do you really need that fancy image carousel or JavaScript library for instance?

RAIL

Google defines an approach and set of goals for web performance called RAIL¹⁵ : Response, Animation, Idle, Load. The goals it aims for are:

- **Response:** 100ms—give immediate feedback to user input
- **Animation:** 10ms—when scrolling or animating produce frames in under 10ms to achieve 60fps
- **Idle:** 50ms—non-critical operations should not take longer than 50ms so that application feels fast
- **Load:** 1s—deliver interactive content in under 1000ms to keep users engaged

It is not always possible to achieve all these goals together, especially on low-end devices, so it is sometimes necessary to prioritize these goals, focusing on Load and Response first for example.

Analytics

Analytics is vital to understanding your visitors and traffic. It can be particularly useful on mobile to help you understand what devices your users are using. For many, however, analytics starts and stops with installing a Google Analytics script. But Google Analytics is not the only show in town.

Analytics tools can collect their data on the client or on the server. It is worth noting that relying solely on JavaScript based analytics can be problematic, especially on mobile. If a device fails to run the analytics script—for example if it is an older device—then you will have no visibility of this device at all, and it can lead you to focus on the wrong devices.

¹⁵ developers.google.com/web/fundamentals/performance/rail

Additionally, many ad blockers also block client-side analytics such as Google Analytics. If you are more serious about your analytics, a more accurate picture of your data can be gained by employing a combination of both client and server-side analytics.

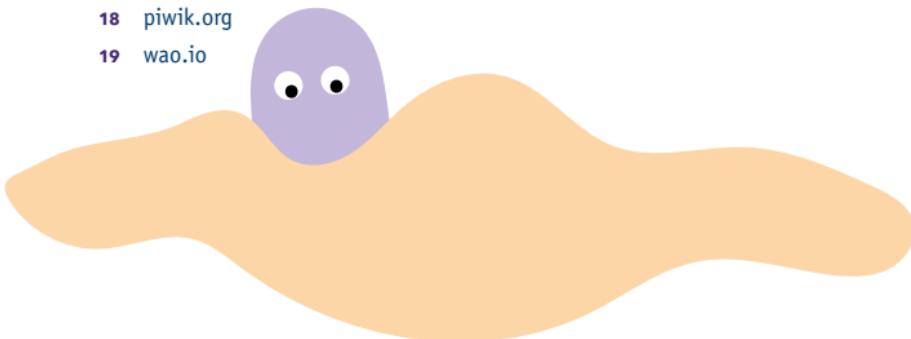
Popular tools include Google Analytics¹⁶, KISSMetrics¹⁷, and Piwik¹⁸. Some tools, such as wao.io¹⁹ provide both server and client side analytics.

¹⁶ analytics.google.com

¹⁷ kissmetrics.com

¹⁸ piwik.org

¹⁹ wao.io



A/B Testing

A/B Testing is a very useful technique that can be used in web development to evaluate the performance of alternative interface design, messaging and layout variants so that you can make informed decisions about the UX you deliver. Most of the main analytics tools support A/B testing, including Google Optimize, Google Analytics (Experiments), and Piwik²⁰.

Real User Monitoring (RUM)

RUM involves continuous monitoring of user interaction with a web application in real time. This allows the website owner to quickly pinpoint, prioritize and remedy issues with availability, functionality, responsiveness, and so on. This in turn gives valuable insights into user behavior and satisfaction and enables you to fine-tune and improve the overall user experience. RUM can take place client-side or server-side and plug-in tools such as Sevenval's *wao.io* can correlate front-end and back-end metrics without having to make any code changes to the application.



²⁰ piwik.org

Monetization

Ads

Ads have traditionally been one of the most common ways to monetize a website, and no less so on the mobile web. There are plenty of ad networks to choose from.

Ad blockers

If you choose an ad-based monetization model, note that there has been a growing backlash against ads—particularly on mobile—since Apple added support for ad-blockers into Mobile Safari in 2015. Ads have a bad reputation for adding unnecessary page bloat, and degrading web performance. Mobile and desktop browsers are increasingly shipping with built-in ad blockers and enhanced privacy protection. This trend is likely to increase in the coming years, especially in light of Google's recent announcement that it would implement an ad blocker in its Chrome browser. Therefore, if you are relying on ad revenue, be sure to know the risks and potential downsides.

When choosing an ad network, pick a reliable one that will not harm the performance of your site. A relative newcomer to the field is AMP for Ads (A4A)²¹. These are ads that are based on AMP technology, but will work fine for desktop too. A4A has strict rules about what is permitted and performance-degrading ads will be removed from a page, so you can be confident that ads will not cause UX or performance issues.

²¹ see ampproject.org/learn/who-uses-amp/amp-ads and github.com/ampproject/amphtml/blob/master/extensions/amp-a4a/amp-a4a-format.md

Ecommerce

There are many ways to build a web ecommerce solution, ranging from off-the-shelf solutions to custom developments. Many new and traditional payment service providers, such as PayPal and Stripe, have good mobile support and are relatively simple to implement. Additionally a new breed of NFC-enabled mobile wallets, such as Android Pay, Apple Pay, and Samsung Pay, can increasingly be used on the web.

Payment Request API

One final technology to watch is the Payment Request API. This is a recent HTML5 specification that aims to provide frictionless payments on the web, in two ways

1. Removing the need for cumbersome input of payment details
2. Removing the requirement to share your credit card details with third-party sites of unknown trustworthiness

Several browser already support the Payment Request API, include Chrome for Android, Edge, and Samsung Internet, and coming soon to Firefox.

Please see the Monetization chapter for more detail on the types of monetization models that can be used.

General UX and Performance Guidelines

We do not have the space here to go into much practical detail here²², but a general set of guidelines for mobile web development includes the following.

UX

- Optimize for mobile
- Do not require pinch-to-zoom
- Make product images expandable
- Ensure tap targets and links large enough for "fat" fingers
- Keep calls to action front and center
- Keep menus short
- Intuitive navigation with prominent link to homepage
- Avoid interrupting pop-ups and large interstitials
- Include a site search, with filters to narrow results where possible
- Include click-to-call where possible
- Only ask for special browser permission when needed, not up-front. E.g. Ask for permission to send push notifications when a user has already indicated a desire to subscribe, or say for updates on an order, and geolocation only when needed for mapping or address.
- Let users browse as guests
- Let users purchase as guests
- Autofill form inputs where possible

²² Check this site for some graphical examples: <https://developers.google.com/web/fundamentals/getting-started/principles>

Performance

- Keep page weight small, do not include unnecessary media such as images and videos
- Compress all images
- Avoid redirects
- Keep number of external resources low, to reduce HTTP requests
- Implement caching
- Lazy load images and content as needed
- Minify text resources
- Avoid JavaScript and CSS frameworks unless necessary
- Avoid unnecessary embeds and includes
- Use ads responsibly, and only use lightweight ads
- Define a performance budget, and try to stick to it

Testing for the Mobile Web

Testing on the web is crucial. And on the mobile web, while simulators, and even desktop browsers are useful, the most reliable testing is that performed on real devices. Earlier we talked about device fragmentation. This, combined with the variety of browsers available on each platform makes for a headache when it comes to testing your web pages. It is next to impossible to exhaustively test on all browsers and devices. None but the biggest and best funded projects will be able to even come close. So, when it comes to testing the mobile web you need to prioritize.

If your budget can afford it, it is recommended to acquire several devices on each of the main mobile platforms, Android, iOS, and Windows. Devices should be a mix of high and low-

end, and with the ability to switch between mobile networks if possible.

Broadly speaking, testing effort is divided between functional and UI testing on the one hand, and performance testing on the other. Functional and UI testing is concerned with testing business logic, user interface, UI components and usability. Sometimes issues will only show up on some devices on some platforms, and this makes testing on mobile difficult.

Performance testing is more concerned with how well the site works: is it fast, does it feel fast, does it work under poor network conditions?

Manual testing will likely be the first step in any web testing. While there are great efficiency gains to be achieved through automated testing, automated testing is not always practical, perhaps due to the small size of a project, or time or budget limitations.

Thankfully, there are plenty of tools to help with all aspects of mobile web testing.

Browser Developer Tools

Even for mobile web first pass testing is often carried out on a desktop browser with its built-in developer tools. Just some of the features of a browser's developer tools include DOM inspection, network inspection, performance profiling

- **DOM inspection:** allows examination of the HTML elements that make up a page
- **Network throttling:** for simulating slow and poor network conditions, such as Edge, 2G, 3G and so on
- **CPU throttling:** to simulate high-end or low-end smartphones
- **Waterfall chart and timelines:** provide visualizations of how a page loads and renders over time

- **JavaScript debugging:** allows the developer to examine, add breakpoints, view events, and step through JavaScript code
- **CPU and Memory profiling:** records how the CPU performance and RAM footprint over time

Other features particularly useful for mobile web development and testing are

- **Responsive design mode:** simulates viewports at a variety of configurable sizes, so that you can see how the UI behaves under differently sized viewports
- **Screen mirroring:** a connected device's browser is mirrored to the developer tools, and can be interacted with via the desktop browser

Remote Debugging

All of the major mobile platforms support remote debugging of mobile devices. Remote debugging allows you to attach a mobile device to a desktop machine and apply the developer tools of that machine's browser to test and profile the web pages on the mobile device.

Remote debugging is an extremely useful tool, since it allows you to test on real devices, and on real networks. Of course, you still need devices to test with, and that can get expensive. At the very least, you should be looking at having a low-end and high-end device on each of the main mobile OSes: Android, iOS, and Windows Phone. Even then, there will be major gaps in your testing coverage; this is where device labs can help.

Performance and UX Testing Tools

Selenium WebDriver

Selenium WebDriver²³ is the leader in automated web testing. Automated testing is very useful for quickly finding issues with user interfaces, and can be used for regression testing to quickly find breaking interface changes.

Selenium additionally supports mobile testing²⁴ on Android and iOS, and both simulator and real device testing is supported.

Webpage Test

WebPagetest²⁵ is an open source and free-to-use performance testing tool which offers remote testing on real desktop and mobile browsers at different locations around the world. It provides waterfall performance charts, as well as measurement of key performance metrics such as time to first byte, speed index, and number of DOM elements.

mobiReady

MobiReady²⁶ is a free tool for developers, designers, and marketers that tests web pages and sites for mobile-readiness based on mobile web best practices and standards. It returns a detailed analysis for a page, and offers recommendations on how to address any detected issues. It also includes:

²³ seleniumhq.org/projects/webdriver

²⁴ github.com/SeleniumHQ/selenium/wiki/WebDriver-For-Mobile-Browsers

²⁵ webpagetest.org

²⁶ mobiready.com

- device visualizations on low, mid, and high-end devices, showing how the page will look on a variety of screen sizes
- breakdown of page weight for each device class
- a benchmark report of how your page scores against the Alexa top 1000 sites

MobiReady also exposes an API that can be used for automated testing of entire sites.

Lighthouse

Lighthouse²⁷ audits a web app for PWA features, including:

- can it load offline or under poor network conditions?
- is it fast?
- is it served from a secure URL?
- does it implement accessibility best practices?

Lighthouse is available as an online service, a Chrome extension, a command line tool, and has recently been integrated with Chrome developer tools. The command line version is useful for automated testing.

PageSpeed Insights

PageSpeed Insights²⁸ is a tool from Google that measures page performance for mobile and desktop visitors. It checks for common performance best practices, and ranks pages out of 100. When issues are detected, it offers advice on how to fix them.

²⁷ developers.google.com/web/tools/lighthouse/

²⁸ developers.google.com/speed/pagespeed/insights

Device Labs

The idea of a Device Lab has been around for many years. A device lab is simply a collection of devices that can be used for development and testing. Device labs fall into two categories: physical and remote.

Remote labs offer the most convenience: you install a client on your computer and it allows you to access real device remotely over the web. On the other hand, with a physical lab, you can hook a device directly up to your laptop and use the browser's remote debugging tools. This can be helpful for solving issues that affect specific devices.

AWS Device Farm

Amazon's AWS Device Farm²⁹ is an advanced device testing lab that features automated testing against a large collection of real devices in the AWS cloud, as well as direct remote access that allows you to interact with swipes and gestures in real time from your web browser. AWS Device Farm contains a wide variety of new and old iOS and Android devices.

AWS Device Farm supports pay-as-you-go and flat-rate charging. On the PAYG model, the first 1000 minutes are free, so it can be a good way to try the service before committing to it.

BrowserStack

BrowserStack³⁰ offers remote testing on a variety of desktop and mobile browsers, and different operating systems. The mobile devices have been chosen for "maximum market coverage" and includes a vast array of real iOS and Android devices. Supports automated testing via Selenium cloud testing.

²⁹ aws.amazon.com/device-farm

³⁰ [browserstack.com](https://www.browserstack.com)

Browserstack is free for open source projects, and offers a free trial for commercial projects.

Samsung's Remote Test Lab

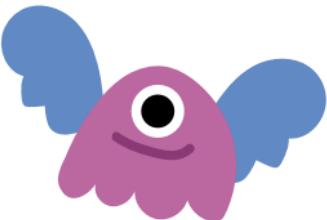
Samsung's Remote Test lab³¹ offers free remote testing on Samsung devices. You reserve and connect to real devices through your web browser. The device list includes old and new phones, tablets, and watches—not surprisingly all Samsung devices—and spread across its Galaxy, Z (Tizen), and Gear brands.

SIGOS App Experience

SIGOS App Experience³², formerly known as Keynote Mobile Testing, formerly known as DeviceAnywhere (it's hard to keep up!) was one of the first Virtual Device Labs. Despite its latest name, SIGOS App Experience can be used to test on the web as well, on its more than 2000 devices. SIGOS supports both manual and automated testing. The service is accessed via a desktop browser. Offers a 7 day free trial.

Perfecto Mobile

Perfecto Mobile³³ offers paid-for remote testing on real devices. Supports manual and automated testing on multiple devices. A free trial is also available.



³¹ developer.samsung.com/rtlLanding.do

³² appexperience.sigos.com

³³ perfectomobile.com

Open Device Lab

The Open Device Lab³⁴ is a community approach to device labs. Participating organizations offer a physical space where developers can go and use for free. There are currently 154 Open Device Labs across 35 countries offering free access to 4255 devices.

Resources

- Resource site for mobile web design and development: *mobiforge.com*
- Browser feature support and compatibility: *caniuse.com*
- Responsive website design with RESS: *smashingmagazine.com/2013/10/responsive-website-design-with-ress*
- Your first Progressive Web App: *developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp*
- Offline Web Applications: *udacity.com/course/offline-web-applications--ud899*
- The AMP project: *ampproject.org*
- Progressive Web AMPs: *smashingmagazine.com/2016/12/progressive-web-amps*
- The Physical Web project: *google.github.io/physical-web*
- A good perspective on software quality and testing using web browsers: *mobiletestingblog.com/2017/05/01/recent-web-browser-quality-related-innovations*
- High Performance Mobile Web - Best Practices for Optimizing Mobile Web Apps, by Max Firtman (O'Reilly Media, 2016)³⁵

³⁴ opendevicelab.com

³⁵ available via <http://shop.oreilly.com/product/0636920035060.do>





Enterprise Apps

Corporate decision makers now view mobile enterprise apps as a strategic factor, a necessity, rather than an item on an accountant's spreadsheet. Internal enterprise apps are able to reduce the latency of information transfer within a company. They increase the agility of the worker by making competitive data & big data available at any time and anywhere. Apps can also allow companies to engage with its customers, suppliers, and end consumers etc. Examples of enterprise apps include field & sales staff software, emergency response, inventory management, supply chain management but also B2C marketing.

It may seem an obvious thing to say, but the major risk at the moment, is **not** having an enterprise mobile strategy. Business is now looking at **Mobile for All** rather than limiting it to senior management, as it may have been in the past. To enable this the traditional IT approach of buying devices and distributing them throughout the management structure is no longer the only enabling strategy being used; we have moved from Bring Your Own Device (BYOD) to BYOx including apps, content, development tools/frameworks and now even wearables, enabling staff to use their personal devices to connect to the IT infrastructure, download secure content and use enterprise apps. With the advent of BYOx, a company exposes itself to risks which traditionally have never been part of the corporate IT strategy. Early adoption of a well thought out and implemented enterprise mobile strategy is key to ensuring data is secured at all times.

And from a developer's point of view, the enterprise sector has a lot to offer as well: Compared to traditional B2C app developers those who create enterprise apps are twice as likely to be earning over \$5k per app per month and nearly 3 times as likely to earn over \$25k according to the Developer Economics report¹.

Key points for Mobile Apps in Shaping the new Business Enterprise

- Cost reduction compared to existing systems
- Streamlining business processes
- Competitive advantage with up-to-date data immediately at hand
- Increase employee satisfaction and effectiveness
- Rapid response compared to existing processes
- Analyzing and utilizing Big Data



¹ www.developereconomics.com/report/next-gold-rush-enterprise-apps

Enterprise Strategy

Many companies nowadays have a Chief Mobile Officer (CMoO) or have extended their CIO position. It is their job to co-ordinate mobile trends and directions and to bridge the gap between business and IT. Depending on the size and main focus of the company, his/her job is also to either build up an internal mobile software development team or coordinate the cooperation with an external development agency. To make sure that the mobile software delivers what the employees / users want, that this is technically achievable and that everything fits the overall company strategy, the leader might consider setting up a Mobile Innovation Council (MIC) or Center of Excellence (COE). This group should contain key members such as: skilled representatives from the mobile development team, stakeholders for mobile within the company, and most importantly end users from various departments with expertise in the relevant business processes.

Topics that the CMoO/CIO needs to focus on together with the MIC/COE include:

- **Strategy** - vision and direction for the general mobile strategy and for the apps.
- **Governance policies** - Bring Your Own Device (BYOD) vs. Choose Your Own Device (CYOD) which is essentially the difference between a Mobile Application Management (MAM) policy (BYOD) and a Mobile Device Management & Security (MDM) policy (CYOD).
- **App specifications**
- **App road map**
- **Budget planning**
- **Acceptance** - signing off the apps into production.

- **App deployment** - early feedback on demos and prototypes, testing, mass deployment.
- **Incentives** - how to increase the adoption and usage of the apps created.

In commercial adoption terms enterprise app development is mostly mainstream now. The question a company writing third party enterprise apps, or a development manager keen to adopt an internal mobile enterprise strategy used to be "This all sounds great, but why do we need it?". This has now become "Mobile will give us a competitive advantage and empower our workforce" which is a compelling reason for a company to adopt a mobile strategy.

Key points when building the business case for Mobile Enterprise Apps

- Create a visionary plan for more mobile apps, on various devices and know how they will aid, shape and empower your enterprise.
- Create an ADS (Application Definition Statement) for each app, specifying purpose and intended audience.
- Create a budget for devices & device upgrades.
- Create a plan for an application & device management strategy & security infrastructure.
- Create a plan for an app dev team using a future proof development architecture - such as a MADP, frameworks etc.

UI/UX Design And The Enterprise

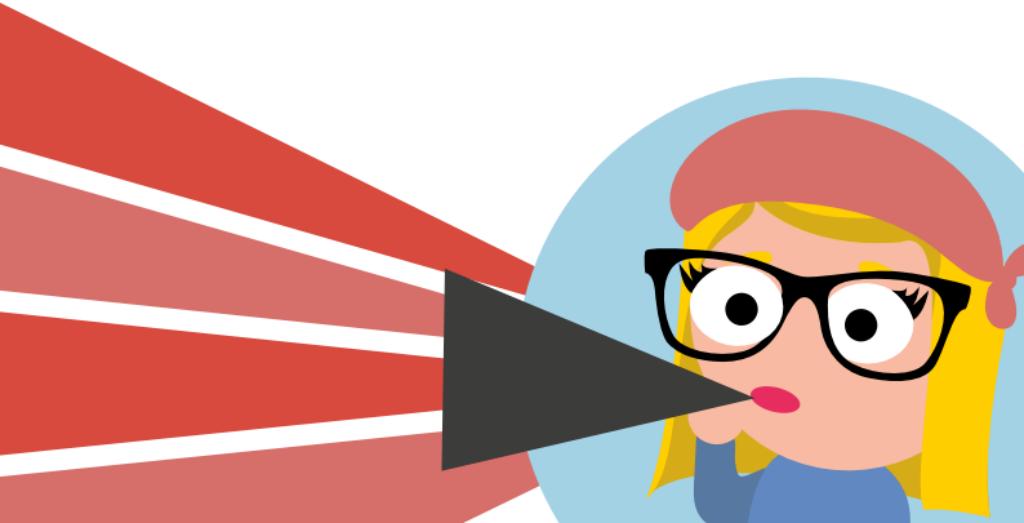
Mobile users have been using their devices for many years now. Some starting out originally as a consumer, but now as mobile is used more and more in the enterprise, they have taken that experience and expectations with them, as end users to business systems as they are mobilized. They have become the Prosumer, the Professional Consumer of enterprise apps. This means, they have relatively high expectations when it comes to user experience and you should take this into account.

Your enterprise app, whichever mobile device it runs on, should follow the standards for that operating system, such as the Human Interface Guidelines or HIG from Apple. There are a number of reasons for why your designers should design following these guidelines, both Apple and Google have great documents which can be found at developer.apple.com/ios/human-interface-guidelines and at developer.android.com/design/index.html.

If you build an enterprise app that responds or interacts in unfamiliar ways users will probably respond negatively and/or will need training on the new functionality you have taken so much care to implement. This is why you should make sure that your app feels complementary to the mobile OS and not competing with it, therefore making the user intuitively feel familiar with your software and its usage patterns. So please read our chapter on UI/UX design carefully and apply the learnings when creating your solution.

Mobilizing Existing Systems

If you are already providing a system to customers which has not yet been mobilized, you will have various decisions to make. It is critical to fully understand the impact of adding a mobile offering to your system before you start implementation of the solution. Common reasons to mobilize your product can include using phone features, such as camera and GPS, or just the ability to capture information on the move, without being connected to the internet. You must ensure you go mobile for the right reasons, as the ongoing support, maintenance and development of a mobile offering will become a separate product road map to your original system and will carry an on-going cost.



Key points when deciding how to mobilize an existing system

- Clearly define the reasons for going mobile and ensure that those reasons are strong enough to take the step into mobile.
- Understand the difference between mobile and desktop. Do not just copy your existing system, so for instance, instead of a form to capture information, you could capture audio and upload that into your system, allowing a user to quickly make notes without the need to type into a small device.
- Do not try and implement all the features of your existing system; implement the important features in a way which suits mobile.
- Ensure you understand which devices your clients use and which features of your system are most required to be mobilized.
- Have a clearly defined mobile testing strategy which covers cross platform testing and multiple device types and operating systems.

Device And Application Management In The Enterprise

When developing an enterprise app, you should always keep in mind that the hardware containing sensitive company data can get lost or stolen. There are now two approaches for securing devices, content and apps. Mobile Device Management (MDM) and Mobile Application Management (MAM). These are now coming together as Enterprise Mobility Management (EMM).

MDM gives an enterprise ultimate control over a device, so when a device is lost, stolen or an employee leaves, taking the device, the enterprise can wipe the device and essentially stop the device from working. This approach is usually taken when an enterprise owns the device so all the data and apps on the device are owned by the company; any personal data stored on the device is stored at the employee's risk.

MAM enables an enterprise to adopt BYOD as it allows an enterprise to secure apps and content downloaded to a device without taking ultimate control away from the owner of the device. When an employee leaves a business, taking their device with them, the business can disable the enterprise apps and wipe any content downloaded to the device without affecting personal data, such as photos and consumer bought apps. Most MDM and MAM solutions are cross platform, supporting multiple devices, and this should always be taken into consideration when deciding upon an MDM or MAM provider.

Various security features are available through both these management solutions, including:

- Device monitoring
- License control
- Distribution via an internal Over-The-Air (OTA) solution
- Software inventory
- Asset control
- Remote control
- Connection management
- Application support & distribution

Security measurements include

- Password protection
- On-device data encryption
- OTA data encryption
- Remotely lock devices
- Remotely wipe data
- Re-provision devices
- Back-up data on devices

Examples of EMM providers are:

- **Airwatch:** *air-watch.com*
- **App47:** *app47.com*
- **Apperian:** *apperian.com*
- **Good:** *good.com*
- **Microsoft:** *microsoft.com/en-us/windows/windowsintune*
- **MobileIron:** *mobileiron.com*
- **Mocana:** *mocana.com*
- **SAP Afaria:** *go.sap.com/product/technology-platform/afaria-mobile-device-management*
- **SOTI:** *soti.net*

Mobile Application Development Platforms (MADP)

Usually, one key element of enterprise applications is data synchronization. The mobile devices have to be refreshed with relevant or up to date data from the company's servers and the updated or collected data has to be sent back. The scope of data access is determined by the job responsibilities of the user as well as by confidentiality policy. In any case synchronization has to be secure, as corporate data is one of your most prized assets. Furthermore, a company-wide accepted app will be multi-platform.

To compensate the shortcomings of the native SDKs as well as the common multi-platform solutions in these regards, you might want to consider evaluating Mobile Application Development Platform (MADP) solutions. MADPs are mobile development environments that provide the middleware and tools for developing, testing, deploying and managing enterprise apps running on multiple mobile platforms with various existing back-end data sources. Their aim is to simplify development and reduce development costs, where skills must be maintained for multiple platforms, tools and complexities, such as authentication and data synchronization.

Available solutions include:

- **IBM MobileFirst Platform:** www.ibm.com/mobilefirst
- **KonyOne:** www.kony.com/products
- **Pega Amp:** www.pega.com
- **SAP Cloud Platform Mobile Services:** cloudplatform.sap.com/capabilities/mobile
- **Spring Mobile Solutions:** www.springmobilesolutions.com

Security In Enterprise Apps

One of the main functions of any IT department is to ensure that all aspects of the company infrastructure is secured against attack so that there are no data leaks and no data is compromised or stolen. As mobile devices are an extension of a company's IT infrastructure, all enterprise apps must be designed to ensure that they cannot be used to illegally gain access to a company's internal network. As an enterprise app writer you will usually be asked to conform to standards which a company has laid out in their security policies, so be prepared to answer questions about securing your app, such as data encryption, network communication and dealing with jail broken or rooted devices.

Many EMM providers actually enhance app security, using techniques such as app wrapping or providing an SDK which app writers can use. These features and regular updates of these platforms, allow an enterprise to lock down their apps remotely and also keep up with the changing security landscape without needing to invest as much time and effort into security.

Key points for securing Enterprise Apps

- If using an EMM provider ensure they have the security required security features to meet your enterprise standards.
- When storing any data on the device ensure it is encrypted.
- When communicating with web services, always use https.
- In addition to using https, when communicating with web services ensure you perform end point checking in both the app and the web service to confirm that the server/device you are connecting with is valid.
- Always check that any settings your app is packaged with have a checksum to ensure that the values cannot be changed once shipped to the device.
- Do not allow the app to run on jail broken or rooted devices.
- Have a method for disabling the app if the app detects that it has been compromised.
- Ensure that all use of encryption complies to export regulations and any laws relevant to the region/s the app is being used in.







Mobile Gaming

The Mobile Gaming Economy

The games market on mobile continues to be a major driver of growth and revenues for developers and in 2016 generated the equivalent of global box offices sales for the year¹. This has now become so definitive that mobile finally became the largest sector of the video games market in 2016 and is expected to represent 42% of total global revenues. Q1 2017 revenues in mobile gaming grew 51% year-over-year to about \$11.9 billion².

