1. Theoretical Foundation
   1. History and theory of moneyless exchange systems
      1. Barter

In the early ages of mankind, long before the concept of money existed, people relied on direct trades of items such as food, clothing, tools and weapons. Such direct exchanges of goods or services are called barter. Bartering has one obvious disadvantage: It requires a double coincidence of wants. If one person owns a hammer that they want to trade for a bag of rice, they need to find somebody who not only wants to obtain a hammer, but also has a bag of rice to offer in return. Otherwise a trade can not take place. Another problematic aspect of bartering is the lack of common measures. Without money or another standardized form of measurement, it is often difficult to determine and negotiate the value of goods or services, which might result in a disadvantage for one of the involved parties.

These commonly recognized drawbacks of direct bartering systems eventually led to the introduction of metal as a standardized representation of value. Around 2500 BC, copper rings were a common form of payment in ancient Egypt. The first recorded evidence of money as an accepted means of payment in exchange of goods dates back to around 600 BC, when the first stamped coins were minted in Lydia, an ancient nation located in an area that is now a part of Turkey.

Throughout the course of history, money evolved into an increasingly abstract concept, establishing a convenient and widely accepted means of payment and thus providing the basis for rapid economic growth of modern societies, while simultaneously raising new severe issues: The possibility to accumulate wealth without any maximum limits and the general concept of interest along with banks growing into powerful and insufficiently supervised institutions are only some of the problems that have created a grave crisis for the monetary system, the effects of which are perceptible more than ever in current times.

Throughout the evolution of monetary societies, barter never ceased to exist, often gaining immense importance during times of financial crisis such as the hyperinflation that took place in the Weimar Republic during the early 1920s.

Over time, the obstacles that direct forms of bartering are facing were met with a more organized system of exchange. Instead of trading goods or services directly and one-to-one, parties would pay and be paid with virtual value units, which in return could be spent on purchases from a different party. A centralized institution within the barter exchange keeps track of each member's account balance and processes all transactions. This opened up a new horizon of possibilities, making it more feasible for businesses to engage in barter exchange while at the same time forming the basis for new networks to emerge.

* + 1. Local exchange trading systems

The term "local exchange trading system" (abbreviated to LETSystem or LETS) was coined in by Michael Linton in 1983. Linton, originally from the UK, migrated to Courtenay, Canada in the 1970s to work as a teacher for the Alexander Technique. During the early 1980s, the small town was hit by the recession and local purchasing power suffered immensely. Linton, motivated to continue treating his clients who couldn't afford to pay him anymore, developed and implemented a local system of community exchange that required no use of cash. He established five fundamental criteria that must be fulfilled by a system in order to be considered a LETSystem:

Cost of service:

The system should be administered from within the community in a professional and sustainable way. Individuals who run the accounts of the system should be appropriately rewarded for their efforts in the currency that is used within the system. However, any attempt to generate profits from the system, for example by taking commisions on transactions, is illegitimate.

Consent:

The system is based on consent given freely by all participants. There is no obligation for any member to participate in a trade. This also means that every new participant joining the network will start with an account balance of zero.

Disclosure:

In order for the system to be trust-worthy, it must be transparent to all participants. This can only be achieved by keeping information about a member's account balance and trading volume available to all other members. This will also facilitate participants in collectively regulate the system.

Equivalence to the national currency:

The LETSystem currency uses national currency as a means of measure. Without the possibility of putting the system's currency in relation to something of real-life value, it would be virtually impossible for participants to determine the value of their efforts or decide whether another participant's offer is reasonably priced.

No interest:

In order for the system to operate without profit, no interest in charged on negative balances or paid on positive balances. This should also discourage participants in storing up large amounts of credits.

Over the past 30 years LETSystems have gained immensely in popularity all over the world, particularly during times of recession. Australia and New Zealand where amongst the most avid promoters of LETSystems. In 1989, the Australian government allocated a budget of $50,000 to faciliate the development of LETSystems all over the country. By the mid-nineties, 250 different systems existed in Australia with the largest one having around 2,000 members. In 2001, Germany counted over 35,000 members across 350 LETSystems. While systems are inherently independent, there are many examples of local systems cooperating with each other to span greater networks.

With technology evolving, many LETSystems have made use of the new possibilites in one way or the other. While most of them at least have a website providing some basic information, only few have fully adapted to the new era by using software for adminestering account balances and offering a digital directory to participant's offers and requests.

* + 1. Time banking

Similarly to LETSystems, but with a very different approach on the aspect of measuring value, time banks use units of time as their currency. The first recorded evidence of the use time units as a means of payment dates back to 1825, the year in which Robert Owen, a wealthy industrialist and social reformer from Wales, emigrated to America, purchased a small town in Indiana and called it "New Harmony". His goal was to realize his vision of a *"New Moral World, a world of enlightenment and prosperity leading to human happiness defined as mental, physical and moral health enjoyed in a rational way of life."[[1]](#footnote-1)*

The experiment failed, but the idea of a time-based was adapted soon afterwards by Josiah Warren, an American individualist anarchist who was amongst the initial participants of Owen's society. In 1927, Warren opened up the Cincinatti Time Store, the first store that accepted "labor notes" which represented a promise to perform labor in the future, as a form of payment to purchase goods. The standard used to determine the value of one hour of labor was 12 pounds of corn, based on the calculation that this is the amount of corn that would be produced by one hour of labor. Warren, who was a strict follower of the labor theory of value, believed that all work should be valued equally and it would therefore be unethical to charge more for a product then the amount of labor it takes to produce it. The store enjoyed great popularity until it was closed down in 1930, when Warren deemed his experiment successful and set out to start new colonies based on his ideas.