In terms of content and brands, the market is increasingly mature; with games like Clash of Clans, Candy Crush Saga and Game of War continuing to dominate the top 10 grossing chart. However, in 2016 we did see much more variety than in previous years with new titles like Clash Royale and Pokemon Go (both from established players) have introduced new mechanics. There have also been a rise in the number of second (and third) generation teams coming into the market with great success such as Futureplay and Next Games. Despite this, the challenges of becoming successful in mobile games continue to increase as user acquisition becomes more costly.

2016 was also a year of experimentation for VR with 6.8m

¹ blogs.unity3d.com/2017/02/01/cant-stop-wont-stop-the-2016-mobile-games-market-report

² sensortower.com/blog/top-mobile-games-q1-2017

devices sold and \$1.8b revenues.³ Despite this success a lot of VR/AR developers remain cautious in the short term⁴

Whilst Apple users remain the highest revenue game players in North America; Android represents a significantly larger audience. There are 74.5M Android only users and 48.1M iOS Players. 31% of the market use a combination of devices according to EEDAR⁵. You can find the latest details on the specification of devices being used for games from Unity's Hardware Stats⁶

Making games work on multiple platforms has become easier to do. Looking at the top 1000 free mobile games 41% are made by natively or using in-house tools; 34% use Unity; 18% use Cocos2D; 2.6% use Corona, 2.2% use Unreal; 1.2% use Marmalade and 0.7% use other tools such as GameMaker, V-Play, etc. Each engine offers different advantages and perspectives to enable developers of different skill types to quickly realize their ideas and prepare them for release. See the cross-platform chapter of this guide for more information on the available frameworks.

The time when a random indie-developer can strike it rich with crazy numbers of users is essentially over. Instead, the mobile games market has become a sophisticated space with many different facets and challenges and most games will still fail. Before you start creating your game you need to pay attention to understanding the nature of the market and

³ blogs.unity3d.com/2017/02/01/cant-stop-wont-stop-the-2016-mobile-games-market-report

⁴ www.recode.net/2015/7/14/11614690/at-investor-event-vr-startups-brace-for-slow-growth

⁵ www.eedar.com/sites/default/files/EEDAR%20-Mobile%20Report%202016%20-%20Whitepaper.pdf

⁶ hwstats.unity3d.com/mobile/

audience. An essential part of this is that it has become an enormously competitive market, with vast numbers of small teams producing huge volumes of content and spending millions on development and advertising to retain their positions.

Making The Right Game

Creating delightful experiences for your target audience requires just as much creativity as ever, perhaps more. Different developers have different approaches. For some it starts with an emotion (how do you want the player to feel); for others it is about taking a game they love and realizing it in their own unique style; there are also those who start with a mechanic or an art style and the gameplay evolves from there. Making games based on existing titles sounds easy, but in practice rarely succeeds unless you bring something special that has not been done before. Search for sideways scrolling platform games using pixel art and you will find out just how saturated that market is. Doing something brand new comes with its own risk too as players need something of the familiar to help them understand and relate to your innovations. Scott Rogers in his book “Level Up” described this as the “Triangle of Weirdness”⁷. He claimed that games consist of a world, activities and characters. We can change any of these for new ideas, but we cannot change all three without risking losing the audience.

For me the kind of fun we are looking for in games is something which happens when the player is able to suspend their disbelief and engage in an experience which is free of real-world consequence. We become totally absorbed in the mechanics and narrative of the experience. Curiously, challenge and frustration are as much the motivations to play as they

⁷ mrbosssdesign.blogspot.co.uk/2008/09/triangle-of-weirdness.html

are potential causes to leave the experience. If we achieve a balance between these states of mind we attain a joyful state that all game designers know about, 'Csikszentmihalyi's Flow⁸.

We have to pay attention to what makes a game fun specifically on a mobile device, is different from what we do other platforms. Generally mobile games tend to be simple and accessible with enough depth and a sense of purpose and progression to retain player attention. But it is not that simple. The phone is our most personal device and that affects the way we play. There is also already so much substitute; games have to stand out and emotionally connect in order for players and app stores to select them. Part of the game design process is to create enough anticipation to encourage the player to download (even if free) and communicate the aspiration that spending money will make the game even better.

The type of game you make matters and for those unfamiliar with the principles, some are developed from emergent mechanics, with building blocks which combine to create surprising or strategic outcomes, like Chess or Clash of Clans. Then there are those games built using a series of progressive decision points each resolved with their own steps chained together to make a story such as FTL or Monkey Island. We can even build games which incorporate player creativity such as Terraria or Minecraft, or simple abstract puzzles like Threes or SuperHexagon. Whichever path we take, balance is at the heart of our thinking as a designer. We have to decide how much the game will be affected by skill and how much by luck, the extent to which the game follows a fixed narrative or is player led and of course the complexity of the internal systems, whether that is about character development or a resource

⁸ scienceandvalues.wordpress.com/2010/02/26/csikszentmihalyis-flow-pleasure-and-creativity

economy. With Free-To-Play we also have to consider the impact of money spent on the game experience.

What ever the kind of game is vital that the designer focuses on what matters to the player. The most important questions we should ask is why the player should care. We need them to not just install, but to make our game their distraction of choice. Mobile play may at times starts out as distraction, but in the end we often spend more time playing on our phones than our consoles. With some much competition we have to ask why would they play your game? We have to be able to answer that question honestly.

Engaging the Mobile Player

When we develop mobile games we are creating an experience to entertain players on a specific kind of device. Tablets and phones fulfill different needs and require an attention to detail on the specific and different mode of use they fulfill. The phone is generally about 'the next minute', it is something we get out when we expect something to happen or need something to occupy ourselves. How do players use their tablet devices? For me I think it is about a longer period of rest or relaxation. What does that mean for the game we want to make?

Part of the appeal of any game is tied into its distinctive vision, visual style, compelling gameplay narrative and how the experience is designed to affect the emotions of the players. This has to be appropriate to the nature of how the game is consumed and in mobile we have to understand the restrictions inherent to these devices. The limited screen size, touch screen controls, accelerometers, battery life, the ability to be interrupted, the ease players have to get out and put away the device, the limited speaker quality, the high quality headphone output, etc all affect the way players interact with the device.

Mobile phones are (mostly) connected to the internet and are the most pervasive of devices as we always carry them.

To show you what I mean, think about the way we implement controls. Touch screens allow a huge range of movement on a 2D plane, but after a short while our skin will get hot and lose capacitance which means the controls will become less reliable. If we simply try to duplicate a twin stick control system (like too many games have) we will soon find problems with the playing experience. Arguably, this is one of the many reasons why first person shooters have not been quite as popular on mobile. Instead we should design game mechanics with controls specifically to make the touch process feel good or which recognize the limits of the methods available to us and make that part of the experience. MiniGore is a great example for me where the difficulty of the game is actively enhanced by the twinstick mechanics becoming harder to control over time. On the other hand, Hayday from Supercell demonstrated how touch could be utterly delightful. The touch motion used to collect your crops is so pleasant that it elevates this game far above other farming games on any platform.

Simplicity is an important factor for any design and is supremely difficult to achieve, because even here we have to balance accessibility with the ability to keep people playing over time. This is why it is useful to think about the game mechanic as separate from the context - the part of the game which gives the player a reason to repeat the mechanic. Mechanics are the bare bones of any game. The core loops of actions with starting conditions, challenge, resolutions and reward. These are the actual things players do in the game. The context is the reason to do that again. This can be a narrative, level structure or even simply a way of gating access to content by player performance. But it has to deliver a sense of purpose and progression. More than that the context needs

to call for the player's attention after their session has ended, sitting in the back of our mind as 'unfinished business' enticing us to return later for another session.

Games like CSR and Candy Crush introduced this method of game design to the mobile market. Finding ways to chain together a series of grinding mechanics to sustain playability over thousands of sessions whilst always giving a sense that the goals are achievable is magical. It builds long term engagement and keeps players involved with your game ever longer, provided the experience is sufficiently meaningful.

Keeping players playing longer has a direct impact in their willingness to spend money in the game. In a 2014 survey by Unity⁹, the reported spend by paying players who spend less than an hour in a game averages at \$0.66, but for those who spend over 10 hours this rises to \$15.15.

Designing the Player's Journey

The realization that long term engagement matters has a profound impact on the way we look at game design. The idea of a game as a mechanic or story is transformed to the realization that it is not only our hero character who is going on a journey, but our player as well. This journey consists of several stages:

1. Discovery

Players have a particular set of needs and aspirations when they first encounter your game and there is really very little beyond the icon and the first sentence of your app store listing to motivate players to download and play the game the first time. Despite that setting the right expectations is essential. If the game charges upfront, let the player know why they should

⁹ www.gamesindustry.biz/articles/2014-10-14-mobile-spending-driven-by-35-44-year-olds

still buy it, and what they will miss out on if they do not. If the game is free we still have to create expectations, but we also have to show the player why the advertising is worth the hassle or if there are in-app purchases, why those players will feel good about buying them. This is a delicate art.

2. Learning

Once they have taken the choice to install your game we have to make it as easy as possible to engage. Make the icon and name of the game instantly recognizable and ideally a tease, a reason to kick off the app. At this stage we do not want them to make choices about what characters to play or what levels they want – they do not know yet. Do not make them sign up to Facebook or set-up an account before playing – show them what the game is all about. Then knock their socks off! I often talk about “The Bond Opening”, comparing this first ever play of the game to the opening 5 minutes of every Bond movie. It blows us away and at the same time set up everything we need to know about the story, super-secret agents and the world the story takes place in. But it does more than that, it ensures we never want to leave the seat. It is this dual role of showing not telling that sets up the expectations for the rest of the movie which applies to games so well. As this is a game, we do not want to show or tell, we need to ‘Do!’. Players need to learn about games by doing and feeling good about their achievements quickly.

3. Engaging

If we succeed and set up the right expectations to keep them playing we really start the process of building long term engagement. At this point the player understands the challenge and progression and is already returning to continue to play for subsequent sessions. At this stage they should also understand the value of investing further time or money into the game. It

is much easier to sell IAP and leverage the use of Opt-In Ads to players who already get the benefits of the game. Sustaining this over the longer term however is challenging. We need longer term 'achievable' goals as well as events and social engagement if we are to keep players, and we have to do all this without over-complicating the game.

3.1 Potential To SuperEngage

Players are not all created equally when it comes to their desire to spend money in your game. Some players become 'Super-fans' who actively desire more and more items to enhance their playing experience. The game becomes their 'Hobby' and something they may be willing to invest beyond the price of a cup of coffee. This generally happens only after they fully engaged and where the developer is able to focus on delivering value for that player. It is important not to confuse these players with people who are addicted. Addiction is where individuals have a compulsion which overwhelms their otherwise rational behavior. In practice the majority of true fans are rational people who have made your game their principle hobby. Addictive behavior is always detrimental to the individual and we should do everything we can to help anyone with these kinds of issues.

3.2. Re-engagement

Players who have become engaged will still have a lifecycle; but it may become possible to re-engage them when you add a new update, new content or even events inside the game.

4. Churning

The final lifestage we have to acknowledge is "Churning". It is inevitable that in the end players will stop playing our game. We want to delay that as long as possible, but to fail to plan for that is going to cause us more problems.

Analytics and Game Flow

Making a game is a little like designing an experiment; especially in this data rich, connected era of lean development and minimum viable products. We make a hypothesis and test it as simply as possible. We want to know as fast as possible if we are on to something or not, ideally before spending a lot of money on unnecessary development. That means we need analytics to help us understand what is going on at every life stage.

First, we do not have to capture everything. There are some kinds of data which are static, reference information. For example, the specific position in a specific map. As long as the version of the map used at that time is known then X,Y,Z coordinates alone can be used to create a heat map later. We can also infer a lot of data from other events as long as there is some connecting information. For example, we do not need to capture the level that the player is using for that game in every event or even a list of all the players in that session. We can capture that information with a specific 'Start Session' events and use the associated session ID to allow us to identify everything that happened in that specific game session. Most commercial analytic platforms will automatically capture common data sets like date/time, X,Y,Z etc into their event collection process.

We also have to understand is that the data we collect will always be incomplete, for example if the battery dies or the player switches to a phone call – we will probably not get the last upload. This is less of a problem with a server-based game but it is never 100% and the compromise is that the game cannot be played offline; impacting our chance for them to create habits of play.

We have a duty to treat player data very carefully, we need

to make sure that players remain anonymous. We do not want or need to spy on our players but we do need to understand how the game plays across all players without falling foul of data protection (especially related to children). In case of any doubt make sure you get qualified legal advice.

What Events Should We Capture?

When considering which events to track, think of them in terms of the timeline in which the player might encounter them. There will of course be a first-time player experience, but it can also be useful to map out more commonly experience 'Engaged player' session flow. We are not necessarily mapping every button press directly. Instead you should look for moments where there are meaningful choices. There is an approach used by the food industry called HACCP¹⁰.

Some typical events worth tracking might include the following:

- **GameMenuLaunch:** AnonPlayerID; TimeIconLaunched
- **SessionLaunch:** TimeSessionLaunched; AnonPlayerID(s); SessionID; LevelIDSelected; OptionSelected
- **SessionStart:** TimeSessionStarted; AnonPlayerID; SessionID;
- **ObjectiveSet** TimeObjectiveSet; AnonPlayerID; SessionID; ObjectiveID;
- **ObjectiveMet:** TimeObjectiveMet; AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation

¹⁰ develop-online.net/opinions/navigating-the-hazards-of-game-data/0187815: In essence the point is that we are looking for the 'Hazards' in the flow of the player experience such as whether they churn (i.e. leave the game) but also trigger points for more positive action such as paying for an IAP or watching a video ad.

- **TargetHit:** TimeTarget; tHit; AttackerID(AnonPlayerID?); SessionID; TargetID(AnonPlayerID?); Damage, XYZLocation
- **PlayerDeath:** TimePlayerDeath; AnonPlayerID; SessionID; XYZLocation
- **LevelComplete:** AnonPlayerID; SessionID; ObjectiveID; Score; Reward; XYZLocation

From creating events in this way we can infer a huge amount of information. For example, if we want to know the percentage of players who complete a level we can count the number of 'GameMenuLaunch' events with the number of 'LevelComplete'. But we can also get smarter with our analysis. We can look at how many people completed a specific ObjectiveID in a specific 'LevelIDSelected' and compare that to the number of 'LevelComplete' in the subsequent level to find out if skipping objectives in earlier levels have a particular impact on performance later.

Once we have the data we need to be able to use it and continually review its accuracy and usefulness. Commercial platforms will often come with comprehensive reporting tools from basic dashboards with KPIs like D2/D7 Retention, DAU (Daily Active Users), ARPU (Average Revenue Per User) and even ARPDAU (Average Revenue Per Daily Active User). There are two essential tools every developer should look to use. First there are 'Funnels' which when we set up events in order of how a player passes through the game we can see where in the pipeline you have the most problems. The second are heat maps which put that data into your level in a way which lets you understand exactly what's going on inside the gameplay session.

Free vs. Paid

The arguments between free and paid games have become almost tribal amongst game developers asking whether business models have tarnished the nature of game design, even asking questions about the morality of these money focused designs.

Looking in economics terms, it is clear what happens when supply goes up, prices fall. With an effectively infinite supply, the price falls to zero. This is exactly what has happened and why Free2Play is so dominant. But wait, what about the 7% of revenue for iOS on premium games? Successful premium games are the ones which have been able to attract an audience by offering something they perceive to be of greater value than the rest of the games available. Games like Monument Valley or The Room have shown that this is still possible, and they are noticed because of their premium pricing. This has not been at the scale of the revenues of the top performing Free2Play games, despite considerable profiling by app stores.

The easiest way to understand Free2Play is to realize that these games have simply taken the retail side inside the game. That means that the people who know the game and its players best (i.e. the developers) can identify items that players will love that compliment and enhance the experience for all players; and sell them directly to the audience. The movement towards free has not proved an easy path for many game providers and attempts to 'clone' the business models of games like Clash of Clans or Candy Crush have rarely seen even a comparable level of success. This is despite the formula appearing on the surface to be so simple and can quickly become 'not much fun' causing a lot of players to churn. If your player feels that the game is merely an exercise in getting me to open my wallet, just how engaged will I be as a player?

Advertising is playing an every increasing role in games and it is fascinating that unlike other media, ads in games

do not seem to cannibalize the audience. Similar to the way we are changing how we look at in-app purchases, we are seeing a movement looking to find way for ads to add value to the overall playing experience, rather than just blocking the players progress. It is also fascinating that unlike other media, ads in games do not seem to cannibalize the audience. The use of banner ads in a game, which can be clicked accidentally and take up valuable screen space, is largely being replaced by interstitials or opt-in video but remains on some highly successful one-handed quick play games like Sixes from Gram. We are also starting to see the introduction of brand based ads which, perhaps surprisingly, seem to add a level of credibility to the whole experience; apparently seeing an ad from Coke or Audi makes players gene more engaged.

What ever business model you try, Free/Paid/DLC/etc the same basic design rules apply.

Seven Rules of Monetisation Design for Games

Rule 1: Utility

Everything starts with utility, an economics term in this case used to express the 'expectation of value' for the player. In Free2Play players are usually not buying the 'gems' or whatever currency you might be using because they are so shiny. Buyers are driven by the expectation of what gameplay these items will unlock. The same principle applies even if you are charging for access to the game. It is not the physical delivery or the download size that players value but the anticipation of the game.

While in premium games we earn money before the users are actually experiencing it (apart from potential additional DownLoadable Content (DLC)), for free2play games the lifecycle of the player after the initial download plays a critical

part in monetization. With this in mind it is often helpful to break your game into three sections:

- **Mechanic:** The core element of play - usually a loop usually with some kind of challenge; resolution and reward
- **Context:** A mechanic used to drive a sense of purpose and progression including narrative, unlock tree, boss modes, etc.
- **Metagame:** The elements that are not about the pixels including collaboration, clan systems, mode of use of the devices, etc.

Each of these stages may include an opportunity for the use of an in-game advert or in-app-purchase (IAP)

The levers we use to add value will usually be different in the different stages of the experience. Consumables which work as a part of the mechanic, might be inappropriate for the context or metagame. Splitting the experience down into sections allows us to understand where the value really is and keeps the focus on the benefits for the players. We can also use this mindset to rethink how we use ads especially opt-in video ads. They can now become a method to extend players behavior, e.g. watch an ad to get a free power-up you have not tried before. Once the player tries it hopefully they will be more inclined to spend a little money on a bundle of 10. Each purchase decision has a different impact depending on where in the game it occurs as well as the players lifecycle. The use of analytics and even machine learning can transform our success as long as we set-up meaningful tests.

To be more specific about what monetization elements we want to use in our game design it is worth looking at it through the lens of our game design toolkit.

This start by looking at the types of goods in games:

- **Sustenance:** What are the elements we require to continue playing?
- **Shortcuts:** What factors increase our chance of success or reduce the impact of failure?
- **Socialisation:** How can players express themselves and their progress in the game?
- **Strategy:** Can we introduce new playing options (without breaking the game)?

Those goods come in different forms as well:

- **Consumable:** A one-time use item.
- **Capacity:** Something which limits growth/play
- **Permanent:** A permanent upgrade or unlock item
- **Generators:** An increase in the supply of a consumable

When we combine the type and form this creates a powerful way to assess your game's monetization model.

GOOD/TYPE	Consumable	Capacity	Generator	Aspiration
Sustenance	Fuel	Fuel Tank	Gas Station	Better Car
Shortcut	Strength Potion	More Strength	Alchemist	Improved Recipe
Social	Heart Gift	Supporters	Wanderers	New Outfit
Strategy	Door Key	New Tool	Booster Pack	Red Katana

Let me illustrate this with a concrete example: We have a driving game where we use fuel as an energy mechanic; but we may decide that you cannot buy fuel (we do not want it to feel like a tax) but we do allow players to watch an opt-in ad to refuel once per 24 minutes. Players can also upgrade the size of their fuel tank which means they get more plays before running out. Then we can have a gas station which means they can replace their fuel faster than before. Finally we have the better car which we desire because it adds advantages; but it also has consequences e.g. using more fuel making this a playful choice - not just an upgrade.

Rule 2: Anticipation

An essential element of monetization design is how we communicate the Utility we are creating to the player. There are generally four forces preventing a player making a purchase: Uncertainty of outcome, Social issues, Opportunity costs and External needs.

A hard lesson from this is that we cannot make people pay; and although it might be possible to manipulate people in the short term that is not sustainable, counterproductive and damages trust for everyone. Instead, we need to create the conditions where people can give themselves permission to play. This means we need to create the following four factors: Expectation of delight, Social capital, Call to action and Abnegation of other priorities.

Rule 3: Scarcity

Just like in the real-life economy, scarcity is also a vital aspect of the economy of your game. However, when creating scarcity, do not forget about rule1: Utility. Any use of scarcity has to be authentic and focus on the enjoyment of the game. Adding opt-in video ads or IAP must be an extension of our overall design and we must consider how spending affects the balance of the game to avoid creating a 'Pay To Win' model (better described as a broken game).

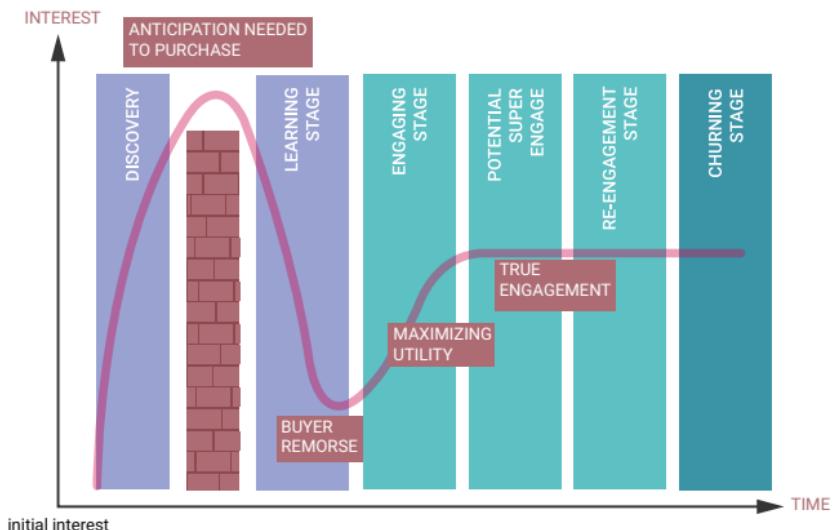
Vital to scarcity is to look at the opportunities in play and through pay to earn those scarce resources and currency and to make sure that you have designed appropriate 'resource sinks' which ensure are not directly about 'winning'. We need soft variables of success to deliver the maximum Utility to the player.

Look at the game Rock-Paper-Scissors. Imagine we have two upgrades 'lizard' and 'spock'. Because these new shapes can both win and lose against the others the actual chance of random success is unchanged. There is no technical advantage. However, there is a psychological advantage as well as increasing complexity limiting peoples ability to second guess their opponent.



Rule 4: Timing

Players needs are not static. This is especially important in free2play games but also affects players willingness to make DLC purchases in premium games. As developers we have to think about the player lifecycle and how that impacts players willingness and interest in making purchases. It is also important that the game feels alive through community engagement, events and regular predictable updates.



- **DISCOVERY:** Will the player Install then play or not
- **LEARNING:** The player not only has to learn the game; but if it will become part of their routine
- **ENGAGING:** The player is committed to the game and exploring if they are willing to spend
- **POTENTIAL SUPER-ENGAGE:** Is this a player willing to invest more in this game than average
- **REENGAGEMENT:** Player is at risk of churning; can we bring them back
- **CHURN:** Player has moved on to the next game

If we understand the status of the player we can be in a better position to understand what they value and how we can make offers which extend their lifecycle and their life-time value.

Rule 5: Repetition

Repetitive actions can become intrinsically rewarding and this can also help build positive habits as well as building trust with your players that they can obtain the Utility they anticipate from your game. Highly repeatable game mechanics are essential to monetization success, especially on mobile where we can play repeatedly throughout the day.

This is reflected in the rapid increase in the willingness of player to spend more the longer they have engaged with a game.

In 2014 Unity Ads did a survey with 3000 online participants which showed a clear correlation between longer play time and increase revenue. This was not just linear either.

HOURS PLAYED	AVERAGE SPEND
10+h	\$15.15
5-10h	\$4.90
1-4h	\$2.55
<1h	\$0.66
ALL	\$4.58

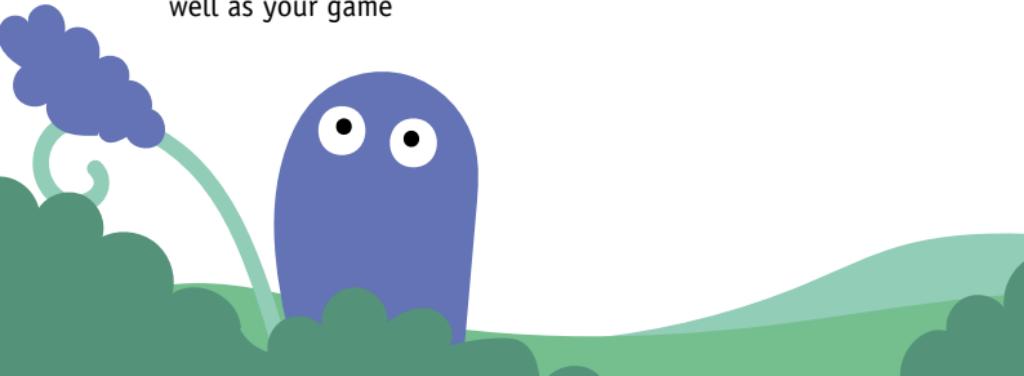
However, it is also important to avoid player fatigue. This means giving players natural breaks where they can switch between 'Frustration' to 'Relief'; from 'Intensity' to 'Relaxation' and feel that they can break away from the game with a reason to return later. Otherwise they may burn out. This can play an important part in the placement and use of different ad formats. Blocking ads like full screen interstitials can be frustrating so work best on screens where the player can rest. On the other hand opt-in ads are less frustrating and by their nature offer the player 'Utility' in return for engaging but this creates another design problem. We have to think about what benefit makes the most sense for the player to watch that ad - it is almost an extension of the thinking we need to do for IAP.

One of the keys to sustaining interest over long periods of play lies in the use of predictable uncertainty. Take 22 Cans game, The Trail. In this game, items we may need for crafting appear along the trail at apparently random intervals, we can only carry so much in our bag and each time we take the walk we have to decide if we will pick up those items we find. If we do we risk not having enough space for a specific item we may need to progress. It is predictable we will get those items; but uncertain when so the player is entertained by the time they spend looking whether they appear or not. The act of looking does not become boring even though it is repetitive because it retains a sense of unfulfilled anticipation.

Rule 6: Evidence

We have talked about the importance of data and at the risk of repetition here is a set of practical steps you can go through to start the process.

1. **Define the objective** Ask why you want this data and set clear objectives on what you expect the data to tell you
2. **Identify what you can measure** Identify what data points will help us answer the questions laid out in our objectives
3. **Identify player decision points** Identify the trigger actions in the game which indicate player decisions
4. **Define common events** What do not we have to specifically collect and which data point allow us to compare events across the game e.g. AnonPlayerID; SessionID/etc
5. **Identify reference data** I.e. What do not we have to specifically collect because these data points do not change during a player session (i.e. we need not repeatedly capture them)
6. **Select your analytics platform** Do you make your own or use an external provider such as Unity Analytics
7. **Segment your data** Create custom cohorts so you can compare different parts of your audience or different builds of your game, etc.
8. **Create funnels, heat maps, define KPIs** as required to track your game's performance
9. **Continue to iterate and test** your reporting process as well as your game



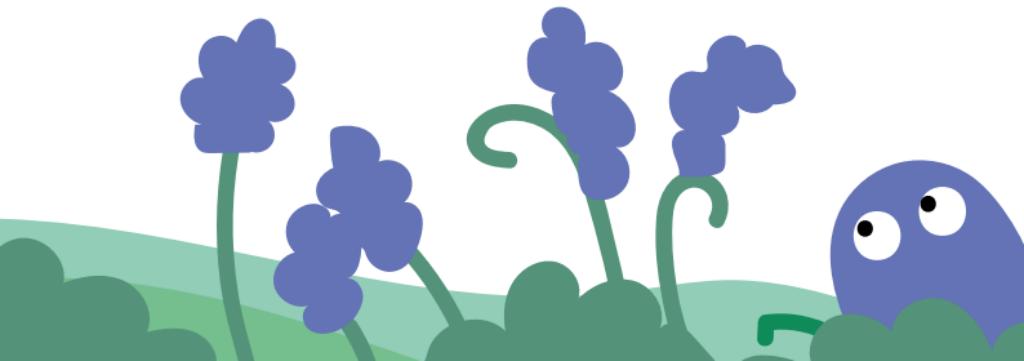
Rule 7: Scale

The most troublesome rule; and perhaps the one rule to rule them all. Scale matters. We need an enormous volume of players watching a significant number of ads per day in order to generate measurable revenues just on ads. Typically 30%-50% of players choosing to watch an opt-in ad. IAP are much rarer, but their value is more tangible. Typical games see 2-3% of players spending on IAP of the whole base which downloaded the game. However, this statistic really misses the point. The percentage of people who spend the most money will be those who remain playing your game longest. At 30 days we expect to see less than 10% of the original downloads. If the average spend on any item in the games is as little as \$1 this also depends on scale.

Scale is not just about the number of players. Puzzles and Dragons and Clash Royale do not have the same size of audience as Candy Crush but they do see enormous revenues.

The key to scale is getting more players, doing more things, more often for longer.

This is an industry led by acquisition, retention and monetization; but we can be smarter if we apply utility, anticipation, scarcity, timing, repetition, evidence and scale.



Getting Discovered

If you have followed these guidelines then you will have already put your game design into the best form that suits your audience and that itself will (hopefully) give you a fighting chance. However, that alone is not enough. We have to use every possible communication route we can and that usually requires investment. It is still possible to succeed without spending money on advertising, but you have to be the winner of a global lottery ticket. This applies on mobile games as on any other kind of mobile app as well. Some hints how to market your software can be found in the monetization chapter. Additionally, here are strategies which you might want to think about especially for games.

Press

Getting noticed by the press can help, particularly if you participate in games awards such as Pocket Gamer's Big Indie Pitch¹¹ or the Indie Awards at Casual Connect¹². If you can get the attention of YouTubers that may also help. However, the benefit of these activities is rarely measured in downloads. Instead they give you more credibility especially with the App Stores.

Advertising

Spending money on advertising can help, but it is important to realize that you are competing with a lot of people and some big players who are seeking large audiences. It is important to remember what you are trying to achieve when creating an advert. There are two motivations, building awareness

¹¹ www.pocketgamer.biz/events

¹² indieprize.org

and direct action (i.e. downloading the game). In games we are able to put adverts in other games and apps on the same device we want the players to experience the game. There is nothing getting in the way between the advert and the app store. One click and you can buy/download the game. That is an amazing thing, no other media has that kind of frictionless experience.

Another peculiarity to be aware of is that the larger the reach (range of players) you are looking for, the more expensive each of the installs. This is because buying space on an advertising network is based on a bidding process and the results will be calculated on the basis of Cost Per Install, Cost Per Mille (i.e. per thousand) or a blend of the two known as eCPM (effective CPM) as well as ad networks like *Chartboost.com* or *AppFlood.com* which offer cross promotion.

Video based advertising is growing and allows the player to instantly understand the nature of the game being shown. This is often combined, such as with Unity ads¹³, *Vungle.com* and *AdColony.com* with incentives inside the game – such as free currency. This kind of incentive is different from external incentives such as offered by providers like Tapjoy and importantly is not allowed on Apple's network.

In Game Events

Regular events and outreach to the community allow us to sustain and to grow our audience. Building on genuine social experiences, such as the recording of gameplay videos and sharing of community data (high scores etc) players can help reach out to their friends and other potential players via Facebook, Twitter, Everyplay and YouTube.

¹³ unityads.unity3d.com

Influencer Marketing

2015 marked a point where the role of social engagement with games hit critical levels and YouTube personalities like Pewdiepie and Yogscast are having significant impact on the take up of games, including mobile. Teams like Hipster Whale and Serious.ly have made engagement with these important celebrity influencers to help propel their games to the top of the app store. However, like any medium this is becoming increasingly commercial but it remains important to consider the audience and why a YouTuber engaging with your game will be entertaining for potential players. This is becoming increasingly professionalized with a rising number of Influencer Marketing agencies representing the Youtuber and Twitch communities. However, it is important to recognize that this comes at a risk if the Youtuber damages their own reputation such as in Feb 2017 when Disney and YouTube ceased working with Pewdiepie after a scandal¹⁴

eSports

On the eSports side it has been interesting to see that as well as the huge audiences on online channels such as Twitch that even television channels like ESPN have started to take mass audiences watching games as seriously as other more physical sports. The level of talent and professionalism in the eSports market is now significant and game developers are starting to consider how this will impact game design. However it is still the case that there are very few mobile games which can legitimately claim to have gained a strong enough following.

¹⁴ <https://www.nytimes.com/2017/02/16/magazine/youtubes-monster-pewdiepie-and-his-populist-revolt.html>

In Summary

In the end despite all the differences in the details, mobile is like any other platform. We have to acquire, retain and monetize our audience. That only happens if we entertain players in the way that works for their devices. Devices which are perhaps the most social and most pervasive devices in human history. Mobile gaming is thriving despite the hurdles and the lessons learned will affect every aspect of game development.



Dev Guy



The Thinker



The Internet of Things (IoT)

The Internet of Things (IoT) is the inter-networking of devices, known as 'things'; whether it be sensors, actuators or gateway devices that enable such devices to collect and exchange data to the Internet. When IoT is augmented with multiple things, typically hundreds or thousands, the technology becomes an instance of the more general class of cyber-physical systems - which encompasses technologies such as smart grids, smart homes, manufacturing, intelligent transportation, smart cities and traditional machine to machine deployments¹. Each thing is uniquely identifiable through its own embedded computing system, and is able to interoperate within the existing internet infrastructure. Experts estimate that the IoT will consist of about 50 billion things by 2020², while several other big numbers are claimed from various industry players. This is made possible by low-cost hardware, affordable connectivity and emerging de-facto standards for how devices can communicate.

Just over a decade ago two distinct technology industries, the mobile industry and M2M³, co-existed while with limited interoperability between each other mainly due to their consumer or industrial focus - each boasting trillion dollar turnovers yet with little shared knowledge and necessary insights on how to ultimately work together. With the rise of IoT focusing more on consumer oriented devices there is a

¹ for examples of IoT use cases see postscapes.com/internet-of-things-examples

² www.eetimes.com/document.asp?doc_id=1321229

³ machine-to-machine, embedded or industrial internet

requirement to provide end-to-end services for customers. If major players had early on identified this need there would be an abundance of tools and services available to bridge the two industries, however this has not been the case and only recently has there been mass panic to provide tools and services to integrate IoT with modern mobile devices. Infrastructure and data collection is not enough, without services or apps nobody will pay a dime in the end.

One of the drivers we see to help strengthen the bridge between these industries, is that several traditionally closed hardware systems are opening up their services via APIs, moving code bases to Github⁴ and other open code repositories to simplify developer access to industrial systems and services. It is very promising having big industrial corporations like Ford, GE, Bosch, Cisco and Siemens migrating towards more open standards, and exposing their industry focused product and service APIs to developers accustomed to IP-based technologies. For those in the mobile industry, hooking up phones to connected things is becoming more common place, and many hardware devices are starting to rely on supporting mobile apps to obtain their full functionality range. Moving out of the self-contained model allows them to release products earlier and the ability to add features over time by upgrading firmware, in tandem with updates to their associated mobile apps.

The concept of releasing in this green banana fashion has proven very useful from a competitive point-of-view and thanks to dynamic software architecture, over-the-air updating, not to mention getting users into the habit to use things that hardly work when first purchased - as it has generated a dynamic and thriving relationship between manufacturers and

⁴ github.com

consumers. One of the side effects of this release tactic has caused a lapse within the field of security⁵ and has become a concern not only for consumers but also for manufacturers as their things are being repurposed, such as the Mirai botnet⁶ in November 2016 that was responsible for major Internet disruptions via Distributed Denial of Service (DDoS) attacks.

Roles for Mobile in IoT

There is already a healthy amount of IoT apps available and you can see three distinct roles for a mobile phone or tablet within the Internet of Things: It can act as a dashboard for displaying data, a remote control for IoT devices and systems or as a gateway for things which do not have their own access to the Internet and need to tether on the phone's cellular or wifi connection (e.g. a wearable sports tracker using bluetooth connectivity).

You might want to argue that there are no specific IoT apps per se - they are just apps, right? The follow-up would then be that a mobile device is primarily a client for a cloud-side database, creating a viewing engine of a database of sensory information. Yet there actually are several cases where features of a smartphone come in handy; its off-line capabilities, such as the ability to read RFID tags (Android), access to local storage, positioning, augmented reality, means for authentication and conditional access in the field, audio and video capture - all are of interest when it comes to creating mobile services for enterprise and industry. Some of these can be instantiated even using a browser, whereas extensions for accessing

⁵ see the IoT Security Statistics: www.gsma.com/iot/iot-security-industry-statistics/

⁶ [en.wikipedia.org/wiki/Mirai_\(malware\)](https://en.wikipedia.org/wiki/Mirai_(malware))

bluetooth Web Bluetooth SIG⁷ is limited to a few browsers and to a sub-set of the native bluetooth stack capabilities.

On the manufacturing floor there are hundreds of devices to communicate with, and we do not expect each and every of them to have a mobile app on their own any time soon. Therefore, systems of systems, where groups or families of products can be controlled from a base line of apps, is a necessity as well as alternatives to the consumer app store model as it does not rhyme well with the industry's needs and wants.

3rd party tools for faster development

While many developers still use native mobile developer toolkits as their weapon of choice, third party tools purposely written for creating IoT software have spawned and proven viable for a number of reasons; they often use web technologies to create user interface views and they come with prewritten components for cloud services, hardware libraries or connectivity. Each market segment within IoT - from wearables, real estate automation, medical applications to surveillance and fleet management - has their own special needs and challenges: offline usage, large datasets, need for strong encryption, real-time lag free interaction, high demands on bandwidth and/or accessibility on a global scale. No single tool or library covers all use cases.

One of the more popular base technologies for hybrid solutions is Apache Cordova⁸, sibling to its commercial incarnation Phonegap⁹. By its open plug-in architecture, web and native components (purposely built for each target operating system)

⁷ www.w3.org/community/web-bluetooth

⁸ cordova.apache.org

⁹ phonegap.com



can be mixed and matched freely, with the end goal to create either an application for publishing on the relevant distribution channels. There are also tools and libraries available outside of Cordova, while few are focused solely on IoT application development, they may well have functionality useful to industrial service development.

Several commercial hybrid SDKs use Cordova as one cornerstone, others rely on C# or C++. Here is a list of relevant frameworks:

- **Evothings Studio**, a rapid development suite for native/hybrid apps using JavaScript: evothings.com/download
- **IBM Mobile First**, for enterprise and industrial tools (mainly for iOS): ibm.com/mobilefirst
- **Intel SDK**, a generic toolkit for enterprise and industry: software.intel.com/en-us/intel-xdk
- **Ionic**, a set of front-end development tools, addressing multi-platform UI: ionicframework.com
- **RAD Studio**, tools for Pascal and C++ builders: embarcadero.com
- **Waygum**, an application platform for mobile IoT, with mobile front-ends: waygum.com
- **Xamarin**, framework for C# development, supports many native features: xamarin.com

Communications and Protocols

One of the standing issues in development for the Internet of Things (IoT) is the occurrence of exotic communication protocols for a mobile programmer, with names like XMPP¹⁰, MQTT¹¹ and CoAP¹². Smartphone apps need ways to communicate using some of these protocols to interact with devices running as IoT. Thankfully some implementations are available such as the Eclipse Paho project which includes an Android client¹³. MQTT can run both over raw TCP/TLS sockets and Websockets, which allows also this format to run inside a web browser.

To be able to interact over low-level TCP and UDP based networking, transport security et cetera, technologies like Chromium sockets (i.e. Berkeley sockets nicely wrapped for JavaScript developers) available as plug-in technology for Apache Cordova needed to be introduced. Establishing mobile plug-in support also for TLS (Transport Layer Security) is also a step forward towards end-to-end strong security from sensor to mobile device safeguards IoT services from many of the uncertainties that face web services and APIs exposed to the public Internet. Coming from the old embedded world can be scary for an organization who has been enjoying bespoke networking not sharing cables with anyone, and then in the flash of an instance after connecting a HTTP gateway to the old M2M system, get all the possibilities, horrors and problems that the internet residents have seen for more than 20 years.

¹⁰ xmpp.org

¹¹ mqtt.org

¹² tools.ietf.org/html/rfc7252

¹³ eclipse.org/paho/clients/android

Securing the message payload, rather than securing the channel allows even public networks, even WiFi hotspots to be used. Otherwise, end-to-end security will be hard to achieve as they are often many moving parts in IoT projects.

IoT apps proved over time to be more than simple web-based database clients, and will be doing much more than visualizing server-side generated data views. A second wave of apps is coming our way in where IoT mobile services for phones converse directly over short-range radio, using low-level IP-based protocols for sensor data and telemetry messaging with a minimum of overhead. The prevailing standard here is Bluetooth Smart, which lately has acquired an increased sense of security, longer range as well as meshing capabilities. Two of the most interesting application thereof are both chipsets allowing the bluetooth radio to be in announcement (broadcasting) mode and connect services concurrently, which takes Bluetooth as a concept beyond the somewhat limiting one-phone-one-device concept. The other being the open Eddystone bluetooth beacon format, assisting in having users consume contextually relevant mobile services on location without access to any centralized hub or third party servers. It allows end-users to discover and evaluate services limited to a geographical area, a service type or system role.

Learn More

- **Introductory article** comparing IoT protocols: electronicdesign.com/embedded/understanding-protocols-behind-internet-things
- **A Cisco view on IoT Application Protocols:** blogs.cisco.com/ioe/beyond-mqtt-a-cisco-view-on-iot-protocols
- **Scaling the Internet of Things** Video by Yodit Stanton recorded at ODI Summit 2015: youtube.com/watch?v=MP2HLCNPgJ0
- **Eclipse IoT protocols:** iot.eclipse.org/protocols.html
- **Realtime data with MQTT**, video covering MQTT and IoT topics: youtube.com/watch?v=gj8mcn9oWgE
- **IoT Demonstration using WebSockets:** developer.mbed.org/cookbook/Internet-of-Things-Demonstration
- **Vision Mobile report on the IoT landscape:** visionmobile.com/product/commerce-of-things-2015



To end this chapter in a progressive manner, here are some good starting points, representing some of the stakeholder of the industry, software, hardware, aggregators and service providers.

- **Estimotes Blog:** *estimote.com*, Manufacturer of iBeacons and mobile SDK
- **Evothings Studio examples and templates:** *evothings.com/developer*
- **RIoT Secure Blog:** *riotsecure.se/blog*, Developer focused awareness of security within IoT
- **IBMs IOT Foundation:** *internetofthings.ibmcloud.com*, IoT cloudware & apps
- **IFTTT:** *ifttt.com*, If-This-Then-That - a cloud company connecting events over the internet
- **Intel IoT, and the Intel XDK:** *software.intel.com/en-us/iot*, Devtools for micro-controllers and mobile apps
- **Phant by Sparkfun:** *data.sparkfun.com*, Maker of IoT hardware and accessories, here linking to their nifty IoT server-side backend, perfect for app makers who want to own their own data.
- **Particle.io:** *particle.io*, Maker of IoT hardware and accessories, with own cloud for data collection and analytics





Artificially Intelligent Apps

Today artificial intelligence (AI) is all the rage – does it make sense to make your app smart and how could you do that? Read on for some perspective on AI in the mobile apps space.

AI Achievements and AI Hype

We have made big advancements in specific aspects of intelligence, sometimes exceeding average human capabilities:

AI tops the best players of the world in some games such as Reversi, Chess, Go, certain Poker variants or even Ms. Pac-Man¹. When limited to one surgery procedure an autonomous robot even outperforms human surgeons². Self-driving cars will not only lead to fewer accidents but will also axe jobs in logistics. It even might tempt us to move farther out of the cities as we can be productive during the ride. Additionally, AI algorithms have learned to lie when negotiating³ or when playing poker⁴. And AI even can create new AI algorithms⁵, and deblur images⁶.

The achievements of AI are surely impressive - then again, AI has a history of over promising which led to several cycles of investment followed by an abandonment of research and

¹ wired.com/story/microsoft-ai-ms-pac-man

² techcrunch.com/2016/05/04/robot-surgeon-outperforms-human-colleagues-doing-same-procedure

³ theresister.co.uk/2017/06/15/facebook_to_teach_chatbots_negotiation

⁴ steemit.com/ai/@sykochica/ai-learns-to-lie-and-beats-humans-at-poker

⁵ technologyreview.com/s/603381/ai-software-learns-to-make-ai-software

⁶ arstechnica.co.uk/information-technology/2017/02/google-brain-super-resolution-zoom-enhance

commercial activities in that area. Vernor Vinge, for example, predicted in 1993 that we will reach the point of singularity⁷ at which AI surpasses human intelligence within the next thirty years (until 2023). This seems highly unlikely in 2017. Creating general artificial intelligence seems to be difficult, maybe because we still do not understand the concept of intelligence⁸ fully. Also, we just do not know what will happen if AI reaches human intelligence – will it then be simple to scale beyond our intelligence?

Artificial intelligence may be overhyped, but we might very well arrive at the tipping point soon, at which AI suddenly influences our lives to a large degree.

AI Terms and Technology

Let us start with a very short primer on AI terminology and state of the art. Hidden underneath the umbrella term artificial intelligence lies a whole world of different set of technologies. We are still far away from a general or strong intelligence that combines all different aspects of intelligence. And no, it is not only machine learning, even though that aspect is getting most of the press hype nowadays.

Arguably the most important aspects of AI include the following:

- Machine learning allows computers to improve automatically with experience. Deep neural networks are often used for this task.
- Pattern recognition allows computers to recognize input such as handwriting, faces or objects.

⁷ edoras.sdsu.edu/~vinge/misc/singularity.html

⁸ en.wikipedia.org/wiki/Intelligence

- Pattern prediction allows to predict the future positions of previously recognized objects from a series of inputs. According to Jeff Hawkins: "Prediction is not just one of the things your brain does. It is the primary function of the neocortex, and the foundation of intelligence."⁹
- Artificial knowledge is often formulated in hierarchical ontologies. This not only requires a lot of manual work, it also misses out unconscious feelings about or subsymbolic meanings of a certain situation that us humans know about. So, there is a lot of work to automate this process and to understand subsymbolic meanings better.

To learn more about foundation of AI just head on to [Wikipedia¹⁰](#) for a detailed overview.

⁹ Jeff Hawkins: On Intelligence (Owlett Paperbacks 2005), p89

¹⁰ en.wikipedia.org/wiki/Artificial_intelligence



Layers of AI Tools

The world of AI is somewhat overwhelming at first, so you should know that there are different layers that you can use, depending on your expertise, your problem and the data you have at hand.

Layer	Description	Examples
Hardware	Hardware optimized for processing AI algorithms	IBM True North ¹¹ , Google Tensor Processing Unit ¹² , Intel Nervana ¹³
Algorithm	Create your own problem-specific algorithms	Neural Networks, Hidden Markov Model, Linear Regression, Naïve Bayes

¹¹ research.ibm.com/articles/brain-chip.shtml

¹² en.wikipedia.org/wiki/Tensor_processing_unit

¹³ nervanasys.com



Layer	Description	Examples
Model	Use an existing algorithm but train it with your problem-specific data	Tensor Flow ¹⁴ , CNTK ¹⁵ , Theano ¹⁶ , Torch ¹⁷ , Caffe ¹⁸ , Caffe2 ¹⁹ , Amazon Machine Learning ²⁰ , MXNet ²¹ , Cyc ²²
API/Online Service	Use an online API	Microsoft Cognitive Services ²³ , AWS Amazon AI ²⁴ , Google Machine Learning Services ²⁵ , IBM Watson Services ²⁶ , HPE Haven OnDemand ²⁷ , api.ai ²⁸ , Cyc ²⁹

¹⁴ tensorflow.org

¹⁵ microsoft.com/en-us/cognitive-toolkit

¹⁶ deeplearning.net/software/theano

¹⁷ torch.ch

¹⁸ caffe.berkeleyvision.org

¹⁹ caffe2.ai

²⁰ aws.amazon.com/machine-learning

²¹ mxnet.io

²² dev.cyc.com

²³ azure.microsoft.com/en-us/services/cognitive-services

²⁴ aws.amazon.com/amazon-ai/#ai_services

²⁵ cloud.google.com/products/machine-learning

²⁶ cloud.google.com/products/machine-learning

²⁷ havenondemand.com

²⁸ api.ai

²⁹ dev.cyc.com

How AI is Used in Apps

There are a billion use cases for weak AI, but how is AI being used in apps today? Here are some examples:

- Google Allo³⁰ is providing context-dependent, AI generated “smart replies” that take your personal style into account and that includes image recognition.
- The Google Lens³¹ camera app recognizes objects and metadata and provides matching interactions like “save event to calendar” or “buy tickets” when you photograph a concert marquee.
- There are various photo effect apps that provide AI-powered artistic effects. FaceApp³² makes you older, lets you smile or even changes your gender upon demand. Artistic app examples include Prisma³³, DeepArt³⁴, and GoArt³⁵.

³⁰ allo.google.com

³¹ techcrunch.com/2017/05/17/google-lens-will-let-smartphone-cameras-understand-what-they-see-and-take-action

³² faceapp.com

³³ prisma-ai.com

³⁴ deepart.io

³⁵ goart.fotor.com

- Microsoft's Story Remix³⁶ app combines face recognition, object recognition, movement following and 3D technologies for impressing video editing power.
- In healthcare³⁷, companies such as SkinVision allow everyone to diagnose moles for possible skin cancer³⁸.
- Microsoft's Seeing AI³⁹ describes people, objects and texts for visually impaired persons.
- Skype⁴⁰ now offers a real-time voice translation between 8 languages.
- And finally, BeerAI⁴¹ suggests beer brands based on your current preferences.

³⁶ techcrunch.com/2017/05/11/microsofts-windows-story-remix-uses-machine-learning-to-make-your-videos-look-awesome

³⁷ for further examples see en.wikipedia.org/wiki/Artificial_intelligence_in_healthcare

³⁸ skinvision.com

³⁹ microsoft.com/en-us/seeing-ai

⁴⁰ skype.com/en/features/skype-translator

⁴¹ app.beerai.com

AI Tooling

So, what should a mere mortal do when you want to add intelligence to your app? This depends on your know-how, your target platforms and your goals.

Online Services

The easiest initial step is to use one of the many AI services out there. With only little configuration setup you are up and running and can process natural language input, recognize objects in videos or translate audio. A drawback is the data leakage, as the data of your users is processed in services not controlled by you. Another drawback is that it may be more difficult to differentiate from competitors. Last but not least, if your central idea is focused on AI, you do not want to use external services for that. The following table lists popular AI services.

Name	Service
AWS Amazon AI⁴²	Natural language understanding, automatic speech recognition, visual search, image recognition, text-to-speech, machine learning
API.AI⁴³	Natural language processing
Google Machine Learning Services⁴⁴	Machine learning, job search, video analysis, image analysis, speech recognition, text recognition, translation

⁴² <https://aws.amazon.com/amazon-ai>

⁴³ <https://api.ai>

⁴⁴ <https://cloud.google.com/products/machine-learning>

Name	Service
HPE Haven OnDemand ⁴⁵	Audio, video & image analysis, indexing & search, text analysis, format conversion
IBM Watson Services ⁴⁶	Conversation, discovery, document conversion, translation, natural language classification & understanding, personality insights, retrieve & rank, speech-to-text/text-to-speech, tone analysis, visual recognition
Microsoft Cognitive Services ⁴⁷	Vision (face, emotion, video, content, custom), speech (translator, speaker recognition, custom), language (understanding, spell check, text analysis, linguistic analysis), knowledge (recommendation, academic, custom), search (autosuggest, image, video, web, news, custom)
Wit.ai (Facebook) ⁴⁸	Natural language processing

⁴⁵ <https://www.havenondemand.com>

⁴⁶ <https://console.ng.bluemix.net/catalog/?category=watson>

⁴⁷ <https://azure.microsoft.com/en-us/services/cognitive-services>

⁴⁸ <https://wit.ai>

Chatbot Frameworks

To create a chatbot use one of the many chatbot frameworks out there. The frameworks typically also offer you to integrate your chatbot into various channels such as Facebook Messenger, Skype or WeChat. You can discuss various aspects of chat bot development at *chatbots.org* and learn more about chat bots at the *chatbotsmagazine.com*.

Name	Programming Languages	Service
Bot Framework (Microsoft) ⁴⁹	.NET, Node.js, REST	Natural language understanding, automatic speech recognition, visual search, image recognition, text-to-speech, machine learning
Chatfuel ⁵⁰	n/a	Facebook Messenger, Telegram (kik, Slack, Viper, WhatsApp announced)
ChatScript (Morton, Willcox) ⁵¹	Own scripting language	Manual
Facebook Messenger Platform (Facebook) ⁵²	Node.js	Facebook Messenger, Website
Motion.ai ⁵³	n/a	Email, Facebook Messenger, Slack, Smooth, SMS, Website

⁴⁹ <https://dev.botframework.com>

⁵⁰ <https://chatfuel.com>

⁵¹ <https://sourceforge.net/projects/chatscript>

⁵² <https://messenger.fb.com>

⁵³ <https://www.motion.ai>

Name	Programming Languages	Service
Pandorabots ⁵⁴	AIML, Go, Java, Node.js, PHP, Python, Ruby	Manual
Snips.ai ⁵⁵	Any (MQTT)	On device voice assistant

On-Device Machine Learning

To ensure privacy and to improve round-trip times, it makes sense to integrate machine learning routines directly on the device. This, of course, requires more effort and know-how compared to using an online API. Find some popular options here:

Name	Platform
Apple Core ML ⁵⁶	iOS
Facebook Caffe2 ⁵⁷	iOS, Android, Windows
Google TensorFlow ⁵⁸	Android
Qualcom Snapdragon Neural Processing Engine SDK ⁵⁹	Android (with supported Qualcomm CPUs or GPUs)

⁵⁴ <https://developer.pandorabots.com>

⁵⁵ <https://snips.ai>

⁵⁶ <https://developer.apple.com/machine-learning/>

⁵⁷ <https://caffe2.ai>

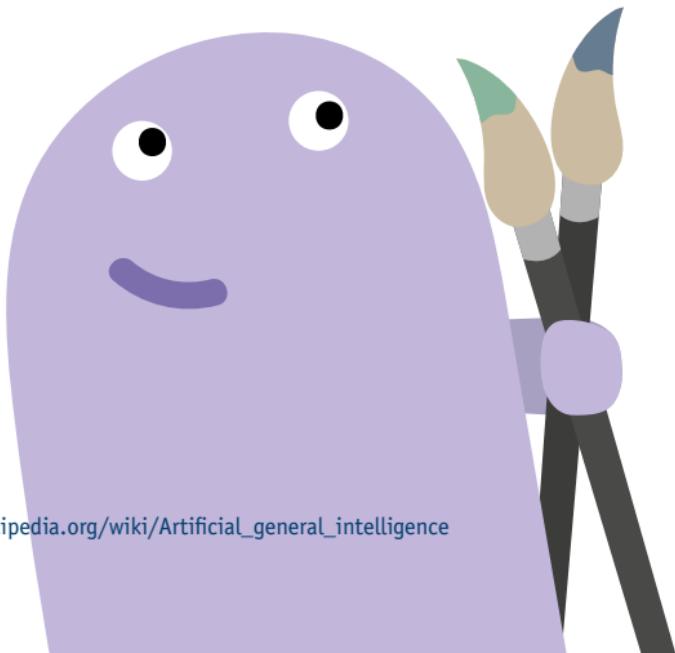
⁵⁸ <https://www.tensorflow.org>

⁵⁹ <https://developer.qualcomm.com/software/snapdragon-neural-processing-engine>

So, What's Next?

According to Google we live in an AI-first world, but the current crop of artificial intelligence pales compared to a general artificial intelligence⁶⁰ of the future. – thankfully, there are already open projects such as *OpenAI.com* around, that try to keep such technology open.

The tools are in place, now it is up to you to realize their potential. While cloud services offer an easy entry, libraries that run on devices will soon catch up – bringing their privacy and performance benefits along, but also costing more battery life. This is when dedicated AI hardware such as IBM's True-North chip will play out its strengths. Then running client-side AI will become common practice.



⁶⁰ en.wikipedia.org/wiki/Artificial_general_intelligence





Security & Privacy

Readers of this guide know how widespread smart mobile devices have become and how useful mobile apps can be. Mobile devices are also much more personal than personal computers ever have been. People wake up with their phones, stay close to them all day, and sleep next to them at night. Over time they become our trusted 'partners'. We are also storing increasing amounts of sensitive data on our phones, such as health information, and that data is often synchronized with cloud services, so the security, privacy and integrity of that data is extremely important to consider.

Many of these apps take advantage of this closeness and trust. For instance, your phone might be treated as part of the authentication for accessing your bank account. Or your tablet could get direct access to the online movies you have bought. The device might even store a wallet of real money for making payments with Near Field Communications (NFC), or virtual money like Bitcoins.

Mobile apps are attracting the attention of hackers and thieves whose interests extend well beyond getting a 99 cent app for free. In Q1 2017 Kaspersky Lab detected 1,333,605 malicious mobile installation packages¹. A malware named "Judy" was found in dozens of Android apps and supposedly reached an astonishing spread of up to 18.5 million downloads². The historical network and endpoint based defenses (like anti-virus tools) are not enough. Embedding security into the mobile application is critical.

¹ securelist.com/it-threat-evolution-q1-2017-statistics/78475/

² blog.checkpoint.com/2017/05/25/judy-malware-possibly-largest-malware-campaign-found-google-play

The architecture of mobile apps continues to evolve. Some apps are native-only, and require distinctly different code bases for each different mobile operating system. Some are web-views, little more than a web site url wrapped in an icon. Others are hybrids, a combination of native app functionality with web views. Most mobile apps need to connect with backend services using web technologies to fetch or update information. Like web apps, classic application security needs to be used with mobile apps. Input needs to be validated for size, type, and values allowed. Error handling needs to provide useful error messages that do not leak sensitive information. Penetration testing of applications is needed to assure that identification, authentication and authorization controls cannot be bypassed. Storage on the devices needs to be inspected and tested to assure that sensitive data and encryption keys are not stored in plain text. Log files must not capture passwords or other sensitive information. SSL configurations need to be tested.

Users want to use your applications safely; they do not want unwelcome surprises. Their mobile phone may expose them to increased vulnerabilities, for instance potentially their location could be tracked using an inbuilt GPS. The camera and microphone could be used to capture information they prefer to keep private, and so on. Applications can also be written to access sensitive information such as their contacts. And malicious applications can covertly make phone calls and send SMS messages to expensive numbers.

The application developer may be concerned about his/ her reputation, loss of revenue, and loss of intellectual property. Users may be concerned about the privacy of their private and often sensitive data stored on the phone (and increasingly also in the cloud). Corporations want to protect business data which users may access from their mobile device, possibly

using your application. Can their data be kept separate and secure from whatever else the user has installed?

Why is Privacy Important?

Many apps store sensitive data that is private to the individual, such as health data. Even data that may not be considered sensitive is subject to data privacy laws, and must be treated appropriately.

When storing user data only on the device, the application developer is more concerned with security and integrity of data; protecting it from attackers who attack the physical device or OS. However, there are potentially still data privacy implications even if the data does not leave the user's device, for example in some countries the data protection laws require "collected data" to be secured from potential abuse, therefore as a developer you would need to show that you had made an attempt to follow, for example, the security advice and best practices in this book.

When storing or synchronizing data to the cloud, a whole range of data privacy and data protection laws can potentially come into play, particularly in the European Union. This is because now you are considered a "data controller" and thus responsible for the protection of that data. If you are acting as a supplier, subcontractor or partner to another party, i.e. they have the relationship with the customer and you are acting on their behalf, you may be considered a "data processor".

However, even when operating in countries that have little or no data-protection/data-privacy laws, it is well worth considering best practice for data privacy, following the recommendations made by the OECD³:

³ [oecd.org/sti/ieconomy/oecdguidelinesontheprotecti...
htm](http://oecd.org/sti/ieconomy/oecdguidelinesontheprotecti...)

- **Data Collection** - Ensure that there are limits to the personal data you collect, and always do so with the knowledge and consent of the user (e.g. do not collect data behind the user's back)
- **Quality** - Keep personal data up-to-date and relevant to the purpose for which it is used (e.g. if you do not need to know the users' age, do not store it)
- **Purpose** - When you collect personal data, inform the user as to why you want to store it
- **Use** - Personal data should only be used for the purpose(s) given to the user at the time of collection, and for nothing else (e.g. do not tell the user you are collecting their heart rate to improve fitness, and then disclose that data to insurance companies)
- **Security** - Protect the personal data against unauthorized access, disclosure or tampering
- **Openness** - Users should be able to find out what personal data you are storing about them

Privacy Laws in the European Union

It would be impossible to summarize worldwide privacy law in a few paragraphs, but beyond the advice given above, it is worthwhile understanding that the European Union's data protection laws go much further than most other countries, particularly in terms of the obligations and responsibilities of the "data controller". Many of those obligations are based on the principles laid down by the OECD, but worth it is worth pointing out some additional requirements:

- Transfer of personal data to "Third-Countries" (i.e. countries outside the EU) - Data may only be transferred to countries that provide adequate legal protection

of personal data. This typically requires an agreement between the EU and the countries in question, for example the United States is covered by the EU-US Privacy Shield Framework.

- The European Union has set a deadline of May 2018 for compliance to the new General Data Protection Regulation (GDPR)⁴, which harmonizes and improves the data protection legislation across the EU. Notably it states that it will "apply for all non-E.U. companies without any establishment in the E.U., provided that the processing of data is directed at E.U. residents". If you think your app may be affected by this legislation, and you have customers in the EU, it is well worth becoming familiar with EU GDPR.

Finally, it is also worth mentioning that although the European Union mandates a minimum set of privacy requirements, each country implements those into national law in their own way. For example Germany has even stricter privacy requirements than European law mandates. Thus, national privacy laws also need to be considered if conducting business in EU countries.

⁴ eugdpr.org

Threats to Your Applications

On some platforms (iOS and Android in particular), disabling the built-in signature checks is a fairly common practice. You need to consider whether or not it would matter to you if someone could modify your code and run it on a jail-broken or rooted device. An obvious concern would be the removal of a license check, which could lead to your app being stolen and used for free. A less obvious, but more serious, threat is the insertion of malicious code (malware) that could steal your users' data, or inject illicit content and destroy your brand's reputation.

Reverse-engineering your app can give a hacker access to a lot of sensitive data, such as the cryptographic keys for DRM-protected movies, the secret protocol for talking to your online game server, or the way to access credits stored on the phone for your mobile payment system. It only takes one hacker and one jail-broken phone to exploit any of these threats.

If your application handles real money or valuable content you need to take every feasible step to protect it from Man-At-The-End (MATE) attacks. And if you are implementing a DRM standard you will have to follow robustness rules that make self-protection mandatory.

Anytime your application connects to the internet/cloud and transmits sensitive data such a personally identifiable information (telephone number, email address, home address etc.), or information that is private to a user (not just passwords), it will typically do so over an insecure network, which may be subject to pervasive monitoring or Man-In-The-Middle (MITM) attacks. Your application should ensure that it is using encrypted transport protocols, and that network endpoints are suitably authenticated.

Protecting Your Application

Hiding the Map of Your Code

Some mobile platforms are programmed using managed code (Java or .NET), comprised of byte code executed by a virtual machine rather than directly on the CPU. The binary formats for these platforms include metadata that lays out the class hierarchy and gives the name and type of every class, variable, method and parameter. Metadata helps the virtual machine to implement some of the language features (e.g. reflection). However, metadata is also very helpful to a hacker trying to reverse engineer the code. Decompiler programs, freely available, regenerate the source code from the byte code, and make reverse engineering easy.

The Android platform has the option of using the Java Native Interface (JNI) to access functions written in C and compiled as native code. Native code is much more difficult to reverse engineer than Java and is recommended for any part of the application where security is of prime importance.

“gcc” is the compiler normally used to build native code for Android, its twin-sister “clang” is used for iOS. The default setting for these compilers is to prepare every function to be exported from a shared object, and add it to the dynamic symbol table in the binary. The dynamic symbol table is different from the symbol table used for debugging and is much harder to strip after compilation. Dumping the dynamic symbols can give a hacker a very helpful index of every function in the native code. Using the `-f visibility` compiler switch⁵ correctly is an easy way to make it harder to understand the code.

⁵ <http://gcc.gnu.org/wiki/Visibility>

Compiled Objective-C code contains machine code and a lot of metadata which can provide an attacker with information about names and the call structure of the application. Currently, there are tools and scripts to read this metadata and guide hackers, but there are no tools to hide it. The most common way to build a GUI for iOS is by using Objective-C, but the most secure approach is to minimize its use and switch to plain C or C++ for everything beyond the GUI.

Hiding Control-Flow

Even if all the names are hidden, a good hacker can still figure out how the software works. Commercial managed-code protection tools are able to deliberately obfuscate the path through the code by re-coding operations and breaking up blocks of instructions, which makes de-compilation much more difficult. With a good protection tool in place, an attempt to de-compile a protected binary will end in either a crashed de-compiler or invalid source code.

De-compiling native code is more difficult but can still be done. Even without a tool, it does not take much practice to be able to follow the control-flow in the assembler code generated by a compiler. Applications with a strong security requirement will need an obfuscation tool for the native code as well as the managed code.



Protecting Network Communications

Network communications are vulnerable to snooping and injection attacks. Apps can be installed and inspected in emulators or simulators. Network analyzers are freely available and able to monitor, intercept, change and redirect network traffic. Some governments monitor electronic communications for censored topics.

Protect all communications using TLS, (i.e. HTTPS instead of HTTP). However, that means more than simply enabling TLS; there are subtleties to encrypted transport that are often ignored, for example:

- You should validate that the hostname or common name of the HTTPS server they connect to is the correct, expected one.
- You should ensure that the cipher-suites your app supports do not include deprecated or outdated algorithms such as RC4 or MD5.
- You should authenticate the server certificate(s) against the keychain provided by the OS, including checking the host/domain name against the one specified in the URL, and checking the certificate is still valid.
- You should not accept self-signed certificates, as these allow potential MITM attacks.

Downloads of javascript libraries from public sources, like map libraries, should use HTTPS, as hackers, using MATE attacks, can inject malicious code into the download if HTTP is used. Downloads of static content, like pictures, from public sources, should use HTTPS, as hackers could replace images with malicious content. One way to step up from transport security is to use asymmetric encryption between the server and the mobile app (using public/private key pairs) to provide end-to-end security.

For sensitive corporate data and applications, install Virtual Private Network (VPN) servers, and install VPN clients on the mobile devices. VPNs generally provide strong authentication, and secure transport above and beyond HTTPS.

You can also consider using DNSSEC to verify that the IP address for the endpoint you are contacting is the correct one, otherwise attacks such as DNS cache poisoning can be used to return malicious IP addresses.

Protect Against Tampering

You can protect the code base further by actively detecting attempts to tamper with the application and respond to those attacks. Cryptography code should always use standard, relatively secure cipher algorithms (e.g. AES, ECC), but what happens if an attacker can find the encryption keys in your binary or in memory at runtime? That might result in the attacker unlocking the door to something valuable. Even if you use public key cryptography and only half of the key-pair is exposed, you still need to consider what would happen if an attacker swapped that key for one where he already knew the other half. You need a technique to detect when your code has been tampered. Tools are available that encrypt/decrypt code on the fly, run checksums against the code to detect tampering, and react when the code has changed. You should also consider encrypting keys with a user-supplied secret in cases where that makes sense, although you will need to provide appropriate warnings about not being able to recover the data if the user forgets his password!

Communications can be monitored and hacked between the mobile app and backend services. Even when using HTTPS, a MITM such as an intercepting web proxy (like Paros) can be setup on a WiFi connection that will inspect encrypted traffic. This is why it is so important to correctly authenticate server certificates and validate against the OS root keychain, as the only way a MITM attack can then succeed is with a wildcard certificate.

Protecting Cryptographic Algorithms

An active anti-tampering tool can help detect or prevent some attacks on crypto keys, but it will not allow the keys to remain hidden permanently. White-box cryptography aims to implement the standard cipher algorithms in a way that allows the keys to remain hidden. Some versions of white-box cryptography use complex mathematical approaches to obtain the same numerical results in a way that is difficult to reverse engineer. Others embed keys into look-up tables and state machines that are difficult to reverse engineer. White-box cryptography will definitely be needed if you are going to write DRM code or need highly-secure data storage.



Best Practices

Do Not Store Secrets or Private Info

Minimize the amount of sensitive information stored on the device. Do not store credentials or encryption keys, unless secure storage is used protected by a complex password. Instead, store authentication tokens that have limited lifetime and functionality.

Log files are useful for diagnosing system errors and tracking the use of applications. But be careful not to violate the privacy of users by storing location information, or logging personally identifiable information of the users. Some countries have laws restricting the tracking information that can be collected — so be sure to check the laws in the countries in which your app will be used.

Do not print stack traces or system diagnostics that hackers can leverage to penetrate further.

Do Not Trust The Device

When you design an application, assume that the device will be owned by an attacker trying to abuse the app. Perform the same secure software development life cycle when building mobile apps as you would for backend services. Do not trust even the databases you create for your mobile apps — a hacker may change the schema. Do not trust the operating system to provide protection — many OS protections can be bypassed trivially by jailbreaking the device. Do not trust that native keystores will keep data secret — some keystores can be broken by brute force guessing unless the user protects the device with a long complex password.

Minimize Permissions

Android has the concept of permissions, iOS has entitlements, which allow the application access to sensors such as the GPS and to sensitive content. On Android these permissions need to be specified as part of creating the application in the `AndroidManifest.xml` file. They are presented to the user when they choose to install the application on their device.

Each permission increases the potential for your application to do nefarious things and may scare off some users from even downloading your application. So aim to minimize the number of permissions or features your application needs.

Perform Threat Modeling

Threat Modeling is a way of designing software and applications with security in mind. It focuses on key questions such as those outlined by Adam Shostack in his seminal book on Threat Modeling⁶:

- What are you building?
- What can go wrong?
- What should you do about the things that can go wrong?
- Did you do a decent job of analysis?

There are a variety of techniques for each stage, for example diagramming your application architecture can be a good start at the first question, and answering the second question can be done by using techniques such as STRIDE analysis (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege).

⁶ Adam Shostack: Threat Modeling: Designing for Security (John Wiley & Sons, 2014)

Even if you do not use any established techniques for threat modeling, you should definitely be thinking about the above questions when designing, implementing and maintaining your app. These are questions not only to be asked when first creating the app, but rather continuously as it evolves and you add new features.

Tools

Protection

Basic Java code renaming can be done using Proguard⁷, an open-source tool and Arxan's GuardIT⁸.

Two vendors for managed-code (Java and .NET) protection tools are Arxan Technologies⁹ and PreEmptive Solutions¹⁰.

The main vendors for native code protection tools and white-box cryptography libraries are Arxan and Irdeto¹¹.

Main vendors for secure mobile source code scanning are Checkmarx¹² and HP¹³.

Techniques for protecting Android code against tampering are documented at *androidcracking.blogspot.com/*. Arxan's EnsureIT allows you to insert extra code at build time that will detect debuggers, use checksums to spot changes to the code in memory and allow code to be decrypted or repaired on-the-fly.

⁷ www.proguard.sourceforge.net

⁸ arxan.com

⁹ arxan.com

¹⁰ preemptive.com

¹¹ www.irdeto.com

¹² checkmarx.com

¹³ www8.hp.com/us/en/software-solutions/mobile-app-security/index.html

Sniffing

A standard free web proxy tool is Paros¹⁴. A standard network sniffing tool available on common platform is Wireshark¹⁵.

De-Compiling

See the Hex Rays de-compiler¹⁶.



¹⁴ sourceforge.net/projects/paros

¹⁵ sourceforge.net/projects/wireshark

¹⁶ www.hex-rays.com

Learn More

Here are some useful resources and references which may help you:

- Apple provides a general guide to software security¹⁷. It also includes several links to more detailed topics for their platform.
- Commercial training courses are available for iOS and Android. Lancelot Institute¹⁸ provides secure coding courses covering iOS and Android.
- A free SSL tester is provided by Qualsys Labs¹⁹.
- Extensive free application security guidance and testing tools are provided by OWASP²⁰, including the OWASP Mobile Security Project²¹.
- A free mobile application performance monitoring tool for iOS and Android is the AT&T Application Resource Optimization tool²².

¹⁷ developer.apple.com/library/mac/navigation/#section=Topics&topic=Security

¹⁸ www.lancelotinstitute.com

¹⁹ www.ssllabs.com/ssltest

²⁰ www.owasp.org

²¹ www.owasp.org/index.php/OWASP_Mobile_Security_Project

²² developer.att.com/application-resource-optimizer

The Bottom Line

Mobile apps are becoming ever more trusted, but they are exposed to many who would like to take advantage of that trust. The appropriate level of application security is something that needs to be considered for every app. In the end, your app will be in-the-wild on its own and will need to defend itself against hackers and other malicious threats, wherever it goes.

Invest the time to learn about the security features and capabilities of the mobile platforms you want to target. Use techniques such as threat modeling to identify potential threats relevant to your application. Perform code reviews and strip out non-essential logging and debugging methods. Run a secure code analysis tool against your mobile code to find vulnerabilities. Consider how a hacker would analyze your code, then use similar techniques, in a safe and secure environment, against your application to discover vulnerabilities and mitigate these vulnerabilities before releasing your application.

Pick trustworthy services and partners. Often we use and rely on third-parties to serve our users and stakeholders. Their approaches to privacy matter as do their practices. Using companies who hold themselves to high standards and who are clear about what they do - and do not do - can help us protect our users and stakeholders. A good example is appfigure's Privacy Policy²³.

For network-based apps, always use TLS for communication and ensure you securely authenticate the network endpoints. And take the time to consider the privacy implications for any data you collect about or on behalf of the user, particularly if storing or synchronizing that data into the cloud.

²³ appfigures.com/privacy



Accessibility

Why Accessibility is Important

According to the World Health Organization (WHO) over 15% of the world's population have some form of disability¹ and rates of disability are increasing due to population aging and increases in chronic health conditions, among other causes. This means that around 1 billion potential users could have difficulties using your app if your app is not accessible.

There has been a huge increase in smartphone and tablet use in the general population, this is no different for those with disabilities. The WebAIM Screen Reader Survey² shows that there has been an astounding increase in smartphone use by blind people who use screen readers. Older people might not have used a computer at work; however they are finding that they can get to grips with touch screen devices more quickly than a traditional keyboard and mouse. As our population ages, the levels of disability increase and this means more and more people will have difficulty accessing services in the traditional way. Providing an alternative accessible digital solution, will ensure disabled people can continue to be independent.

For example, if they are unable to get out of the house to do their shopping or banking, then providing accessible online services means they can access these services independently. It is important to recognize how important independent access to services is for people with disabilities. Apple have put some videos together with disabled people showing how they use

¹ www.who.int/mediacentre/factsheets/fs352/en

² webaim.org/projects/screenreadersurvey6/

technology in their everyday life³. This demonstrates just how important accessibility is to enabling independence.

There are lots of other reasons to make your apps accessible:

- Implementing accessibility can often improve overall usability: For example, if you ensure that every button and form element has appropriate label, that is helpful to everyone, not just those with disabilities as the user will know how to interact with it. Embedding accessibility into your apps ensures an excellent user experience for all.
- It just makes good business sense: For example, people with disabilities have spending power and if they find an accessible app that works for them they will not only use it, they will also tell others. You may discover a significant new market when you develop apps that suit these users.
- Access to goods and services for all is the law in many countries: For example in the UK the Equality Act 2010 requires there to be access to goods and services for everyone and this does include services which are provided via an electronic means such as websites and apps. Public bodies also have an anticipatory duty to ensure their services are accessible, so they cannot consider accessibility as an afterthought.
- Where accessible solutions are mandated by legislation, your app may be the only option for that business to realistically use: For example, your app may be able to tap into government funded market sectors such as education where legislation, such as Section 508 of the Rehabilitation Act in the US, may mandate an accessible solution.
- The organization that the app is being developed for may

³ www.apple.com/accessibility/stories/

have a corporate social responsibility (CSR) statement or program: For example, web and app accessibility provides social inclusion for the people with disabilities which is a primary aspect of corporate social responsibility.

- Mobile platforms from Apple, Google and Microsoft leverage their accessibility APIs for UI automation testing: Making your app accessible can make automated testing easier.

What Accessibility Features?

As many of your potential users may have a disability this can make it more difficult for them to use a mobile phone and related apps. Disabilities could include various levels of sight or hearing impairment, cognitive disabilities or learning difficulties, physical disabilities, dexterity issues, and so on.

Many of these users rely on third-party software to assist them in using their devices. This software is sometimes called Assistive Technology, and includes different utilities depending on the type of disability. Traditionally these types of software or utilities have had to be 'added on' to a mainstream device, often at high cost, in order to make them accessible or easier to use for someone with a disability. Many smartphones and tablets now provide robust enough Assistive Technology built into the operating system that some users with disabilities can use the devices without needing to pay for extra Assistive Technology. What is offered depends on the platform and the version of the OS. However - to work - these features may need the app to be designed and implemented to support them.

- **Partially sighted users** - Someone who is partially sighted benefits from being able to change the font size, font

style, colors and use of bold and color contrast too. iOS, Android and Windows offer various options to change these in the settings. As well as the universal 'pinch to zoom' feature, iOS, Android and Windows offer a magnification or zoom feature, which enlarges a section of the screen and keeps this magnification level when moving throughout the phone. This has unique gestures associated with it and often each OS has its own unique gestures. iOS also has a built in app which utilizes the camera on the phone to aid with spot reading on items such as clothes labels and restaurant menus.

- **Blind users** - Someone who is blind has to have information on the screen and navigation around the screen announced to them in synthetic speech. This is often called a 'screen reader'. iOS was the first OS to offer a screen reader built-in and it is called 'VoiceOver'. Android offers 'Talkback' which is fast catching up in popularity with the blind community as it is constantly improving. Windows first delivered the Narrator screen reader in Windows Phone 8.1 and it is even more improved now in Windows 10 Creators Update. Blind users may also make use of a Braille display, which is an item of hardware which provides feedback from the screen one text line at a time in the form of a line of Braille characters. Each Braille character consists of six or eight movable pins in a rectangular array. Most OS versions now support braille displays via Bluetooth.
- **Users with hearing loss** - Someone with a hearing impairment will often make use of a smartphone that is hearing aid compatible and offers features as iOS does such as 'LED Flash for Alerts' or 'Phone Noise Cancellation'. There are also options in settings for iOS, Android and Windows to switch on subtitles and captioning. Making use of

vibration for alerts is also helpful and haptic feedback has improved in recent versions of iOS in particular. A number of phones also provide support for hearing aids and tele-type (TTY) devices⁴. A TTY device allows people who have hearing loss or who are speech impaired to type messages to anyone else who has a TTY, using a telephone line.

- **Users with physical disabilities** - If a user has a motor impairment, they may well be using a third party hardware product to access the phone, such as a switch as some devices do support this. Alternatively they could be making use of voice recognition to access the device. Siri in iOS now enables the user to access certain settings and functions and switch them on and off.
- **Users with a learning disability** - If a user has a cognitive impairment or learning difficulty, then depending on what the disability is, they may make use of the features in the settings that a partially sighted user does. Especially something like color options. Other users may make more use of voice recognition.

For people with disabilities, their overall experience is affected by how well an app works with the assistive technology. As these features are built into the OS and can be switched on in the settings, it is important that as a developer you consider that they may be used with your app and ensure you test for this.

As screen readers and screen magnification in the OS makes use of their own gestures, gestures in the app may be affected when screen readers or magnification are enabled. For example

⁴ A TTY device allows people who have hearing loss or who are speech impaired to type messages to anyone else who has a TTY, using a telephone line.

a screen reader user can navigate a screen using left and right swipes or by exploring the screen by moving their finger across the screen of the device in a consistent movement. As they undertake a swipe, or encounter something underneath their finger, the item is announced. So an item is selected by tapping once and opened by tapping twice. When using screen magnification, depending on the OS, they may need to use a three finger gesture. Including testing early on with accessibility features ensures that these gestures are supported by the app and that any redesign can happen before it impacts on users.

One of the best ways to learn more about these features is to switch them on and try them for yourself in different apps.

App Design Guidelines

The accessibility APIs look for text in specific attributes of standard UI elements. Screen readers used by blind people, such as VoiceOver and TalkBack, transform the text into synthetic speech which the user listens to. The screen reader software may also determine the type of control and related attributes to help provide the user with more contextual information, particularly if no text is available. It is important that the user understands what the label of the control is, what the control is and how to interact with it. In some instances there may also be a tooltip to give extra information.

Just as web developers make use of standards and guidelines such as WCAG 2.0 to make accessible websites, it is important that as app developers, you do the same. At present there is no de facto industry standard for app accessibility, although there are standards out there that can help.

The international standard, ISO 9241-171 ('The Ergonomics of Human-system Interaction: Guidance on Software

Accessibility')⁵ is a helpful standard as it is platform agnostic. This covers elements of accessibility and usability for a wide range of software.

The Royal National Institute of Blind People (RNIB)⁶ have created a pan-disability app standard and testing process based on their experience in this area of accessibility. Their standard for native apps also reflects on principles from ISO 9241-171. They provide consultancy and training for organisations and agencies in this area and have an accreditation badge that can be awarded to apps that, following an audit process and user testing, are accessible. This accreditation is called 'RNIB Approved'⁷.

The BBC have developed a set of BBC Mobile Accessibility Guidelines⁸ that they use internally for their mobile content. Their guidance covers mobile websites, hybrid and native apps. They state that "they are intended as a standard for BBC employees and suppliers to follow however they can also be referenced by anyone involved in mobile development".

Here are some of the principles that are helpful to be aware of when developing an app. If you stick to them, you will also give your app the best chance of interoperating with assistive technology that the user may be running in conjunction with your software:

⁵ www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39080

⁶ rnib-business.org.uk

⁷ For more information contact RNIB Business at businesslink@rnib.org.uk.

⁸ www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile_access.shtml

APIs and UI Guidance

- Find out what accessibility features and APIs your platform has and follow the best practice in leveraging those APIs if they exist.
- Use standard rather than custom UI elements where possible. This will ensure that if your platform has an accessibility infrastructure or acquires one in the future, your app is likely to be rendered accessible to your users
- Use the Accessibility API for your platform, if there is one. This will enable you to make custom UI elements more accessible and will mean less work on your part across your whole app.
- Follow the standard UI guidelines on your platform. This enhances consistency and may mean a more accessible design by default
- The user should be able to apply their preference settings that the OS provides, such as accessibility settings.

Navigation

- Navigation should be logical and consistent. For example if a back button is provided on each screen it should be located in the same place on every screen and consistently labeled.
- Support programmatic navigation of your UI. This will not only enable your apps to be used with an external keyboard but will enhance the accessibility of your app on platforms such as Android where navigation may be performed by a trackball or virtual d-pad.

User elements

- All user elements should be discoverable and operable via assistive technology, unless it is clear they are not required.
- Where a user element has a status associated with it, that status should also be available to be read by assistive technology. For example if a toggle button is 'on' this should be announced by the screen reader. If the status changes, that should also be announced.
- Ensure touch screen targets are a reasonable size to ensure everyone can easily select them.

Labeling

- All elements, including form elements, buttons, icons and so on, should be labeled visually and programmatically with a short and descriptive name. The label should also be adjacent to the element it relates to.
- Each screen should have a unique descriptive name that relates to its content and aids navigation.

Colors and Fonts

- Ensure there is a good contrast between background and foreground colors. In particular consider buttons which include text. Does the contrast between the text and the background color meet the ratio requirements in WCAG 2.0 or ISO 9241-171?
- Avoid using color as the only means of differentiating an action. A color-blind user will not be able to identify errors if they are asked to correct the fields which are highlighted in red for example.
- Consider the size of your smallest font. Is it reasonable that most people could read it without difficulty?

Notifications

- Error messages, notifications and alerts should be consistent, identifiable and clear. They should be announced by the screen reader and clearly visible, ensuring they do not disappear from the screen after a short time period.
- Ensure that error messages, notifications and alerts are not provided by color/haptic/audible output alone. For example, someone with hearing loss will not recognize audible notifications.



Testing

- Do not forget to test your app on the target device with the assistive technology built-in to the OS with more than just the latest OS version. When testing on an Android device please remember that unless the user has a pure Android device, like a Google Pixel, they are unlikely to get access to all of the latest OS upgrades. This is because for OS upgrades you are at the mercy of your phone manufacturer, so the Android OS versions in the wild can be quite diverse. Because some handset manufacturers skin the OS, this can sometimes interfere with the way the accessibility features should work. Therefore, it is always recommended that testing for Android Accessibility is undertaken on a Google device. That way you can be sure there is nothing interfering with the way the accessibility features should work and you are working to a common denominator.
- Ensure your user testing includes people with disabilities too!

Apple, Google and Microsoft, have increased the importance of their respective Accessibility support by using the Accessibility interface to underpin their GUI test automation frameworks. This provides another incentive for developers to consider designing their apps to be more accessible.

Looking at the different mobile platforms more closely, it becomes obvious that they differ largely regarding their APIs, but they are starting to implement a lot of the same accessibility features.

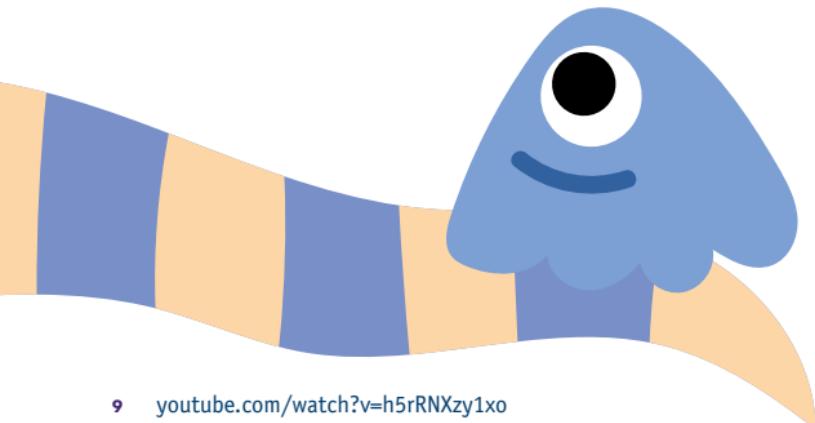


Custom Controls and Elements

If you are using custom UI elements in your app, then, those platforms that have an Accessibility API enable you to make your custom controls accessible. You do this by exposing the control to assistive technology running on the device so that it can interrogate the properties of the control and render it accessibly.

You can get more information about Accessibility on Android from various resources on YouTube including the Google I/O presentations from 2017⁹, 2016¹⁰, 2015¹¹ and 2013¹².

The Apple developer program has helpful resources too. Take a look at their accessibility video presentations from the WWDC conferences available in the iOS Developer Center¹³ by searching for 'accessibility'.



⁹ youtube.com/watch?v=h5rRNXzy1xo

¹⁰ youtube.com/watch?v=2qjgxH384Nc

¹¹ youtube.com/watch?v=euEsfNR5Zw4

¹² youtube.com/watch?v=ld7kZRpMGb8

¹³ developer.apple.com/wwdc/videos

Android App Accessibility

Accessibility was first a realistic proposition with Android 4.1 (Jellybean) and it is much improved since then. In Android 7 ("Nougat"), there has been more prominent featuring of accessibility settings to let users independently configure their device. 'Display size' was added as an option and it gives the user an alternative view of the standard screen, making the icons and text larger as a native setting without any magnification or font adjustment. This is also available as a live preview. Individual OEMs have created Android skins that offer the feature of a dark theme (bright text on a dark background) as it was removed in Android Nougat, but expected to return in future versions. 'BrailleBack' works with Talkback for a combined speech and braille experience enabling the user to add a braille display via bluetooth. It is now possible to change colors or contrast using 'high contrast text' or 'color inversion' options. This is equivalent to what Apple have undertaken with color filters in display accommodations. There is also the addition of a 'Click after cursor stops moving' option, which helps people with dexterity issues or with low vision. Also popular is the addition of a large mouse cursor.

Android 8 further enhanced accessibility by adding the possibility to control accessibility volume independently from media volume. It also introduced an accessibility shortcut for easily turning the accessibility service on and off from any screen.

Accessibility features in Android include (but are not limited to) things such as:

- **Talkback** - Speech output for blind users.
- **Select to speak** - Selective speech output for those who sometimes need some assistance.
- **Font Size** - For partially sighted users and some users with learning difficulties.
- **Magnification gesture** - Zoom style magnification for partially sighted users.
- **Display Size** - For partially sighted users and some users with learning difficulties.
- **Click after cursor stops moving** - For those with dexterity issues or who are partially sighted.
- **High Contrast Text** - For partially sighted users and some users with learning difficulties. This is an experimental setting.
- **Color inversion** - For partially sighted users and some users with learning disabilities who prefer an inverted color palette. This is still an experimental setting
- **Color correction** - This can change the balance of how the color is presented on the screen and aids those with color blindness. This is an experimental setting.
- **Color adjustment** - For users who have particular color preferences.
- **Captions** - Providing captions or subtitles for those with hearing loss.
- **Mono audio** - For those with hearing loss using headphones.
- **Switch access** - For those with physical disabilities who prefer to access apps using a hardware device.
- **Touch and hold delay** - For users with motor control issues.

There are some helpful resources in the Support Library¹⁴ which also includes ways to improve the accessibility of custom views.

For specifics on how to use the Android accessibility API along with details of best practice in Android accessibility, please see Google's document entitled Making Applications Accessible¹⁵.

You will also find more examples in the training area of the developer documentation in a section entitled Implementing Accessibility¹⁶. Testing the Accessibility is also covered online¹⁷.

iOS App Accessibility

Apple were the first company to embed accessibility features directly into the OS. Because of this the support for accessibility in iOS is a little better than in Android, although Android is fast catching up. There are certainly comparable features now but it is a legacy issue as Apple was the first to move into this area. A lot of blind and partially sighted users also find the gestures in iOS easier to use.

On iOS, several accessibility features are available in the display settings. This demonstrates that people have recognized that some of these settings are relevant for everyone. Accessibility is going mainstream as people just want to make the display more personal to them and have a device that is easy to use. This means it is even more important that

¹⁴ developer.android.com/tools/support-library/index.html

¹⁵ developer.android.com/guide/topics/ui/accessibility/apps.html

¹⁶ developer.android.com/training/accessibility/index.html

¹⁷ developer.android.com/tools/testing/testing_accessibility.html

developers consider accessibility settings when creating apps as the number of people using these features has increased.

There have been some hardware changes with the iPhones 7 and 7+ as the devices have a 'virtual' rather than physical home button. However, this button does feel fairly physical to a user as it provides haptic feedback. One other change does relate to haptic feedback generally as it is now natively deployed across the OS. If you use a 'picker wheel', it now delivers a discernible click when it is moving through options. This is a mainstream usability feature but certainly enables some disabled users to use the picker wheels more easily. Apple have also put more granularity into the haptic feedback that they provide, as the user can create lots of levels of sensitivity in a way they can not do on other OS's.

Some of the accessibility features in iOS include, but are not limited to:

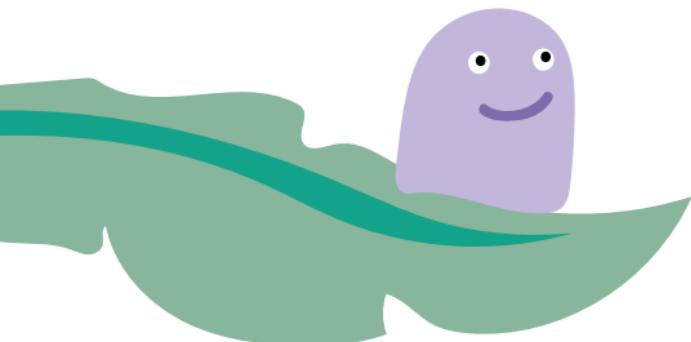
- **VoiceOver** - A screen reader. It reads out the objects and text on screen, enabling your app to be used by people who are blind.
- **Zoom** - This magnifies the entire content of the screen.
- **Magnifier** - This uses the camera to magnify items and can be used for spot reading. When enabled, the home button can be triple clicked to activate it.
- **Invert Colors** - This option inverts the colors on the display, which helps many people who need the contrast of black and white but find a white background emits too much light.
- **Color Filters** - This helps color blind users differentiate colors and also aids those who have difficulty reading text on the display.
- **Reduce White Point** - This reduces the intensity of bright colors.

- **Larger Text** - This can help a broad range of people from those who use glasses, through to partially sighted people and those with learning difficulties.
- **Bold text** - This can help a broad range of people from those who use glasses, through to partially sighted people and those with learning difficulties.
- **Increase Contrast** - There is an option to reduce transparency which takes the transparency from certain areas such as the notification shade and control center and folders
- **Switch control** - For those with physical disabilities who wish to access the app using a third party hardware device.
- **AssistiveTouch** - For users who have difficulty touching the screen or might need to create custom gestures.
- **Touch Accommodations** - These options give the user the ability to change the settings, on how the screen will respond to touch gestures.
- **Hearing devices** - Includes the connections for hearing aids for people with hearing loss.
- **LED flash for alerts** - To enable people to choose what works best for them for notifications, particularly helpful for those with hearing loss.
- **Mono Audio** - Helpful for those with hearing loss.
- **Phone Noise Cancellation** - To reduces the ambient noise on phone calls when you are holding the receiver to your ear. Helpful for all, but also used by people with hearing loss.
- **Subtitles and Captioning** - For people with hearing loss.
- **Audio Descriptions** - For people with sight loss.
- **Guided Access** - This is helpful in education, or just where someone wants to limit what is accessible on the screen to a user.
- **Accessibility shortcut** - Set up to switch features on and off using the home button.

- **Siri** - This enables users to make phone calls and operate various other features of their phone by using voice commands. This can be helpful for a broad range of people including those with motor control issues, learning difficulties and vision loss.

If you are working on iOS, make sure check out Apple's Developer area¹⁸ and to follow Apple's accessibility guidelines¹⁹. These guidelines detail the API and provide an excellent source of hints and tips for maximising the user experience with your apps.

Apple also provide some helpful guidance on testing the accessibility on your app with Voiceover²⁰.



¹⁸ developer.apple.com/accessibility/ios/

¹⁹ developer.apple.com/library/ios/documentation/UserExperience/Conceptual/iPhoneAccessibility/Introduction/Introduction.html

²⁰ developer.apple.com/library/ios/technotes/TestingAccessibilityOfiOSApps/TestAccessibilityonYourDevicewithVoiceOver/TestAccessibilityonYourDevicewithVoiceOver.html

Windows App Accessibility

It is fair to say that Microsoft have been playing catch up with iOS and Android as far as accessibility features go because they were later to deliver built in accessibility features to phones. Although they had made a move towards accessibility with earlier Windows phone versions the initial delivery of Windows 10 and the Edge browser was somewhat of a backwards step accessibility wise. However, since the first release of Windows 10, Microsoft have made a commitment to put accessibility at the heart everything they do and universal design is central to that. Since this statement there has been an improvement in Windows 10, the Edge browser and Office applications, especially Office365.

There was good support for magnification, text enlargement and changing of colors in earlier Windows phones and in Windows Phone 8.1 things moved on again with the introduction of the screen reader Narrator which reads out text in synthetic speech. At the time of delivery it was not as comprehensive as the iOS and Android offerings and could only be used with core functionality and navigation functionality.

Now there has been a unified release of Windows 10 Creators Update which offers much more in the way of accessibility, particularly for those people with sight loss. This OS is available on a limited number of Windows devices. There is good support for magnification and colors and Narrator is a lot more fully featured and includes better information relating to the context of controls. There is even beta support for a number of braille displays and different languages. It is also possible to use a controller on Xbox 1 to handle Narrator.

Narrator can still be launched using Cortana and there are new text-to-speech voices which offer multilingual reading, by switching between voices dynamically. There is now a much

better link between Cortana and Narrator as Cortana now ignores the Narrator voice when the two are used together.

Some of the accessibility features on Windows 10 Creators Update include but are not limited to:

- **Narrator** - This is the screen reader for those users who are blind or have little vision.
- **Activate keys on touch keyboard when I lift my finger off the keyboard** - This is to assist those with dexterity issues.
- **Screen Magnifier** - This feature is for partially sighted people who wish to magnify the text on the screen and change the zoom level. It has its own gestures.
- **High Contrast Theme** - This theme changes text to black and white and provides a solid background behind words that would otherwise be on top of pictures. This is helpful for partially sighted users and some users with learning disabilities.
- **Closed Captions** - It is possible to change the font size, color, background and window transparency of captions. This is helpful for people with hearing loss that may also have some vision loss.
- **Text Scaling** - The size of text can be enlarged to aid those with learning difficulties or users who are partially sighted.
- **Cortana** - Is the 'personal assistant' that is only available on Windows Phone 8.1. This is a main feature for all users, but will be helpful for those with disabilities too as it is speech activated.

You can find out about accessibility for Windows apps

with some Microsoft video resources²¹ and platform specific documentation.

Microsoft has published Guidelines for Designing Accessible Apps and a dedicated paper about Accessibility for Windows 10 / UWP apps²².

Mobile Web App Accessibility

As mentioned earlier in the chapter, much has been written about web accessibility, but less has been written on accessibility relating to apps. This is also true of mobile website accessibility or web app accessibility. It is an area which has growing interest and the World Wide Web Consortium (W3C) have created a Mobile Accessibility Task Force²³ concerned with the required work in this area.

On the main W3C Mobile Accessibility page²⁴ you can find lots of helpful resources related to mobile accessibility.

It is suggested by the W3C that anything that uses HTML and is web based should still follow the Web Content Accessibility Guidelines (WCAG) 2.0 while also referring to Mobile Web Best Practices (MWBP). So if you are a web content developer, then these guidelines are a good place to start. You will also find Relationship between Mobile Web Best Practices (MWBP) and Web Content Accessibility Guidelines (WCAG)²⁵ a helpful resource.

²¹ developer.microsoft.com/en-us/windows/accessible-apps

²² docs.microsoft.com/en-gb/windows/uwp/usability/index

²³ www.w3.org/WAI/GL/mobile-a11y-tf/

²⁴ <http://www.w3.org/WAI/mobile/> www.w3.org/WAI/mobile/

²⁵ w3.org/TR/mwbp-wcag/

If your app is intended to mimic a native app look and feel, then you should follow the guidelines mentioned above in this chapter.

As support of HTML 5 is increasingly adopted on the various mobile platforms, consider reading Mobile Web Application Best Practices²⁶ as this is likely to form the foundation of any mobile web application accessibility standard that emerges in the future. One of the other key areas of guidance is Accessible Rich Internet Applications 1.0 (WAI-ARIA)²⁷, as it has been designed to ensure that more dynamic HTML functionality is accessible to screen readers.

An interesting area of work happening at the W3C is in the Independent User Interface (IndieUI) Working Group²⁸. The group states "Independent User Interface (IndieUI) is a way for user actions to be communicated to web applications and will make it easier for web applications to work in a wide range of contexts — different devices, different assistive technologies (AT), different user needs". This work is going to be very important for accessibility and device independence. It is worth looking at the documentation that they currently have available.

²⁶ w3.org/TR/mwabp

²⁷ w3.org/TR/wai-aria

²⁸ w3.org/TR/indie-ui-context

Developing Accessible Games

Accessible games for disabled users were always very basic, if there were any available at all. Now some developers are beginning to think about engaging with a wider audience when developing games and finding ways to prevent unnecessarily excluding players.

Much of the guidance and standards already shared will be very valuable when creating games as this information is still relevant to gaming development. Do not forget to ensure that any accessibility features are clearly obvious to users in the game description so when they make the choice to purchase, they know that the app may be of interest to them.

There are a couple of sites which have some accessibility guidelines for game developers. They include Game Accessibility Guidelines²⁹ and Includification³⁰.

If you would like to look at some accessible games that have already been developed, then Game Accessibility³¹ is a good resource.

²⁹ gameaccessibilityguidelines.com/

³⁰ www.includification.com/

³¹ game-accessibility.com/



Testing

Introduction

In practice, testing software has often been performed as a distinct one-time activity, after the code has been written and before the code has been released. Over the last decade testing industry leaders have seen the advantages of starting testing earlier in the project lifecycle, for instance involving testing in the design phase. This is sometimes known as 'shift-left' testing. And more recently testing is being extended to incorporate information and feedback gleaned post-deployment when the software is live. This is becoming known as 'shift-right' testing. These concepts are a great fit for mobile apps where users provide feedback in reviews and where mobile analytics and crash analytics provide additional information about usage of the app.

So testing is becoming more of a mindset and a continual practice than a one-shot activity. Teams who use 'Agile' development are sometimes prone to limit testing of a feature to within a sprint which may constrain and box-in the testing. New testing approaches may need to emerge that take a more holistic approach, not limited to testing within a sprint. More needs testing than the new stories and features, and testing should not be limited to testing the code we write - testing of libraries, tools, services, approaches, etc. can all help improve the quality of our work and enable teams to make better, more informed decisions rather than relying on advertising, gut-feel, or cost.

"But test them all; hold on to what is good"¹

Testing also needs to extend beyond the mobile app - our users expect no less! Apps often rely on third-party and cloud-based architectures and testing needs to factor these in. Users expect to be able to work seamlessly across multiple devices, for instance by creating an email with a photo attachment on their smartphone and then editing the email on their laptop using a web browser before sending it from yet another device and flavor of software. Therefore we need to consider how users may use our apps and whether the app extends beyond the mobile device. In parallel, mobile apps may be part of a larger ecosystem where smartwatches, fitness monitors, etc. are integrated and provide data to the app. Our testing needs to reflect the likely usage patterns by end users.

Finally for now, our apps need to cope well in an ever changing and often expanding microcosm. New releases of the platform, new devices, and new usage patterns can all expose weaknesses and limitations in an app. Unless we actively and continually pay attention our app will be left to fend for itself and may let down or disappoint our users.

Testing might be seen as an impediment but failures in your app can be all too public. And recovering your credibility is hard when your app has a poor score in the app store. Rather than waiting for users to decide the fate, testing your mobile apps can adjust the balance in your favor. By reading this chapter you have the opportunity to help equip you and your testing team so they can test your app more effectively.

This chapter covers the general topics; testing for specific platforms is covered in the relevant chapter.

¹ 1 Thessalonians 5:21

Beware of Differences

Platforms, networks, device settings, device models, and even firmware, are all specific and differ from each other. Any could cause problems for your applications². This means we need a variety of devices to test on.

The range and variety of devices continues to accelerate, and users are increasingly connecting mobile devices together, for example IoT devices, wearables, cars or home appliances. More and more devices are needed for a testlab to create sufficiently representative test environments. Obtaining additional devices is important. From a user's perspective a mobile app is the combination of software, hardware and environment. They want and expect an app to work regardless of all these details. Furthermore, the boundaries of what needs to work extends even further, as many apps now interact with external devices and sensors. As an example, Disney have designed seamless experiences using smart wristbands³ which integrate into a larger ecosystem. Devices can also be used to pay for travel⁴ and shopping, cameras are used by banking software to identify and authorize cash withdrawals, and so on. Any problems, incompatibilities, and so on can adversely affect the UX and may have significant impact for instance if users are denied access to transport, money, and more.

A basic strategy for testing mobile apps is to assume that every combination is unique and different from another and will behave slightly differently. It would be impractical to test each combination, a more fruitful approach is to invest time

² An example of device specific problems with Android on Samsung is anasambri.com/android/special-place-for-samsung-in-android-hell.html

³ wired.com/2015/03/disney-magicband

⁴ vodafone.nl/shop/mobiel/abonnement/extr-opties/smartlife/wallet/reizen

in learning what the impact of the differences are and testing a subset of the combinations that maximizes the insights and confidence we have in the behavior of the app across most of the combinations. Key skills include:

- Device analysis: What are the main differences? And when are these differences relevant to the app? (And which ones can we ignore).
- Extrapolation: What does a test on one device say to the thousand of devices out there in the wild?

Discovering Differences

There are several ways to identify the effects of specifics, for instance, a tester may notice differences in the performance of the app and the behavior of the UI when testing on different devices. Automation may also detect differences and can help you selecting devices that support the required features⁵.

Conversely, Mobile Analytics can help identify differences in various aspects including performance and power consumption when the app is being used by many users on the vast variety of their devices. Some compelling examples of differences in behavior and on ways issues were addressed in a paper published by computer scientists from the University of Wisconsin⁶. A book is also available from HP Enterprise on the

⁵ mobiletestingblog.com/2017/05/30/optimizing-android-test-automation-development

⁶ "Capturing Mobile Experience in the Wild: A Tale of Two Apps", available as a download at static.googleusercontent.com/media/research.google.com/en//pubs/archive/41590.pdf

confluence between mobile analytics and testing for mobile apps⁷.

Testing Needs Time - You Need a Strategy

The strategy defines how much test time is spent to the different parts of the mobile app and during the different development phases. There are tradeoffs on how best to spend whatever time you have. For instance, testing features in more detail versus testing on a wider variety of devices versus testing various quality aspects including performance, usability and security.

The conditions a mobile app need to operate are vast and to factor in these conditions into the testing is challenging. There may be more productive ways to obtain some information, for instance by using feedback from end-users and from mobile analytics. However, the risks of deferring data gathering (versus testing internally) need to be actively considered. An effective test strategy aims to balance both approaches. With the risk analysis, the quality perspectives and the available time the test plan can be created.

Continuous Testing

Continuous delivery needs continuous testing. Viable apps need to be updated on an ongoing basis. Updates may include fixes for new platform versions or device models, new functionality and other improvements. Therefore, testing is not a one-off task; high quality apps befit ongoing, optimized testing, including testing in production. Production testing

⁷ themobileanalyticsplaybook.com (co-written by Julian Harty, one of the authors of this chapter)

includes testing engagement and validation as well as early detection of potential problems before they mushroom.

Manage your Testing Time

Testing as you have discovered can take many hours, far more than you may want to do, particularly if you are close to a deadline such as a release date. There are various ways you can manage time spent in testing, in parallel testing can be made more interesting, rewarding, and more productive.

- **Reduce setup time:** Find ways to deploy apps quickly and efficiently. Implement mechanisms to provide the appropriate test data and configuration on both the mobile device and the relevant servers. Aim to have devices and systems 'ready to test'.
- **Reduce time needed for reporting & bug analysis:** Data, screenshots, and even video, can help make bugs easier and faster to investigate. Data can include logs, system configurations, network traffic, and runtime information. Commercial tools can record actions and screenshots to reduce the time and effort needed to report and reproduce problems.
- **Risk analysis:** You can use the risk analysis to decide how and when to allocate testing effort. Risks are hard to determine accurately by the tester or developer alone; a joint effort from all the stakeholders of the mobile app can help to improve the risk analysis. Sometimes, the mobile app tester is the facilitator in getting the product risk analysis in place.
- **Scaling testing:** Increasing the throughput of testing by scaling it, for instance using test automation, cloud-based test systems, and more humans involved in the testing can help increase the volume, and potentially the quality,

of the testing. Using static analysis tools to review code and other artifacts can also help the team to find and fix problems before the app is released.

Involve End-Users in your Testing

Development teams need a mirror to develop a useful mobile app. Early user feedback can provide that mirror. You do not need many end-users to have good feedback⁸. Bigger value is gained with early involvement, multiple users, regular sessions, and multiple smaller tests. Testers can guide and facilitate the end-user testing, for instance, by preparing the tests, processing log files and analyzing results. They can also retest fixes to the app.

Whenever others are involved in testing an app, they need ways to access and use the app. Web apps can be hosted online, perhaps protected using: passwords, hard-to-guess URLs, and other techniques. Installable apps need at least one way to be installed, for instance using a corporate app store or specialist deployment services.

When the app is closer to being production-ready, users can test the more mature version of the mobile app in Alpha & Beta tests phases. A development team or organization can offer an online community to give end-users early access to new releases, give loyalty points, ratings. This community should be a friendly ecosystem to receive feedback before the mobile app is released into the app store.

⁸ nngroup.com/articles/why-you-only-need-to-test-with-5-users

Proxy users

There are various services available that facilitate testing by other people. These people might be users of your app, however they are more likely to be proxy users, people who take the role of end users and provide additional perspective and perhaps insights into the behavior of an app. Possible sources of these testers include crowdsourcing⁹. Lookback¹⁰ provides a different more personal flavor.

Effective Testing Practices

Testing, like other competencies, can be improved by applying various techniques and practices. Some of these need to be applied when developing your mobile app, such as testability, others apply when creating your tests, and others still when you perform your testing. Testdroid offers a good checklist¹¹ on getting the right testing expertise into your team.

Implementing Testability

Start designing and implementing ways to test your app while it is being developed. This applies especially for automated testing. For example, using techniques such as Dependency Injection in your code enables you to replace real servers (slow and flaky) with mock servers (controllable and fast). Use unique, clear identifiers for key UI elements. If you keep identifiers unchanged your automated tests require less maintenance.

⁹ Crowdtesting service providers include [Applause.com](https://applause.com), [PassBrains.com](https://passbrains.com), and [TestBirds.de](https://testbirds.de)

¹⁰ lookback.io/

¹¹ testdroid.com/testdroid/6336/get-the-superb-expertise-in-your-testingqa-team

Separate your code into testable modules. Several years ago, when mobile devices and software tools were very limited, developers chose to 'optimize' their mobile code into monolithic blobs of code, however the current devices and mobile platforms mean this form of 'optimization' is unnecessary and possibly counter-productive¹².

Provide ways to query the state of the application, possibly through a custom debug interface. You, or your testers, might otherwise spend lots of time trying to fathom out what the problems are when the application does not work as hoped.

Designing Test Environments

Test environments describe **where** we will perform the testing. It includes many facets. ISTQB defines test environment as: "An environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test."¹³.

A key skill is to understand and be able to create suitable test environments. Often these include additional software tools and utilities, for instance to be able to read and filter log files, to control the network behavior, and/or to represent systems the app depends on including third-party authentication, payment processing, and so on. Like craftspeople of old, we are responsible for choosing the tools we use. Sometimes we may choose to create our own tools, for instance to mock the behaviors of an application server, to be able to inject errors, and so on.

¹² To learn more about the reasons, read Google's blog: googletesting.blogspot.co.uk/2015/03/android-ui-automated-testing.html

¹³ istqb.org/downloads/send/20-istqb-glossary/186-glossary-all-terms.html

Mnemonics and Tours

Mnemonics and Tours help us focus on **what** and **how** while we are testing a mobile app. Both concepts are heuristics, often useful, but also fallible.

A couple of mnemonics to help test mobile apps are:

- **I SLICED UP FUN**¹⁴: Input (Test the application changing its orientation (horizontal/vertical) and trying out all the inputs including keyboard, gestures etc.), Store (Use appstore guidelines as a source for testing ideas), Location (Test on the move and check for localization issues), Interaction/Interruption (See how your app interacts with other programs, particularly built-in, native apps), Communication (Observe your app's behavior when receiving calls, e-mails, etc.), Ergonomics (Search for problem areas in interaction, e.g. small fonts), Data (Test handling of special characters, different languages, external media feeds, large files of different formats, notifications), Usability (Look for any user actions that are awkward, confusing, or slow), Platform (Test on different OS versions), Function (Verify that all features are implemented and that they work the way they are supposed to), User Scenarios (Create testing scenarios for concrete types of users), Network (Test under different and changing network conditions)
- **COP FLUNG GUN**¹⁵ summarizes similar aspects under Communication, Orientation, Platform, Function, Location, User Scenarios, Network, Gestures, Guidelines, Updates, Notifications.

¹⁴ kohl.ca/articles/ISLICEDUPFUN.pdf

¹⁵ moolya.com/blogs/2014/05/34/COP-FLUNG-GUN-MODEL

Karen Johnson provides helpful material on using heuristics and mnemonics for testing software at [karenlicolejohnson.com/wp-content/uploads/2012/11/KNJohnson-2012-heuristics-mnemonics.pdf](http://karennicolejohnson.com/wp-content/uploads/2012/11/KNJohnson-2012-heuristics-mnemonics.pdf).

Tours help you focus your testing, Cem Kaner describes a tour as "a directed search through the program. Find all the capabilities. Find all the claims about the product. Find all the variables. Find all the intended benefits. Find all the ways to get from A to B. Find all the X. Or maybe not ALL, but find a bunch."¹⁶ With the combination of different tours in different perspectives (see the I SLICED UP FUN heuristics) coverage and test depth can be chosen.

Examples of Tours¹⁷ include:

- **Configuration tour:** Attempt to find all the ways you can change settings in the product in a way that the application retains those settings.
- **Feature tour:** Move through the application and get familiar with all the controls and features you come across.
- **Structure tour:** Find everything you can about what comprises the physical product (code, interfaces, hardware, files, etc.).
- **Variability tour:** Look for things you can change in the application - and then you try to change them.

¹⁶ kaner.com/?p=96; also see developsense.com/blog/2009/04/of-testing-tours-and-dashboards/

¹⁷ from michaeldkelly.com/blog/2005/9/20/touring-heuristic.html

Personas

Personas can be used to reflect various users of software **who** will be expected to use the mobile app. They may be designed to reflect, or model, a specific individual or a set of key criteria for a group of users. Regardless of how they are created each persona is singular, not a group of people. Personas can be used to have a clear picture of various end-users to include so that representative tests are executed for those end-user. Various research material are available at *personas.dk*.

Testing on Various Devices

Some bugs are universal and can be discovered on any mobile device. Others, and there are plenty of them, are only happening on a subset of devices or in a certain environment (see paragraph above: Beware Of Differences). Rather than trying the never-ending task of a full testlab another solution is possible: start testing in production, for instance by involving end-users who use their own devices and combinations.

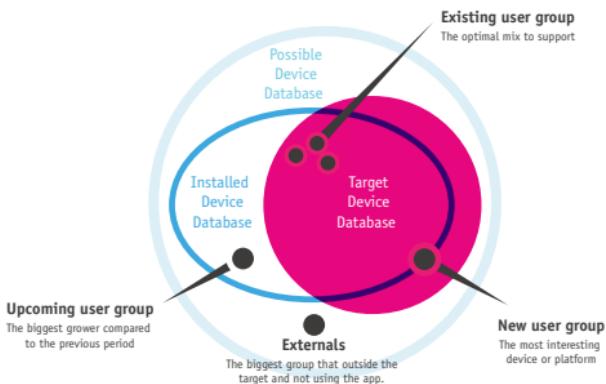
Physical and Virtual Devices

Physical devices are real, you can hold them in your hands. Virtual devices run as software, inside another computer. Both are useful hosts for testing mobile apps.

Virtual devices are generally free and immediately available to install and use. Some platforms, including Android, allow you to create custom devices, for instance with a new screen resolution, which you can use for testing your apps even before suitable hardware is available. They can provide rough-and-ready testing of your applications. Key differences include: performance, security, and how we interact with them compared to physical devices. These differences may affect

the validity of some test results. Beside the Android platform virtual devices you can use *GenyMotion.com*, a faster and more capable Android emulator, for instance, to control sensor values.

The set of test devices to use needs to be reviewed on an ongoing basis as the app and the ecosystem evolve. Also, you may identify new devices, that your app currently does not support, during your reviews. The following figure illustrates these concepts.



Ultimately your software needs to run on real, physical, phones, as used by your intended users. The performance characteristics of various phone models vary tremendously from virtual devices on your computer. So: buy, rent, beg, borrow phones to test on. A good start is to pick a mix of popular, new, and models that include specific characteristics or features such as: touch screen, physical keyboard, screen resolution, networking chipset, et cetera. Try your software on at least one low-end or old device as you want users with these devices to be happy too.

Here are several aspects to test explicitly on physical devices as the devices have a significant impact on these aspects:

- **Navigating the UI:** for instance, can users use your application with one hand? Effects of different lighting conditions: the experience of the user interface can differ in real sunlight when you are out and about. It is a mobile device – most users will be on the move. Rotate the screen and make sure the app is equally attractive and functional.
- **Location:** if you use location information within your app: move – both quickly and slowly. Go to locations with patchy network and GPS coverage to see how your app behaves.
- **Multimedia:** support for audio, video playback and recording facilities can differ dramatically between devices and their respective emulators.
- **Internet connectivity:** establishing an internet connection can take an incredible amount of time. Connection delay and bandwidth depend on the network, its current strength and the number of simultaneous connections. Test the effects of intermittent connectivity and how the app responds.

As mentioned earlier, crowdtesting can also help to cover a wide range of real devices, but you should never trust on external peoples' observations alone.

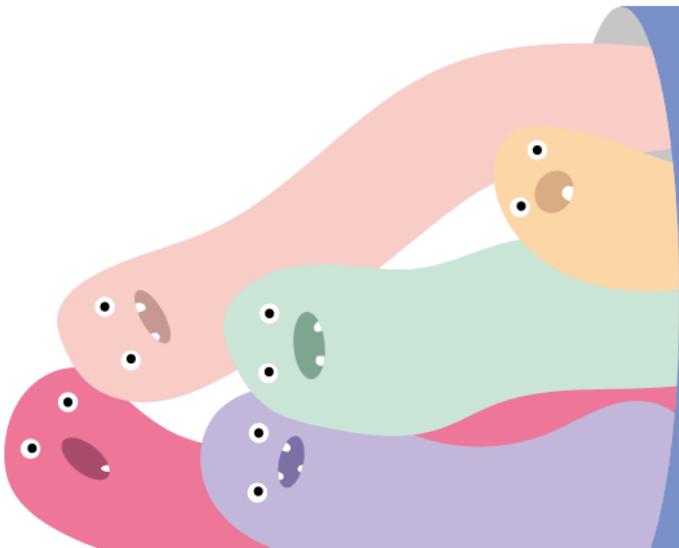
Remote Devices

If you do not have physical devices at hand or if you need to test your application on other networks, especially abroad and for other locales, then one of the 'remote device services' might help you. They can help extend the breadth and depth of your testing at little or no cost. Device farms are becoming commonplace, and are clearly strategic where Google and Amazon in particular now provide them.

You can also use commercial services of companies such as *SauceLabs.com*, *Testdroid.com*, *PerfectoMobile.com* or *Sigos.com* for similar testing across a range of devices and platforms. Some manufacturers brand and promote these services however you often have to pay for them after a short trial period. Some of the commercial services provide APIs to enable you to create automated tests.

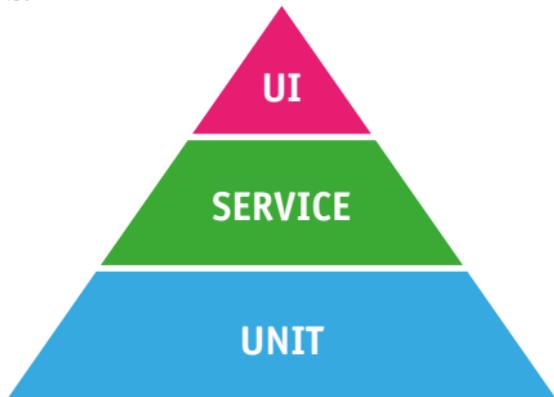
You can even create a private repository of remote devices, e.g. by hosting them in remote offices and locations.

Beware of privacy and confidentiality when using shared devices.



Test Automation

Automated tests can help you maintain and improve your velocity, your speed of delivering features, by providing early feedback of problems. To do so, they need to be well-designed and implemented. Good automated tests mimic good software development practices, for instance using Design Patterns¹⁸, modularity, performing code reviews, et cetera. To automate scripting and coding skills are needed. The level of skills is dependent on the chosen tool. Test automation tools provided as part of the development SDK are worth considering. They are generally free, inherently available for the particular platform, and are supported by massive companies. Test automation can be performed at different levels, see the automation pyramid figure below. It is a strategic choice what should be automated in the unit tests, what on the service or API level and what scenarios on the UI level of the application. The pyramid represents trust that is built up from the unit test to the higher levels. Multiple test levels are needed to prove that the app works.



¹⁸ en.wikipedia.org/wiki/Design_Patterns

GUI Level Test Automation

The first level of automation are the tests that interact with the app via the Graphical User Interface (GUI). It is one of the elixirs of the testing industry, many have tried but few have succeeded. One of the main reasons why GUI test automation is so challenging is that the User Interface is subject to significant changes which may break the way automated tests interact with the app.

For the tests to be effective in the longer term, and as the app changes, developers need to design, implement and support the labels and other hooks used by the automated GUI tests. Both Apple, with their XCTest framework¹⁹, and Android²⁰ underpin their test automation frameworks with Accessibility features incorporated into their platforms.

Some commercial companies have open sourced their tools, e.g. SauceLabs' appium²¹ and Xamarin's Calabash²². These tools aim to provide cross-platform support, particularly for Android and iOS. Other successful open source frameworks include Robotium²³ which now offers a commercial product - a test recorder. Several other tools have effectively disappeared, perhaps the industry is now maturing where only the stronger offerings survive?

¹⁹ developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/09-ui_testing.html

²⁰ developer.android.com/training/testing/ui-testing/uiautomator-testing.html

²¹ <https://github.com/appium/appium>

²² github.com/calabash

²³ github.com/robotiumtech/robotium

Service Level Test Automation

There is a lot of business logic implemented inside an API. Changes in this logic or in the backend system can be monitored with automated API tests. The focus of the test can be functional-regression but also reliability, performance and security. For functional regression testing a tool like Postman²⁴ is useful.

Several tools can help with API testing. They include Fiddler by Telerik²⁵ and Charles Proxy²⁶. Both enable you to view and modify network traffic between your mobile device and the network.

Unit Level Test Automation

Unit testing involves writing automated tests that test small chunks of code, typically only a few lines of source code. Generally, they should be written by the same developer who writes the source code for the app as they reflect how those individual chunks are expected to behave. Unit tests have a long pedigree in software development, where JUnit²⁷ has spawned similar frameworks for virtually all of the programming languages used to develop mobile apps.

²⁴ getpostman.com

²⁵ telerik.com/fiddler

²⁶ charlesproxy.com

²⁷ en.wikipedia.org/wiki/JUnit

BDD Test Automation

BDD is a Behavior-Driven Development²⁸ approach within the Test Driven Development family. The behavior is described in formatted text files that can be run as automated tests. The format of the tests are intended to be readable and understandable by anyone involved with the software project. They can be written in virtually any human language, for instance Japanese²⁹, and they use a consistent, simple structure with statements such as **Given, When, Then** to structure the test scripts.

The primary BDD framework to test mobile apps is Calabash for Android and iOS³⁰. Various others have not been developed or maintained in the last year and can be considered defunct for all but the most persistent developers. General purpose BDD frameworks may still be relevant when they are integrated with frameworks, such as Appium, that use the WebDriver wire protocol (a W3 standard)³¹.

Automation can also help the manual testing, for instance, to replace manual, error-prone steps when testing, or to reduce the time and effort needed, for instance, to automate a collection of screenshots. Developers can help testers to be more efficient by providing automated tools, e.g. app deployment via ADB³².

²⁸ en.wikipedia.org/wiki/Behavior-driven_development

²⁹ github.com/cucumber/cucumber/tree/master/examples/i18n/ja

³⁰ github.com/calabash

³¹ w3.org/TR/webdriver

³² thefriendlytester.co.uk/2015/11/deploying-to-multiple-android-devices.html

Testing Through The Five Phases of an App's Lifecycle

The complete lifecycle of developing a mobile app fits into 5 phases: implementation, verification, launch, engagement and validation. Depending on which phase(s) you are involved in, there are different tasks for a mobile app tester to be performed. For example, when joining a development team, the task can be the analysis of the error in the log files on a device. When joining a beta test phase, a task can be the analysis of usability tests results like recording movies.

Testing applies to each phase. Some of the decisions made for earlier stages can affect your testing in later stages. For instance, if you decide you want automated system tests in the first phase they will be easier to implement in subsequent phases. The five phases might suggest that they follow one after the other; this is not the case. Every step in the different phases provides the possibility to learn and improve. When testing the team learns both how good the mobile app product is and also about areas for improvement in how the app is produced. Mobile app development is a challenging complex, dynamic activity that does not go perfectly the first time, therefore, teams should incorporate an improvement cycle so they can learn and actively improve what they do, see Mobile improvement model³³.

Phase 1: Implementation

This includes design, code, unit tests, and build tasks. Traditionally testers are not involved in these tasks; however good testing here can materially improve the quality and success of

³³ <https://improvement.polteq.com/en/ti4mobile/>

the app by helping us to make sure the implementation is done well.

In terms of testing, you should decide the following questions:

- Do you use test-driven development (TDD)?
- Help review designs on what are the main, alternative and negative user flows
- Which test data do you use to validate the user flows?
- Will you have automated system tests? If so, how will you facilitate these automated system tests? For instance by adding suitable labels to key objects in the UI.
- How will you validate your apps? For instance, through the use of Mobile Analytics? Crash reporting? Feedback from users?

Question the design. You want to make sure it fulfills the intended purposes; you also want to avoid making serious mistakes. Phillip Armour's paper on five orders of ignorance³⁴ is a great resource to help structure your approach. And again: do read the first chapters in this guide to learn more about this important phase in app development.

Phase 2: Verification

Review your unit, internal installation, and system tests and assess their potency: Are they really useful and trustworthy? Note: they should also be reviewed as part of the implementation phase, however, this is a good time to address material shortcomings before the development is considered 'complete' for the current code base.

³⁴ www-plan.cs.colorado.edu/diwan/3308-07/p17-armour.pdf

For apps that need installing, you need ways to deploy them to specific devices for pre-release testing. Based on your test strategy you can decide on which phones, platforms, versions, resolutions are in scope of testing and support.

System tests are often performed interactively, by testers. You also want to consider how to make sure the app meets:

- Usability, user experience and aesthetics requirements
- Performance, particularly as perceived by end users³⁵
- Internationalization and localization testing

Phase 3: Launch

For those of you who have yet to work with major app stores be prepared for a challenging experience where most aspects are outside your control, including the timescales for approval of your app. Also, on some app stores, you are unable to revert a new release. So if your current release has major flaws you have to create a new release that fixes the flaws, then wait until it has been approved by the app store, before your users can receive a working version of your app.

Given these constraints, it is worth extending your testing to include pre-publication checks and beta tests of the app such as whether it is suitable for the set of targeted devices and end-users. The providers of the main platforms publish guidelines to help you test your app will meet their submission criteria. These guidelines may help you even if you target other app stores and can be used as a checklist during the implementation phase.

³⁵ A relevant performance testing tool is ARO (Application Resource Optimizer) by AT&T: developer.att.com/application-resource-optimizer, open source project at github.com/attdevsupport/ARO

App Store Guidelines

Apple	developer.apple.com/app-store/review/guidelines/
Android	developer.android.com/develop/quality-guidelines/

Phase 4: Engagement

This includes search, trust, download and installation. Once your app is publicly available users need to find, trust, download and install it. You can test each aspect of this phase in before and in production. Try searching for your app on the relevant app store, and in mainstream search engines. On how many different ways can it be found by your target users? What about users outside the target groups - do you want them to find it? How will users trust your app sufficiently to download and try it? Does your app really need so many permissions? How large is the download, and how practical is it to download over the mobile network? Will it fit on the user's phone, particularly if there is little free storage available on their device? And does the app install correctly? - there may be signing issues which cause the app to be rejected by some phones.

Phase 5: Validation

This includes payment, usage and user feedback. As you may already know, a mobile app with poor feedback is unlikely to succeed. Furthermore, many apps have a very short active life on a user's phone. If the app does not please and engage them within a few minutes it is likely to be discarded or ignored. And for those of you who are seeking payment, it is worth testing the various forms of payment, especially for in-app payments.

Consider finding ways to test the following as soon as practical:

- Problem detection and reporting. These may include your own code, third-party utilities, and online services.
- Mobile Analytics. Does the data being collected make sense? What anomalies are there in the reported data? et cetera?
- Feedback. For all the flaws and limitations we can glean a lot from feedback users provide including their sentiments, feature requests, bugs, and clues we can use to improve our testing.

You can read more about Mobile Analytics and Feedback in this book.

Learn More

Testing mobile apps is becoming mainstream with various good sources of information. Useful online sources include:

- katrinatester.blogspot.de/2015/08/mobile-testing-pathway.html - A comprehensive and well presented set of possible steps for testing mobile software.
- github.com/julianharty/testing-heuristics - An online open source project to learn more about testing heuristics for mobile apps.
- enjoytesting.files.wordpress.com/2013/10/mobile_testing_ready_reckoner.pdf - Contains short, clear testing ideas with examples, mainly for Android devices
- developers.google.com/google-test-automation-conference/ - The annual Google Test Automation Conference (GTAC) often includes several presentations on testing mobile

apps. These are recorded and available free of charge, worth watching.

- [*testdroid.com/blog*](http://testdroid.com/blog)- A fertile blog on various topics including testing mobile apps. They also have a series on testing mobile games³⁶.
- [*genymotion.com/blog/android-testing-showdown/*](http://genymotion.com/blog/android-testing-showdown/) - A useful guide on selecting the best devices to test on.
- [*appqualityalliance.org/resources*](http://appqualityalliance.org/resources) - The official App Quality Alliance AQuA website including their useful app testing guidelines.

A good place to start learn testing mobile apps is reading books like:

- [*sensible.com/*](http://sensible.com/) - Steve Krug has written several immensely popular books that cover low-cost d-i-y usability testing. Various chapters are available online, his work is well worth reading and much applies to testing mobile apps.
- [*leanpub.com/testmobileapps*](http://leanpub.com/testmobileapps) - Tap Into Mobile Application Testing, by Jonathan Kohl provides help and advice on ways to test mobile apps.
- [*handsonmobileapptesting.com/*](http://handsonmobileapptesting.com/) - Hands-on Mobile App Testing, by Daniel Knott. A well-written book on various aspects of testing mobile apps. A sample chapter is available from the web site.

And of course do not miss the platform-specific articles in this guide, especially the mobile web chapter to gain deeper insights into mobile testing.

³⁶ testdroid.com/testdroid/7790/best-practices-in-mobile-game-testing



Mobile Analytics

Introduction

Over 80 years ago pilots learned instruments could help them fly better and their skills in using instruments could save them particularly if they needed to fly 'blind'¹. Of course, the quality of the instruments was vital too. Today, many app developers incorporate software that can help them know how their apps behave when they are being used. Similarly, a range of instrumentation exists to help us learn and understand more about how the app is being used and how it is performing for our users. This chapter introduces mobile analytics, crash analytics, and heatmaps. These can be combined with each other and complement other sources of information, including app store ratings, crowd testing, and usability studies.

Data from mobile analytics can help many aspects of our work, including the business, social, operational, and technological aspects. The data captured can be used to target your work and reduce inefficiencies. You would be in good company, according to SafeDK analytics is the most popular library in mobile apps and at least 82% of the Android apps they scanned incorporate analytics².

There is an incredible richness in the mobile galaxy where your software can be used on a wide range of devices that exhibit significant differences in performances and behaviors.

¹ csobeech.com/files/Blind-Or-Instrument.pdf

² May 2017 Mobile SDKs Data Trends In the Android Market: mobile-sdk-data-trends.safedk.com/full-report-May-2017 (free registration is required to download the report)

Researchers discovered battery drain varied by 3x when their app was used on devices with similar hardware specifications and, amongst other things, they discovered an app used custom code to reduce the screen's brightness when running on Kindle Fire's to improve battery life by 40% and significantly increased the session durations as a result.

Analytics can also teach us ways to improve the ways we develop and test the software.

In all the excitement we need to remember to protect user's privacy and respect their preferences and expectations. The effects of mobile analytics can upset users by consuming valuable resources or abusing sensitive information about the user and their use of the app.

Deciding What To Measure

What would you like to measure to understand how the app is being used? Some suggestions are:

- **User Interfaces:** How people interact with the app, and particularly the GUI.
- **Key usage events:** What the users do; for instance, when they use various internal functions or when users launch social networking from your app.
- **Business-centric events:** Any interaction by the user that generates revenue for you. How often do your users purchase the premium version of the app or other items offered within your software? When do they cancel orders or discard their shopping cart before checking out?
- **Application-centric events:** Performance, usability, reliability, and other data about the behavior of the app.

Once you have defined your main areas of interest, you will

need to design the analytics measures, for instance, what data elements need to be reported. Invest time in deciding and designing what data is key to capture, when and how. Various guides are available, including Facebook's App Events Best Practices Guide³. Also, the Mobile Games chapter in this guide includes a useful example of deciding what data to collect.

Create meaningful names for your interaction events so you can easily and correctly remember what they measure. For each event you want to record, decide what elements it needs to include. Consider how the data will be used once it has been gathered, for instance, sketch out typical reports and graphs and map how the various data elements will be processed to generate each report and graph.



³ developers.facebook.com/docs/app-events/best-practices

Analytics for Layers of a Mobile App

Conceptually an app consists of several layers that build on each other. The topmost layer is the UI which communicates with the user. Virtually all apps include a graphic UI (GUI) which is displayed on the screen of the device. There may be other UIs, for instance, to capture movement, audio, and video. The next layer contains the logic, what the app **does**. Often there is also some sort of communications layer. And at the lowest level, there is the physical device with the operating system, or platform, installed, which supports and provides the runtime for the app.

There are various types of analytics available, they can overlap to a certain extent. There are various types of analytics available, they can overlap to a certain extent. Importantly, users do not tend to perceive the app as layers, it is simply the app. Their perceptions and their user experience can be affected by any of these layers (and by other sources such as feedback and reviews by other users).

Perception	App-Store, in app, and social feedback
GUI	GUI analytics (heatmaps)
App	In-app mobile analytics
Platform	Instrumentation

Layers of an app

The most popular form of GUI analytics is based on heatmaps, they are particularly well suited to capturing data on how the GUI is being used. Heatmaps are enabled by incorporating software into an app to track all the user-interactions with the app's GUI. There are tons of commercial options available, Appsee provides a particularly polished service and offers many free resources including e-books⁴ on heatmaps and related topics.

In isolation, heatmap data can be used to track individual user "journeys" through the GUI. In aggregate, various analytics related to User Experience (UX) can be inferred from the data, including problematic areas of the GUI. There does not appear to be equivalent services to capture other forms of input, so you may need to write your own code if you want/need to gather data about other UIs.

In-app mobile analytics suits the application logic, it may also record some details of the layer above - the UI - and the layer below - the device. Crash analytics are also useful to record and report on when and where crashes (also known as exceptions) occur in the field, so we can find and fix flaws in how the app behaves. Both Android and iOS record crashes. Users need to permit these to be shared with the developers. Alternatively, crash reporting can be incorporated into an app and sent automatically. Some mobile analytics libraries can be configured to also record crashes, for instance, in Google Analytics V4⁵.

The platform's behavior can be measured in the lab using instrumentation and specialist software tools. Measuring the behavior of users devices is harder partly because of security

⁴ appsee.com/ebooks

⁵ developers.google.com/analytics/devguides/collection/android/v4/exceptions

enforced by the platform. It is unlikely our apps will record information about the platform directly, however some limited aspects may be available such as free storage, running apps, and resource utilization. What is available and how to obtain the information is platform specific.

Choosing which Analytics to use

We have plenty of options competing for our attention. There are four main sources:

- **Platform-default:** these are bundled with the platform and incorporated into the respective app store's development console. For single platform apps these may be the default choice since they are likely to be well documented and integrated into the development and release process.
- **Commercial:** these are provided by commercial companies. They may be free-of-charge particularly for small volumes of usage. However, the services generally need to be paid for once volumes increase and to unlock additional features.
- **Opensource:** there are several opensource mobile analytics offerings that include source code for both the client and the server. Examples include Count.ly⁶ and Piwik⁷. These allow the software to be customized and tailored. They can also be hosted on servers you choose and provide significant flexibility.
- **In-house:** it is possible to create in-house mobile analytics software. Various companies have done so. Reasons to develop in-house implementations include: flexibility,

⁶ count.ly/community-edition/

⁷ github.com/piwik

performance, and security. Cost may also be a factor compared to commercial offerings.

Our choice is likely to depend on several, sometimes competing, factors. Here are some suggestions to consider: cost, flexibility, performance, control, access to data, and cross-platform consistency for organizations and teams who need to support more than one platform, app or implementation. Additional considerations include the richness and flexibility of the API and the control that is available to the developers and the users (in case you want to allow users to decide which analytics data are collected and transferred).

Consider several of the potential solutions before committing to any of them. Discover what other apps use and why. Read documentation and example code to see how easily you can implement them into your app, and check the legal agreements, including privacy. Then pick at least one of them so you can experiment with implementing mobile analytics into your app. By integrating their code, you are likely to learn much more about what you would like to achieve by using mobile analytics in your app, and how mobile analytics works in practice.

Two providers are well worth studying. Segment.io⁸ abstracts a wide range of mobile analytics offerings. Their open source code⁹ reduces the effort needed to adapt to different analytics providers. Count.ly¹⁰ provide open source implementations of their server as well as of their client libraries, and they encourage you to create a complete test environment to evaluate their product.

For multi-platform apps, you may want consistency across

⁸ segment.io

⁹ github.com/segmentio

¹⁰ count.ly

each platform. Otherwise, you may be trying to compare dissimilar, or even disparate, data sets - particularly if different mobile analytics solutions are used for the various platforms. Consider picking a common solution that supports every platform you want to launch your app on.

Implementation Considerations

There are a wide range of topics to consider when implementing and integrating mobile analytics. Broadly they are:

- **Costs:** There are various costs. These include implementation, financial, operational and privacy (as they collect data from users of how they are using the app). These costs need to be considered and justified and only implemented where the value significantly exceeds the costs and where the actual and potential costs can be mitigated.
- **Testing and calibration:** Our work, and the services and infrastructure we use, all need to be tested and calibrated. Do not blindly trust or assume the system and the numbers they present are complete and accurate, they often are not. Read on for more information.
- **Customization:** including augmenting and repurposing.
- **Integration into our microcosm:** Integration can help improve how we work as well as the value of the data and information we obtain. What are the integration options, for instance with communications tools such as IRC and Slack? And are API's available so we can query and retrieve data, reports, and analysis?
- **Time, latency, time zones, cut-offs:** Time-related aspects can reduce the completeness and the value of the data, reports, and insights.

Many mobile analytics solutions will automatically record and report data elements to the server. It is worth checking what these elements are, how and when they are reported, and how they are formatted. Then you can decide whether you want to use and rely on these automatically-reported elements.

Customization

Custom event tags augment predefined events, and many mobile analytics solutions provide ways for your app to generate them. You may need to format the custom event messages. If so, pay attention to an encoding of the elements and separators. For instance, they may need to be URL encoded¹¹ when they are sent as REST messages.

You may want to consider how often the app should report events to reduce the risk of flooding the available capacity of the analytics system, which might affect the reliability and accuracy of the delivered analytics data. Localytics has some good integration tips online¹². One method to reduce the volume of data processed by the analytics solutions is called sampling. Adam Cassar published an interesting blog post on this topic¹³.

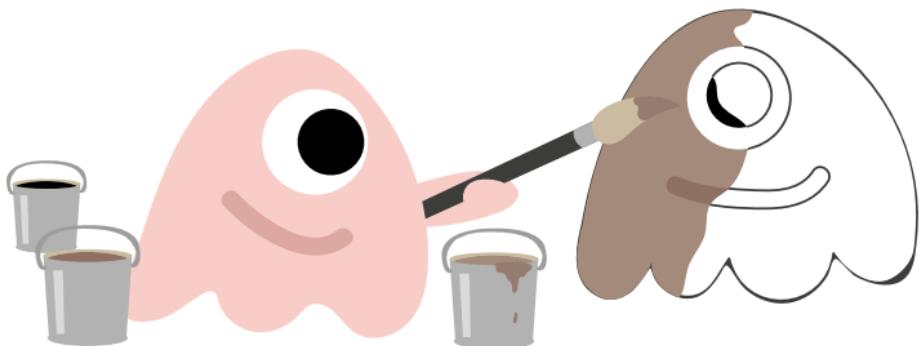
¹¹ en.wikipedia.org/wiki/Percent-encoding

¹² support.localytics.com/Integration_Overview

¹³ periscopix.co.uk/blog/should-you-be-worried-about-sampling

Time-related topics

There is a lag from when an app sends an analytics event to when the information is processed and made available to you. The lag, or latency, varies from near 'real-time' to many hours. You, and your business sponsors, need to decide how long you can afford to lag real-time events. Also, remember to address globalization issues such as the timestamp of each element. Does the app detect the time of an event according to the device's location, the device's settings or does it use a global time like UTC time? Some providers, including Google, have cut-offs for when data arrives in order for the data to be reported on. We may have limited influence on when an app transmits the data, so the 'problem' is hard to fix and it is therefore important to remain aware that reports may lack data from some users for various reasons.



What can go wrong?

The road to hell is paved with good intentions, there are many things that can go wrong when implementing analytics in mobile apps. Some of the most critical include:

- **Uncalibrated results:** blindly trusting the data can lead to a maelstrom of problems. The result can be inaccurate and misleading which causes knock-on problems when you use these results to manage the business and your work. A good practice is to test the analytics implementation at the outset, starting with no users, then one, before testing with more users. Look at latency, accuracy, and reliability of the recorded data.
- **Betraying trust:** Users implicitly trust apps to behave nicely on their mobile devices. However, apps or the SDKs may accidentally or deliberately break that trust, for instance by tracking users, recording and then using sensitive data etc. Try not to hide behind click-through agreements which we knew few people read and even fewer understand. Instead, make sure your app and any analytics libraries you use behave nicely and "Do as you would be done by and do not snoop."
- **Handing over the jewels:** The analytics data is sufficiently valuable that some companies provide mobile analytics services free-of-charge. As a minimum, make sure that you do have sufficient rights and access to the data that is collected by the analytics software. This is especially relevant when using third-party libraries and services.

Be aware, some mobile analytics solution providers may use data reported by your app, and they may provide and sell it to others. They may control the life of that data, which means

they could make it inaccessible to you. Conversely, they may preserve and use it long after you have retired your app. If there is personally identifiable information in the data, there may be additional legal and privacy implications.

SafeDK are a recent startup who focus on the behavior of SDKs, including mobile analytics. SDKs added to apps can adversely affect the performance, security and reliability of the app in addition to other problems and concerns. SafeDK's blog¹⁴ discuss the concerns and provide advice on how to select SDKs by understanding the behaviors they exhibit.

Remember to explain to the end-users that the app is designed to record and share information about how the app is being used, ideally in your terms and conditions. You may need or want to enable users to decide if they want their use of the app to be tracked. If so, make it easy for the user to control the settings; and consider providing the user a way to access the recorded data, delete it, or contact the analytics solution provider.

Providers of third-party libraries seem to have a range of attitudes to privacy. Some claim the privacy of users is paramount and stresses the importance of not tracking users. Google Analytics clearly prohibit tracking personally identifiable information in their terms of service¹⁵. Others provide examples, including snippets of source code, that demonstrates how to record clearly personally identifiable data. For instance, KISSmetrics provides the following code snippet¹⁶:
[identify:@name@email.com]. Mixpanel provides an example of how to track specific users¹⁷.

¹⁴ blog.safedk.com

¹⁵ google.com/analytics/terms/us.html

¹⁶ support.kissmetrics.com/article/show/24034-objectivec-ios-library

¹⁷ mixpanel.com/activity-feed/

There are several places to learn more about privacy and ethics of working with data related to users, e.g.:

- Jeff Northrop's blog post on mobile analytics¹⁸
- Kord Davis' book "Ethics of Big Data" (O'Reilly, 2012)¹⁹

Learn More

We hope this chapter has whetted your appetite to learn more about mobile analytics. Here are some places to start your ongoing research:

- Various articles by Michael Wu or Lithium Technologies. A good place to start is the article "Are Your Big Data Analytics Actionable?"²⁰
- Capturing Mobile Experience in the Wild: A Tale of Two Apps²¹, a study from the University of Wisconsin highlighting the importance of application-centric analytics based data collected on 1M+ users over 3 years.
- The Beginner's Guide To App Analytics²², available as a free download.
- The Mobile Analytics Playbook²³ includes material on using mobile analytics to help improve testing of mobile apps.

¹⁸ jnorthrop.me/privacy-considerations-with-mixpanel-people-analytics

¹⁹ available at shop.oreilly.com/product/0636920021872.do

²⁰ community.lithium.com/t5/Science-of-Social-blog/Are-Your-Big-Data-Analytics-Actionable/ba-p/129029

²¹ static.googleusercontent.com/media/research.google.com/en//pubs/archive/41590.pdf

²² info.localytics.com/download-beginners-guide-to-app-analytics

²³ themobileanalyticsplaybook.com



Collecting & Understanding User Feedback

Feedback from our users can help us develop apps users want to use and improve the apps we have. Users also pay attention to the feedback of their peers, particularly how well apps are rated in the app store. When we learn how to leverage their feedback and when we respond to their feedback we show users we care and we increase the chances of retaining these users and encouraging more users. We may also reap the rewards in terms of achieving business objectives such as increasing revenues.

App stores have established themselves as the default place where users write and share their feedback so it is important to learn how to make sense of the feedback. App stores include tools for doing so; there are many additional services and tools available which are worth considering as part of our working practices. In-app feedback can increase the quality and quantity of feedback we receive, as well as enable us to respond to gripes and issues directly. By responding directly we can reduce negative feedback in the app store which may adversely affect downloads, usage, and the perceptions of other current and potential users. As volumes of feedback increase it is vital to find ways of managing the volumes and remaining alert and responsive, particularly as user feedback may be an early indicator of problems with an app.

User feedback can be complemented with usability testing and with techniques such as mobile analytics to provide additional insights. Usability testing can be performed by current users and by professional testers and researchers. Do read the chapters about app concepts and about UI/UX design

in this book to learn more about usability testing. Mobile analytics provides automated software-oriented feedback, see the dedicated article in this guide to learn more about your options in that area. This chapter focuses on technologies that help you gathering and understanding human feedback.

Feedback Mechanisms

Lots of options are available to obtain feedback in addition to the services app stores provide. We will cover the main ones next.

App Store Ratings and Feedback

App stores provide a public forum for users to rate apps and provide written comments. As many of us know, apps with low ratings are much less likely to be downloaded by users. Furthermore, app stores seem to use the app rating as a factor for deciding the priority of including an app in search results and even promoting some of the more popular apps.

Microcosms have matured around app stores, particularly to help development teams process and interpret user feedback. Estimates vary on the percentage of users who provide reviews, some apps do not receive any, many are in the range of 0.1% to 1%, and a few popular apps receive ratings and reviews from up to 10% of their userbase. Some of the reviews may be fake, which means that they do not come from actual users of the app - instead they may be rogue or paid-for. Beware of being tempted into providing reviews or sponsoring reviews for your apps or those of your competitors - app stores have harsh penalties, for instance Apple have removed apps and closed the

developer's account when they believe the developer gamed the feedback and ratings¹.

Initially, we might consider the priority of the star ratings on a linear scale, where 1-star is the lowest and 'worst' rating. However, work by Shopify and others discovered that users gave 2-star ratings for the most serious and important problems. They published their findings in a paper called What Do Mobile App Users Complain About?².

In-app Feedback

There are numerous ways of building in feedback channels into your app. The most basic one is of course offering e-mail feedback e.g. from your imprint screen, but you should not expect a lot if this is the only channel you are offering.

Instead, you should consider asking users pro-actively to share their perceptions with you. You can choose to create your own in-app feedback mechanisms or use existing offerings. Vendors like Apptentive claim they offer numerous benefits including increasing the quantity of feedback as well as user-satisfaction³. Feedback can range from basic, such as multi-choice questions and answers, to rich and complex where the user decides what feedback to provide and how. Some marketing products, such as *insert.io* provide support for questions and answers.

¹ appleinsider.com/articles/17/05/24/dash-returns-to-app-store-seven-months-after-review-system-manipulation-accusations

² doi.ieeecomputersociety.org/10.1109/MS.2014.50

³ apptentive.com/why-apptentive/

Comprehensive user surveys can also be built into your app by using third-party integrations. Their branding may be presented, alternatively there may be ways to configure their user interface to use your branding and to blend their UI with those of the app, for instance SurveyMonkey offers customization of the UI⁴ and offer tips on ways to integrate surveys⁵.

A good place to start learning about in-app surveys is quora.com/What-is-the-best-in-app-survey-SDK.

Social Media

Social media includes services such as Facebook and Twitter, and more recently social video sites such as YouTube, where people share their thoughts, impressions and feelings online with various social groups such as friends, colleagues and acquaintances. They may share things publicly, where anyone can view the shared materials. Some feedback relates to mobile apps. A good example is Facebook's own iOS app which drained the battery of mobile apps. One of the Facebook engineering managers, Ari Grant, explains the causes and the fixes in an online article⁶. Interestingly, some of the subsequent comments indicate the problem may have returned several releases later.

You may want to consider some kind of social media monitoring even if you do not have active channels for your app on social media. Monitoring can notify you, particularly when people are complaining about your app or service, and enable you to respond promptly and professionally.

⁴ surveymonkey.co.uk/mp/mobile-sdk/

⁵ surveymonkey.com/blog/2016/03/03/7-tips-for-in-app-surveys-and-collecting-responses-with-the-mobile-sdk/

⁶ facebook.com/arig/posts/10105815276466163

Working with Feedback

In his book "The Art of the App Store"⁷, Tyson McCann defines the following main steps when dealing with user feedback:

- Categorize feedback, for instance, to identify constructive criticism. Vague comments can be filtered out at this stage too.
- Convert feedback into actionable tasks, including suggested fixes and assigning priorities.
- Update the app with various fixes and improvements.
- Add release notes so users can easily read the good news about the improvements and install the updates promptly.

If you have a developer's account you can access and respond to reviews for your apps directly. For Android apps on the Google Play Store, Google provides downloads of various data, including ratings and reviews. They also offer API access (and free mobile apps).

Various commercial services offer to reduce the effort of collecting and analyzing app store data, for instance by providing API access. They may offer the ability to download and analyze reviews for other apps, for instance if you would like to monitor reviews for similar apps. They have various strengths such as sentiment analysis, for example *AppBot.co* and workflow integration, for example *Appfollow.io* offers integration with Slack. You can also roll-your-own analysis if you are willing to write software and maintain it as the data sources change.

⁷ Tyson McCann: The Art of the App Store: the Business of Apple Development, ISBN 978-0-470-95278-8

Data Mining

We may need to deal with lots of text on an ongoing basis, for instance, some popular apps receive many 1,000's of pieces of feedback each day, which is expensive to process without software. Data mining can help process vast amounts of data, and help us to identify trends, and discover fresh insights from the feedback we receive. Data mining is a rich research topic, and there is even a dedicated academic research project called UCLappA⁸.

There is also a friendly free introduction to data mining written for programmers online⁹.

Dealing with Inconsistencies

In some instances, the user rating and the remarks may contradict each other. People may not understand the scoring system, while app stores consider 5 stars the highest rating, users may assume 1 star is the highest rating instead. Alternatively, the ratings may seem to be almost random.

Thankfully, the major app stores now offer the possibility to respond to reviews. You can use this mechanism and ask users to correct their rating in case you think it is based on a misunderstanding.

Rogue Feedback

Rogue feedback is feedback deliberately submitted to affect the overall rating of an app. The feedback may be aimed at inflating or deflating the rating. Some people try to inflate the ratings for various reasons, for instance, to try and get their app promoted by others. Others may target the apps of

⁸ www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/home.html

⁹ guidetodatamining.com

competitors to drag down the ratings and adversely affect their attractiveness, reduce the number of downloads, etc.

Like SPAM email, some rogue feedback may be easy to detect and either report or filter out from our analysis, for instance if there are lots of identical reviews with the same text, etc. Others may be very poorly written. Some app stores are working hard to reduce rogue feedback. Analogously Amazon has removed lots of fake feedback and is suing various people who wrote the feedback¹⁰.



¹⁰

<http://www.wired.co.uk/news/archive/2015-10/19/amazon-fake-reviews-legal-action-fiverr>

Interpreting and Inferring

As we know from personal experience, the words people write are not necessarily what they mean or exactly what they want to express. They may not spell everything correctly and the grammar may be poor. Also, there are nuances in interpreting writing. For instance, in texting using a period to end a sentence may be considered insincere¹¹.

Feedback can be in various languages. App stores may use automatic translation to help us read and interpret, none-the-less our understanding will be incomplete and imperfect. Finding native language speakers can help us work more effectively with the users who provide feedback in languages foreign to us. Emotional and sentiment analytics extend feedback in several dimensions, we cover them shortly.

Emotional Analytics

Emotions can be highly relevant for some apps, and the emotions of users can affect their perceptions of an app and any feedback they provide. Apps are available that claim to measure a user's emotions using visual¹² and auditory¹³ sources of data. They are early indicators of what might be not only possible but useful, in the near future.

Sentiment Analytics

Sentiment analytics processes what users communicate to determine their feelings, their sentiments, and their wish to communicate to others. Those may be us, our organization,

¹¹ lifehacker.com/ending-text-messages-with-periods-can-make-them-seem-in-1747411231

¹² affectiva.com/solutions/mobile

¹³ beyondverbal.com

their friends or anyone who discovers what the users have communicated. By paying attention to user's sentiments we learn what they like and dislike so we can do more of what they like and address what they do not.

Designing for Feedback about the Mobile App

There are several key aspects to designing feedback. Here we cover what to ask, when to ask and how to ask.

What to ask

Designing trustworthy responses involves more than asking Yes/No questions. Yes does not always mean yes to the question, and No does not always mean no. They may simply be "Go Away!", that is, stop asking me questions! Insert's company blog¹⁴ gives a nice example: When you ask something like "Are you familiar with the new 'Touch to Pay' button?" a user probably expects many follow-up questions asking for further clarification if he answers No. So he might answer with Yes just to get rid of the questioning.

When to ask

Asking for feedback e.g. via a pop-up is a simple way to get started. However, when designing feedback try not to obstruct or frustrate users, for instance, do not block them midway through a process, a game level, etc. - at least do not if you have any hope of getting positive feedback.

Savvy app developers design their apps and their systems to encourage public feedback when people are more likely to provide positive feedback and direct feedback if the feedback

¹⁴ insert.io/in-app-surveys/

is likely to be negative or critical. Users may be in a good mood after successfully completing an action, for instance, a level in a game, or a purchase on an e-commerce site. A discreet request for feedback on completion may be well received by some users, and others can simply continue with whatever else they want to do. Some designers may include prepared feedback statements in the hope of encouraging users to use them.

How to ask

When implementing a feedback mechanism into your app, it may be wise to ask users how they feel **before** asking them to share their views in the app store. You may want to route negative feedback away from the app store, where it would be public and adversely affect the app's rating on the store. A dialogue offering the following options would be one way of achieving this:

1. **You like this app? Please add a review now!** (leads the user to provide feedback on the app store)
2. **Write a review later** (closes the dialog)
3. and here is the detour: **You do not like this app? Please let us know why!** (does NOT open the app store dialog, instead open the email client, or in-app feedback, so that feedback goes to you directly and is not public)

In other words "**Please tell us first; and then once you're happy - tell others (on public fora).**"

Apple have revised their policy on rating reviews and placed strict requirements on developers¹⁵.

¹⁵ see 9to5mac.com/2017/06/09/app-rating-custom-prompts-app-store-banned

Responding to Feedback

For many organizations being able to respond accurately and rapidly enables them to improve not only the user's perceptions but also the perceptions of many more people either directly or indirectly (for instance through the user telling others about the good things we have done).

Most app stores offer the opportunity to respond to reviews for their apps. None-the-less, according to Appbot¹⁶ on Google Play 97% of reviews go unanswered. Perhaps you can out compete many of your competitors by answering all your reviews? In 2017 Apple added this option as well and provide useful advice on how to do so¹⁷. By responding (especially for bad reviews), app providers can help their users and increase engagement. If you help users solve an issue they encountered, they can (and hopefully will) revert their poor rating and give your app more stars than in their initial review. But be careful: Do not respond with general templates, otherwise you may give users the impression that you are not treating their review with sufficient respect (e.g. "Thanks for your feedback, we will look into this."). Instead, let the user know they are special and their feedback is valuable by incorporate details they provided (including their name if it is available).

Some project teams may decide to incorporate commercial feedback services, such as HelpShift¹⁸ or UserVoice¹⁹, to help them proactively manage feedback by users of their app.

¹⁶ blog.appbot.co/97-of-google-play-app-reviews-go-unanswered/

¹⁷ developer.apple.com/app-store/responding-to-reviews

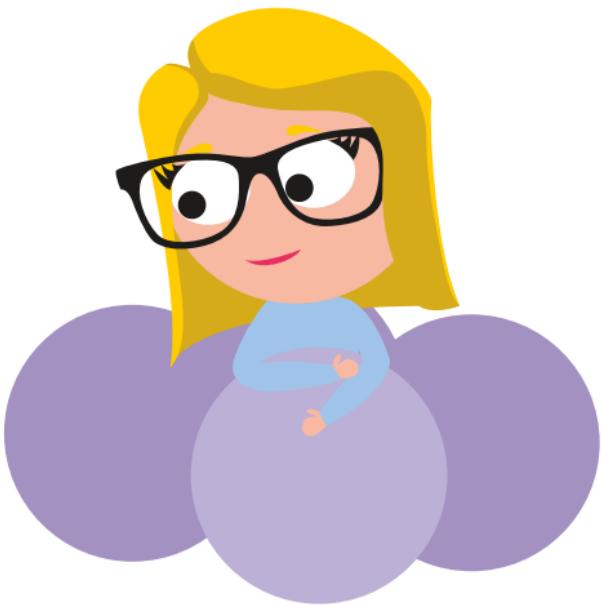
¹⁸ helpshift.com

¹⁹ uservoice.com

Learn more

- A PDF explaining methods of how to automatically analyze tens of millions user ratings and comments in mobile app markets: "Why People Hate Your App - Making Sense of User Feedback in a Mobile App Store", available at cs.cmu.edu/~leili/pubs/fu-kdd2013-wiscom.pdf
- An excellent read on that includes material on app store reviews and ratings is the "App Quality Book" by Jason Arbon²⁰.
- A high-level overview on 5 methods of collecting user feedback on GetApps blog: lab.getapp.com/collect-customer-feedback-customer-centric-company
- A list of the 20 most popular survey offerings (not mobile-centric) can be found at: capterra.com/survey-software/#infographic

²⁰ appqualitybook.com





Monetization

Finally, you have finished your app or mobile website and polished it as a result of beta testing feedback. Assuming you are not developing as a hobby, for branding exposure, et cetera, now it is time to make some money. But how do you do that, what are your options?

In general, you have the following monetization options:

- **Pay per download:** Sell your app per download
- **In-app payment:** Add payment options into your app
- **Mobile advertising:** Earn money from advertising
- **Sponsorships:** Receive money for each user signing up to your sponsor
- **Revenue sharing:** Earn revenue from operator services originating in your app
- **Indirect sales:** Affiliates, data reporting and physical goods among others
- **Component marketplace:** Sell components or a white-label version of your app to other developers
- **App platform subscriptions:** Create small apps and lease them to businesses

When you come to planning your own development, determining the monetization business model should be one of the key elements of your early design as it might affect the functional and technical behavior of the app. "Five strategies to monetize your mobile app"¹ is an excellent article on how to design the financial aspects so they do not annoy users or lose the revenues you hoped to receive.

¹ medium.com/@signored/dont-fall-below-the-app-poverty-line-9b800a214e4a

Pay Per Download

Using pay per download (PPD) your app is sold once to each user as they download and install it on their phone. Payment can be handled by an app store, mobile operator, or you can set up a mechanism yourself. Once the most popular and profitable monetization method, today it is only used by 5% of all mobile users, yet generating about 37.8% of all direct revenue (PPD, mobile ads or in-app purchase) according to Invespc²

When your app is distributed in an app store, the store will handle the payment mechanism for you. In return the store takes a revenue share (typically 30%) on all sales. In most cases stores offer a matrix of fixed price points by country and currency (\$0.99, EUR 0.79, \$3 etc) to choose from when pricing your app.

Payment for downloaded apps is generally handled in one of two ways: operator billing or credit-card payments.

Operator billing enables your customers to pay for your app by just confirming that the sale will be charged to their mobile phone bill or by sending a Premium SMS. In some cases, operator billing is handled by an app store (such as Google Play, which supports operator billing for a number of carriers around the world). In other cases, it can be implemented directly by the developer.

Each operator will take a revenue share of the sale price (typically 30% to 65%, but some operators can take up to 95%), and, if you use one, an aggregator will take its share too. Security (how you prevent the copying of your app) and manageability are common issues with the PPD model, but in some scenarios it might be the only monetization option. Operator billing can be quite difficult to handle on your own,

² invespcro.com/blog/in-app-purchase-revenue

particularly if you want to sell in several countries, as you need to sign contracts with each operator in each country. For unknown reasons some operators, like Vodafone, seem to remove operator billing as an option for Android Play in some key markets, like UK and Germany and only just recently (2016) reintroduced those options. Possibly because better alternatives, like local mobile bank payments become available.

It is worth noting that most of the vendor app stores are pursuing operator billing agreements. The principal reason they are doing this is that typically, when users have a choice of credit card and operator billing methods users show a significant preference for operator billing.

Credit-card billing is used by Apple, Google (in some cases), Amazon and other stores. Apple has required iPhone users to provide credit-card data at registration for many years, and Google now requires this as well for Android users. Having this information entered before purchase is, according to analysts, a key differentiator for higher monthly per app revenue.

The last payment option is to create your own website and implement a payment mechanism through that, such as PayPal mobile, SEPA payments or others.

Using PPD can typically be implemented with no special design or coding requirements for your app and for starters we would recommend using the app store billing options as it involves minimal setup costs and minor administrative overhead.

For each form of payment it is important to determine price elasticity of demand³. Increasing the price does not necessarily mean higher total revenue (and vice versa), your price needs to match expectations of your user base.

³ en.wikipedia.org/wiki/Price_elasticity_of_demand

In-App Purchase

In-app purchase (IAP) is a way to charge for specific actions or assets within your application. A very basic use might be to enable the one-off purchase of your application after a trial period — which may garner more sales than PPD if you feel the features of your application justify a higher price point. Alternatively, you can offer the basic features of your application for free, but charge for premium content (videos, virtual credits, premium information, additional features, removing ads and alike). Most app stores offer an in-app purchase option or you can implement your own payment mechanism. If you want to look at anything more than a one-off “full licence” payment you have to think carefully about how, when and what your users will be willing to pay for and design your app accordingly.

Recurring in-app payments, also known as subscriptions, are offered by most platforms as well. These type of payments fit well when your app offers content that is regularly updated, such as online newspapers or digital magazines. Recently the revenue split between appstores and developers increased depending on the commitment of your users. e.g. the longer your users stay involved and purchase in-app, the higher the percentage of your developer cut (ranging from 70% to 85%).

In-app purchases have become the leading monetization model in many markets, particularly among those that use free distribution to get users hooked and obtain a larger user base before turning them into buyers (for features such as buying extra power, extra levels, virtual credits and alike).

Appboy expects in-app purchases to generate 48% of all direct revenue by the end of 2017⁴.

⁴ www.appboy.com/blog/in-app-purchase-stats/

If you target specific countries, be aware of different behavior, e.g. in China the initial purchase is 99% of all revenue generated, while IAP is very low, while in the US it is the other way around.

It should also be obvious that you will need to design and develop your application to incorporate the in-app payment method. If your application is implemented across various platforms, you may need to implement a different mechanism for each platform (in addition to each app store, potentially).

As with PPD we would recommend that you start with the in-app purchasing mechanism offered by an app store, particularly as some of these can leverage operator billing services (such as Google Play) or utilize pre-existing credit-card information (such as Apple or Amazon), or with in-app payment offered directly by operators. From a user's perspective, this is the easiest and most convenient way to pay (one or two clicks, no need to enter credit card numbers, user names or other credentials), so developers can expect the highest user acceptance and conversion rates.



Mobile Advertising

As is common on websites, you could decide to earn money by displaying advertisements. According to VisionMobile's survey among 21,000 developers⁵, 38% of mobile app developers are still reliant on this revenue model- although it seems proven that it is proving profitable for only a small minority: 83% of the study's participants who rely on ads, make less than \$10,000 a month.

There are a number of players who offer tools to display mobile ads and it is the easiest way to make money on mobile browser applications. AdMob⁶, MobAds⁷ and InMobi (for games)⁸ are a few of the parties that offer mobile advertising. However, because of the wide range of devices, countries and capabilities there are currently over 70 large mobile ad networks. Each network offers slightly different approaches and finding the one that monetizes your app's audience best may not be straightforward. There is no golden rule; you may have to experiment with a few to find the one that works best. However, for a quick start you might consider using a mobile ad aggregator, such as Madgic⁹, smaato¹⁰ or inneractive¹¹ as they tend to bring you better earnings by combining and optimizing ads from 50+ mobile ad networks. Most aggregators can also operate as an Ad Exchange, providing Real Time Bidding (RTB),

⁵ visionmobile.com/blog/2016/08/mobile-developers-advertising

⁶ google.com/admob

⁷ mobads.com

⁸ inmobi.com

⁹ madgic.com

¹⁰ smaato.net

¹¹ inner-active.com

like a live auction where the price of each ad is determined at run-time.

Most ad networks take a 30% to 50% share of advertising revenue and aggregators another 15% to 20% on top of that, but even with those numbers aggregators are still more profitable than trying to integrate all separate ad networks yourself.

If your app is doing really well and has a large volume in a specific country you might consider selling ads directly to advertising agencies or brands (Premium advertising) or hire a media agency to do that for you.

Again many of the device vendors offer mobile advertising services as part of their app store offering and these mechanisms are also worth exploring. In some cases you may have to use the vendor's offering to be able to include your application in their store.

In-application advertising will require you to design and code your application carefully. Not only the display location of ads within your app needs to be considered with care, also the variations and opt-out mechanism. If adverts become too intrusive, users may abandon your app, while making the advertising too subtle will mean you gain little or no revenue. Relatively new compared to traditional banner advertising is interstitial advertising: This term is generally used to describe an ad that takes up the entire screen and typically has a "skip screen" button at the bottom. Other new ad formats include playable ads or rewarded ads, especially for games. It may require some experimentation to find the right level and positions in which to place adverts.

Sponsorships

The German startup Appponsor¹² offers a new way of earning money without the need to display advertising or charge a download fee: The user gets your app for free and is prompted to sign-up for a newsletter of your sponsor. The sponsor will in return pay the developer an amount for each newsletter registration. Not to be confused with App Sponsors, companies who pay the development costs of your app in return for a stake, like Apps Funder¹³.

Indirect Sales

Another option is to use your application to drive sales elsewhere.

Here you usually offer your app or website for free and then use mechanisms such as:

- 1. Affiliate app programs:** Promote third party or your own paid apps within a free app. *CheetahMedialink.com* is a service provider that offers this type of monetization.
- 2. Data reporting:** Track behavior and sell data to interested parties. Note that for privacy reasons you should not reveal any personal information, ensure all data is provided in anonymous, consolidated reports.
- 3. Virtual vs. real world:** Use your app as a marketing tool to sell goods in the real world. Typical examples are car apps, magazine apps and large brands such as McDonald's and Starbucks. Also, coupon applications as Groupon often use this business model. Only 10% of the participants

¹² appponsor.com

¹³ appsfunder.com

in the DevEconomics Report Q1 2017 are relying on this business model, although mobile commerce developers are more than three times as likely to make over 100K USD per month than those monetizing with ads.¹⁴

4. **In-app data collection:** Use capture forms for third party newsletters sign-ups or even present full surveys using services like Pollfish¹⁵.

There is nothing to stop you from combining this option with any of the other revenue generation options if you wish, but take care that you do not give the impression of overly-intrusive promotions.

Component Marketplace

A Component Marketplace (CMP) provides another opportunity for developers to monetize their products to other developers and earn money by selling software components or white-labelled apps. A software component is a building block piece of software, which provides a defined functionality, that is to be used by higher level software.

The typical question that comes up at this point is on how CMPs contrast to open source. As a user, open source is often free-of-charge. Source code must be provided and users have the right to modify the source code and distribute the derived work.

Some component providers require a license fee. They may provide full source code which enables the developer to debug into lower level code. Some CMPs support all models:

¹⁴ vmob.me/DE1Q17

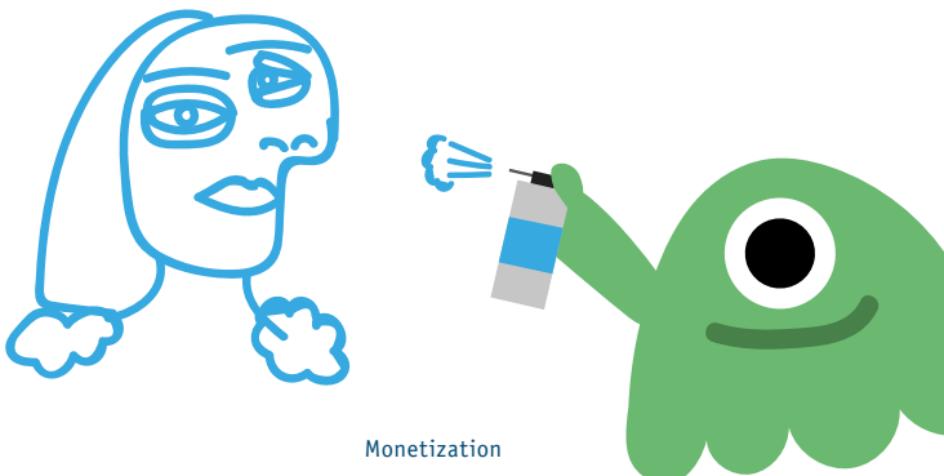
¹⁵ pollfish.com/publisher

Paid components with or without source code as well as free components with or without source code.

If you are a developer searching for a component, CMPs offer two major advantages: First, you do not have to open source your code just because you use software components. All open source comes with a license. Some licenses like the Apache are commercially friendly; others, such as AGPL and OSL, require you to open source your code that integrates with theirs. You might not want this. Secondly, CMPs provide an easy way to find and download components. You can spend days browsing open source repositories to find the right thing to use.

Component marketplaces have existed for decades now. The most prominent marketplace is for components for Visual Basic and .NET in the Windows community. Marketplaces such as componentOne and suppliers like Infragistics are well known in their domain. The idea of component marketplaces within the mobile arena is quite new. ChupaMobile¹⁶ is a relevant player in this domain.

¹⁶ www.chupamobile.com



App Platform Subscriptions

The 2nd wave after the mobile consumer app adaption came from the small and medium enterprises. According to Gartner¹⁷ the development capacity required to meet the demand of enterprise apps will become critical in 2017.

Smaller businesses, including your bakery around the corner, are interested to have their own app, but do not have the budget to justify the development. For this target market the App Platforms are an excellent choice. The developer designs an app with some default options, adds the content and sells a subscription to the company. All content is hosted online, only the app framework is downloaded from the app stores. So the company has to renew his subscription each month or year to keep his app alive. Mastering the App Platform development tools and selling a couple of new subscriptions a month ensures a recurring revenue for the developer.

Choosing your Monetization Model

So with all these options what should your strategy be? It depends on your goals, let us look at a few:

- Are you convinced users will be willing to buy your app immediately? Then sell it as PPD for \$0.99, but beware while you might cash several thousand dollars per day it could easily be no more than a few hundred dollars per week if your assessment of your app is misplaced or the competition fierce. The Application Developer's Alliance recommends the PPD monetization method mainly for high-production apps, apps with barriers to entry and

¹⁷ www.gartner.com/newsroom/id/3076817

high-volume apps¹⁸. This includes games and apps for entertainment, productivity, navigation and news.

- Do you target a large user base? Consider distributing your application for free with in-app purchases, or with mobile advertising (you could even offer a premium ad-free version).
- Are you offering premium features at a premium price? Consider a time or feature limited trial application then use in-app purchasing to enable the purchase of a full version either permanently or for a period of time.
- Are you developing a game? Consider offering the app for free with in-app advertising or a basic version then use in-app purchasing to allow user to unlock new features, more levels, different vehicles or any extendable game asset¹⁹.
- Is your mobile app an extension to your existing PC web shop or physical store? Offer the app for free and earn revenue from your products and services in the real world.
- Does your app contain content that is updated frequently, like digital magazines? Offer recurring in-app payments and make sure that visitors return.
- Do you offer physical goods, like a webshop app? Offer your app for free and make money on the margins of your customer's purchases.

¹⁸ www.appdevelopersalliance.org/app-monetization

¹⁹ Learn more about monetizing games in the respective section of the “Mobile Games” chapter

User Acquisition

The flip side of revenue generation is marketing and promotion. The need might be obvious if you sell your application through your own website, but it is equally important when using a vendor's app store. App stores are the curse and the blessing of mobile developers. On the bright side they give developers extended reach and potential sales exposure that would otherwise be very difficult to achieve. On the dark side the more popular ones now contain millions of apps, decreasing the potential to stand out from the crowd and be successful, leading many to compare the chances of app store success to the odds of winning the lottery.

So, here are a few tips and tricks to help you raise your odds.

Strategies To Get High Rankings

The most important thing to understand about app stores is that they are distribution channels and not marketing machines. This means that while app stores are a great way to get your app onto users' devices, they are not going to market your app for you (unless you purchase premium positioning either through banners or list placings). You cannot rely on the app stores to pump up your downloads, unless you happen to get into a top-ten list. But do not play the lottery with your apps, have a strategy and plan to market your app.

We have asked many developers about the tactics that brought them the most attention and higher rankings in app stores.

Many answers came back and one common theme emerged: there is no silver bullet – you have to fire on all fronts! However it will help if you try to keep the following in mind:

- You need a kick ass app: it should be entertaining, easy to use and not buggy. Make sure you put it in the hands of users before you put it in a store.
- Polish your icons and images in the app store, work on your app description, and carefully choose your keywords and category. If unsure of, or unsatisfied with the results, experiment.
- Getting reviewed by bloggers and magazines is one of the best ways to get attention. In return some will be asking for money, some for exclusivity, and some for early access.
- Get (positive) reviews as quickly as possible. Call your friends and ask your users regularly for a review.
- If you are going to do any advertising, use a burst of advertising over a couple of days. This is much more effective than spending the same amount of money over 2 weeks, as it will help create a big spike, rather than a slow, gradual push.
- Do not rely on the traffic generated by people browsing the app store, make sure you drive traffic to your app through your website, SEO and social media.
- Update your app often with appealing features. This will notify your existing users that there is an update available, bringing them back in action and engage with your app more. Potential new users are more confident with apps that are updated regularly than apps that have not been updated for ages. Instead of publishing all your features at once at launch, limit them to the most basic ones and gradually update the app by adding a new item.

Multi-Store vs Single Store

With 120+ app stores available to developers, there are clearly many application distribution options. But the 20 minutes needed on average to submit an app to an app store means you could be spending a lot of time posting apps in obscure stores that achieve few downloads. This is why a majority of developers stick to only 1 or 2 stores, missing out on a potentially huge opportunity but getting a lot more time for the important things, like coding! So should you go multi-store or not?

Multi-store	Single store
The main platform app stores can have serious limitations, such as payment mechanisms, penetration in certain countries, content guidelines.	90%+ of smartphone users only use a single app store, which tends to be the platform app store shipping with the phone.
Smaller stores give you more visibility options (featured app).	Your own website can bring you more traffic than app stores (especially if you have a well-known brand).
Smaller stores are more social media friendly than large ones.	Many smaller app stores scrape data from large stores, so your app may already be there.
Operators' stores have notoriously strict content guidelines and can be difficult to get in, particularly for some types of apps.	For non-niche content, operator or platform stores may offer enough exposure to not justify the extra effort of a multi-store strategy.

Multi-store	Single store
Smaller stores may offer a wider range of payment or business model options, or be available in many countries.	Some operators' stores have easier billing processes – such as direct billing to a user's mobile account -- leading to higher conversion rates.
Some developers report that 50% of their Android revenues come from outside of Android Market.	iOS developers only need 1 app store.

The platform app stores should give you general coverage for users, but over time, it is in your interest to adapt your app store strategy to match your targeted user base, and utilize the app stores that best reach it. This could mean using particular operator stores, stores popular in a specific country, or simply sticking with the platform stores. There are some third-party app stores with large audiences, such as the Amazon app store for Android, which offers developers a number of ways to monetize their apps, such as PPD and in-app payments in several countries. Additionally, in some countries, there are locally popular app stores, such as [AndroidPit²⁰](#) in Germany, or one of the many China-specific Android stores.

²⁰ androidpit.de

What Can You Earn?

One of the most common developer questions is about how much money they can make with a mobile app. It is clear that some apps have made their developer's millionaires, while others will not be giving up their day job anytime soon. Most app developers are not generating enough revenue to break even with development costs and single platform developers confirmed it was not enough to support a standalone business.

According to VisionMobile, over 50% of them are below the "app poverty line" of \$500 per app per month²¹.

Mobile games seem to offer the most options to make money according to Pulse²². And even if Android phones and tablets are outnumbering iOS devices, China overtakes the U.S. in iOS App Store revenue²³.

Ultimately, what you can earn is about fulfilling a need and effective marketing. Experience suggests that apps which save the user money or time are most attractive (hotel discounts, coupons, free music and alike) followed by games (just look at the success of Angry Birds) and business tools (office document viewers, sync tools, backup tools and alike) but often the (revenue) success of a single app cannot be predicted. Success usually comes with a degree of experimentation and a lot of perseverance.

²¹ www.vmob.me/DE1Q17

²² www.linkedin.com/pulse/mobile-gaming-monetization-making-more-money-your-app-jehan-damji

²³ <https://techcrunch.com/2016/10/20/china-overtakes-the-u-s-in-ios-app-store-revenue/>



Epilogue

Thanks for reading this 17th edition of our Mobile Developer's Guide. We hope you have enjoyed reading it and that we helped you to clarify your options. Perhaps you are now ready to get involved in developing a mobile app or have discovered new options in the app business. We hope so. Please also get involved in the community and share your experiences and ideas with us and with others.

We said it before and we cannot say it often enough: If you like to contribute to this guide as a writer or support upcoming editions as a printing sponsor please write to . If you are using Twitter, follow the project on twitter.com/mobiledevguide.

You can of course also get this guide as an ebook- just check amazon.com. Or you simply download the pdf version on our website: www.mobiledevelopersguide.com.





About the Book

This project was initiated by Enough Software in 2009 with the aim to spread knowledge about mobile technologies and to encourage people to enter our community or deepen their existing knowledge. With this edition, Open-Xchange (OX) took over the project and will be driving it forward. Until today, we have given away almost 100,000 hardcopies at events worldwide. Universities and schools in Germany, Netherlands, UK, Spain and South Africa use the book as teaching material. The electronic versions (ebook and pdf) have been downloaded hundreds of thousands of times and the content has been translated into several languages. The book is a non-profit project: the writers, editors, translators and designers contribute their work free of charge. The printing and distribution costs are usually covered by sponsors, this time by OX alone.

The Publisher

Open-Xchange

Open-Xchange (OX) leads the fight to keep the digital landscape open, secure and transparent for all. Ruthlessly committed to an open internet and software ecosystem, OX opens opportunities for the world's leading service providers and telecommunications without compromising the privacy or safety of users.

Since 2005, we have partnered with some of the largest providers in the world to deliver email, messaging and collaboration solutions that include secure storage, file and document management, and best-in-class IMAP and DNS management. Offering this full stack of open source technol-

ogy OX helps global IT businesses deliver innovative and high quality customer experiences.

Our culture is founded on a commitment to openness, a rebellious business spirit and the desire to leave the world better than we found it. Effective teamwork is central to our growth.

Learn more about the benefits of working with OX:

🌐 www.open-xchange.com

🐦 [@openexchange](https://twitter.com/openexchange)

The Authors & Contributors

This project would not have been possible with the ongoing support from the mobile community. These are the folks that have been involved as authors this time. Some of them are on board since 8 years already, others just recently joined. All are awesome.

Aaron Ardiri / RIoT Secure AB

Aaron has been working with mobile technologies since 1999 when Palm OS was the biggest player in the mobile market. He has extensive experience in senior technical management roles but put a lot of his spare time towards mobile game development, cross platform development techniques, security, digital rights management and his current passion focuses on the Internet of Things in regard to security and getting the most out of low powered micro-controllers. He



is the CEO of RIoT Secure, focusing a developer awareness hub and security solutions for the Internet of Things.

 [@riotsecure](https://twitter.com/riotsecure)

 www.riotsecure.se

Andrej Balaz / IXDS

Andrej currently works as a Senior Service Designer at IXDS where he helps established companies and fresh founders alike to build services that people will buy and happily use. With his holistic approach, over 7 years of experience and skills in research and experience design for all the individual bits and pieces that make up a service, he bridges the gap between research and making. Andrej values clear communication and pragmatic approaches that combine iterative development with lean research. In his free time he works as an illustrator. Andrej can be reached on LinkedIn, Twitter or the Jobs to be Done meetup, which he co-organizes.

 [@Designamyte](https://twitter.com/Designamyte)

 www.ixds.com

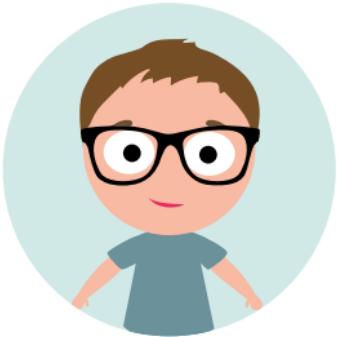


Daniel Böhrs / Open-Xchange

Daniel started developing software in 2006 with PHP and web development. During his computer science studies he focused on Java and especially Android development. After obtaining his master degree he joined Enough Software in 2015, which later got acquired by Open Xchange. In the company he is mainly an Android developer, although he sometimes helps out in the server development area.

 @Boehrsi

 boehrsi.de



Davoc Bradley / MiraLife

Davoc has been working as a software engineer since 1999 specializing in architecture and design of high usage web and mobile systems. Currently he is CTO at MiraLife who specialize in providing web based and mobile software, which aims to improve the lives of people suffering from dementia and other terminal illnesses. Davoc is also a keen musician, avid cricket fan and loves traveling.

 @davocbradley



Sally Cain / RNIB

Sally has worked at RNIB in the area of digital accessibility for more than 17 years. She believes passionately in equal access to digital technology for people with disabilities.

Sally is her organization's representative on W3C standards groups and is also sits on a number of groups at the British Standards Institute (BSI) that relate to standards for ICT. This includes the group responsible for BS8878 the Code of Practice for Web Accessibility. Sally is currently the Accessibility Technology Manager at RNIB with responsibility for the accessibility of all internal and customer facing systems, ensuring that RNIB is delivering on accessibility not only for customers, but for staff too. She also led the writing of RNIB's own internal app standard for accessibility. For this version of the publication, Sally was assisted by Robin Spinks, RNIB's Senior Strategy Manager, with information relating to the latest features in iOS, Android and Windows 10.

 [@sallycain](https://twitter.com/sallycain) and [@robinspinks](https://twitter.com/robinspinks)

 www.rnib.org.uk



Dean Churchill / AT&T

Dean works on secure design, development and testing of applications at AT&T. Over the past several years he has focused on driving security requirements in mobile applications, for consumer applications as well as internal AT&T mobile applications. He has been busy supporting AT&T's emerging Mobile Health and Digital Life product lines. He lives in the Seattle area and enjoys downhill skiing and fly fishing.



Neil Cook / Open-Xchange

Neil Cook has been working in software development and security since 1994, including developing military messaging applications, designing and deploying large-scale internet messaging, and fighting spam and abuse in internet services. He has a PhD in Computer Science from the University of Nottingham, and is author of RFCs 5593 and 5616. Neil is currently responsible for software and service security at Open-Xchange.



 [@neilthepeel](https://twitter.com/neilthepeel)
 blog.open-xchange.com/author/neil

Oscar Clark / Unity Technologies

Oscar Clark is an author, consultant and evangelist for Everyplay from Unity Technologies. He has been a pioneer in online, mobile and console social games services since 1998. He provided 'vision' for one of the first Online games communities (Wireplay - British Telecom); was global lead for games at Hutchison Whampoa (3UK) which included (perhaps) the first mobile in-App purchase; and was Home Architect for PlayStation®Home.



He is a regular columnist on PocketGamer.Biz and Develop-Online, an outspoken speaker at countless games conferences, a mentor for accelerator GameFounders and has guest lectured for several universities. His first book, "Games As A Service - How Free To Play Design Can Make Better Games" is available online.



@athanateus



www.gamesasaservice.net



Julian Harty / Commercetest

Julian's mission is to help people live better lives through mobile technologies. He has worked globally in leadership and engineering roles for Google, eBay, Salesforce, Klarna, Badoo, and many others to help improve engineering practices and deliver better quality software while also improving the fulfillment and motivation for the people involved. He has been actively involved in the modern mobile ecosystem since 2006 and this guide from 2010 onwards. Currently he is working independently, writing mobile apps & suitable test automation tools, and helping others to improve their mobile apps. He is also trying to complete his PhD on improving development and testing of mobile apps.

 [@julianharty](https://twitter.com/julianharty)

 GitHub: github.com/julianharty



Alex Jonsson / Evothings Labs

Alex has been active in the internet business for over 25 years, with a partiality for connecting physical stuff to mobile. He holds a Tech, Dr. habil in Computer Science/Media Technology from the Royal Institute of Technology in Stockholm and freely shares his ideas and thoughts with both the industry and academia. Dr Jonsson also has an eclectic urge to investigate how apps and services act as drivers for new business, by bringing



novel values and ways to make things more connected, thereby binding the universe together in new, clever ways. Alex is co-founder and chairperson of Evothings Labs in Stockholm, Sweden.

 @dr_alexj

 www.evothings.com

Vikram Kriplaney

Vikram has been a mobile developer since when WAP was still cool and Symbian and J2ME were still fashionable. He founded mobile at local.ch in 2007, where he went on to singlehandedly develop massively successful mobile web, iOS and Android apps. He is now Mobile Architect and lead engineer for local.ch and search.ch (Swisscom Directories), where he helped build one of the most awesome mobile engineering teams in Switzerland.

 @krips

 www.local.ch and www.iPhonso.com

Cornelius Kwietniak /

Open-Xchange

Cornelius specializes in graphic, UI, UX and visual design for mobile applications and other interactive technologies. He was in charge of the layout and design of the previous guides. When not involved with something mobile, he loves to experiment with digital art and illustration.



Ruadhán O'Donoghue / Western Technological

Ruadhán is a web and mobile developer based in Ireland. He has worked in web development since 1999, and developed his first mobile web application, a WAP dictionary, back in 2000 when the mobile web was built on WAP and WML, and was browsed on tiny monochrome phone screens. Since then, he has worked in many different roles including Head of Engineering at Afilias/dotMobi. He has been an Editor and contributor to mobile technology site mobiForge.com for over 10 years, and publishes articles on mobile web development regularly. He has just published his first book, "AMP: Building Accelerated Mobile Pages". He currently runs his own web consultancy, Western Technological.

 @rodono

 westerntechnological.com and ruadhan.com



Alex Repty

Alex is a freelance software engineer, specialized in OS X, iOS, watchOS and tvOS software. He has been developing software for Apple platforms ever since he got his first Mac in 2004. Since then, he has helped create a wide variety of applications, some of which were even featured in Apple's "There's an app for that" campaign or won an Apple



Design Award. His passion for clean code, software engineering trends and user experience design make him get up on stage on various iOS-related conferences.

 @arepty

Michel Shuqair / AppValley

Starting with black and white WAP applications, iMode and SMS games in the 1990's, Michel moved to lead the mobile social network Wauwee. Serving almost 1,000,000 members, Michel was supported by a team of Symbian, iPhone, BlackBerry and Android specialists at headquarters in Amsterdam. Wauwee was acquired by MobiLuck, which is now part of Paris based Madgic.com, a mobile monetization platform.

 www.appvalley.nl



Marco Tabor / Open-Xchange

Marco started working in the mobile software business in 2005. Three years later, he joined the dream team at Enough Software where he has been responsible for PR, sales, project management and much more. Since the team has become a part of Open Xchange, Marco fully concentrates on his role as a product owner, mainly for mobile apps. He is also the main coordinator of this book project.

 @enoughmarco



Ian Thain / SAP

Ian is a Mobile Evangelist at SAP, though he started 13 years ago with Sybase Inc. He regularly addresses audiences all over the world providing mobile knowledge and experience for the Enterprise. He also writes articles, blogs & tweets on Enterprise Mobility and is passionate about the Developer & Mobile Experience in the Corporate/Business world.

 @ithain

 www.sap.com and scn.sap.com/blogs/ithain/



Marc van 't Veer / Polteq

Marc is a mobile app test consultant and trainer at Polteq and has worked in different test roles for over 12 years. He is helping companies with mobile device analysis, test strategy, implementing mobile test automation and giving training on mobile app testing in practice, CMAP and API testing. Currently Marc supports companies in improving the mobile app testing based on the TI4 Mobile approach.

 @marc_vantveer

 www.marcvantveer.niobe.nl and www.polteq.com



Robert Virkus / Open-Xchange

Robert has been working in the mobile space since 1998 starting with the WAP mobile Internet technology in the 1990's. As he experienced device fragmentation issues first-hand when developing and porting a mobile app on the Siemens SL42i, he started J2ME Polish - an Open Source feature development framework - in 2004.



After founding the corresponding company Enough Software he joined OX in 2016 and now heads the app development department of OX. He loves retro computers, Lego and his two kids and adorable wife Olga.

 [@robert_virkus](https://twitter.com/robert_virkus)

 Mastodon: [@enough](https://mastodon.cloud/@enough)

Mladenka Vrdoljak / Open-Xchange

Mladenka is completing an apprenticeship as a digital media designer at Open-Xchange. That means she is engaged with User Interface, User Experience and graphic design for mobile applications, as well as coding. She is also responsible for the layout, design and text editing of this guide. In her spare time, she likes to travel, draw and play video games.

 [@_mladenka](https://twitter.com/_mladenka)



DON'T JUST
GO TO WORK
THRIVE!



Open people for an open source!

People here are committed to openness, a rebellious business spirit and the will to revolutionize the market.

We are eMail

Worldwide companies like: 1&1, STRATO, Host Europe, NetCologne, network solutions/web.com and Rackspace use our "white label" products for eMail and collaboration.

Work with professionals around the world!

Join the force which develops products for everyday digital life without sacrificing personal freedom.

Interested in becoming part of something big? Find your new career at open-xchange.com/jobs or contact recruiting@open-xchange.com



Stay Open.

OX®