The first succesful modern time bank was founded in 1991 by Paul Clover in Ithaca, New York. After a recently established LETSystem in Ithaca had failed to attract enough members, Clover developed the system of "Ithaca HOURS". He initially managed to convince 90 participants, amongst them a massage therapist and a toy store owner, to accept HOURS as an exchange for services or goods. The system became an immense success and is still flourishing today, serving as an inspiration for many time banks to appear all over the world throughout the last decade.

* 1. Case studies
     1. Warnowknoten

Established in 1996, "Warnowknoten" is a local trading exchange circle in Rostock, Germany. The currency used by participants to exchange services or goods is called "Knoten" (German for "knots"). The value of one Knoten is freely negotiable and not tied to a specific monetary or time-based measure, though as a guidance it is suggested to roughly set a value of 10 Knoten for one hour of work. During its peak time in 2003, the community counted about 200 members. During the last 10 years, the level of participation decreased significantly. Only few people still actively take part in the exchange and the community has failed to attract enough new members.

One reason for this development might be the lack of appeal for younger generations due to the community's relatively old-fashioned mode of operation. A newspaper, containing general information about meetings and organizational matters as well as a full directory of all offers and requests, is issued quarterly by the administering team. The community also has a website that offers various information, a PDF version of the newspaper's current issue and printable forms necessary for signing up as a new member, advertising new requests or offers and submitting a transaction to be processed on the participating party's account.

The process of engaging in an exchange requires several steps: If a member finds an advertisement in the paper that they are interested in, they can look up the contact information of the person behind the offer in a member directory through a 4-digit identification number. The offers and requests have no initial prices assigned to them – members have to negotiate the amount of "Knoten" they are willing to accept as a reasonable price. After the service has been provided or exchange of items has taken place, both participating members have to fill out and sign a form about the transaction and send it to the team that takes care of all accounting. The responsible team then updates the account balance of both parties according to the value documented in the transaction form.

While the community has obviously made an attempt to adapt to the demands of modern times by providing a website and making content available in a digital form, the necessary steps to truly make the network more accessible and convenient have not been taken. The option for members to log into the platform to view their account balance, initiate contact with other members and digitally submit their transaction claims would immensely simplify the process of participation and potentially make it easier to attract new, younger members who are used to getting things done on the internet.

* + 1. Community Exchange System

The system, orginally known as the "Cape Town Talent Exchange" was founded 10 years ago in Cape Town, South Africa by political activist Tim Jenkin. It started off as a regular LETSystem, introducing the "Talent" as the unit of currency. The community grew fast and started to attract attention from all over the country. In the same year, six more exchange groups were established in different parts of South Africa. Initially, they operated independently from each other, but as demands to interact with particpants from different groups started to rise, the initiators and administrators of the communities started to think about ways to connect their systems into one big network. The initial problem with this idea was the fact that all communities are inherently closed off systems within which all transactions have to balance out to zero. If one person would be allowed to simply spend their credit within a different system, both affected systems would suffer from an imbalanced economy. But since all of the newly established communities in South Africa already made use computer technology to manage accounts and directories, an internet-based solution was quickly found:

*"The solution was to use "virtual users", who are like real users with accounts in all other groups. Thus if seller A in group A wants to sell something to buyer B in group B, the account would record that seller A actually sold to virtual user A in group A. Virtual user A, who has accounts in all other groups, could "step across" to group B (and be called virtual user B) and sell on to final buyer B in group B. The account in group A would balance to zero as the credits from the sale by seller A would cancel out with the debits of the purchase by virtual user A; and the account in group B would also balance to zero as the credits from the sale by virtual user B would cancel out the debits of the purchase by user B. The balance of trade between the groups could then be measured by the figures recorded for the virtual users."*[[2]](#footnote-2)

The newly created network to connect local exchange groups was called "Community Exchange Service" (CES). It quickly grew in popularity with more and more groups joining from all over the world to take advantage of the new possibilites of cooperation amongst different groups. By the beginning of 2013, the CES website counted 499 separate community exchanges from 52 different countries.

Unfortunately, not all of the groups listed appear to be still active. During my research, I tried joining a group in San Francisco and in Berlin through the "sign up" section on the CES website. After the basic registration process, I was asked to wait for an e-mail of approval from the respective group administrator, but those e-mails never came. Without the approval, the registration process can't be completed, hence I'm not able to log into the CES system to browse offers and requests or interact with other members.

* + 1. TaskRabbit

TaskRabbit is an online marketplace where people can outsource personal tasks and find people in their neighborhood to do these tasks for them. The company was founded in 2008 in Boston, Massachusetts and has since spread to eight more big cities in the U.S., including Los Angeles, New York City and the San Francisco Bay Area. TaskRabbit has drawn a lot of attention in the start-up scene and is backed by major investors.

The principle is fairly simple: Someone posts a task on the platform, specifies the location, a time frame and and what they're willing to pay somebody for getting the task done. "TaskRabbits" can then bid on the task – this can be higher or lower than specified by the user who posted the task. Task posters can decide whether to review the bidders of the task and assign it to a bidder of their choice or have the system automatically accept the first bidder who is willing to accept the specified price.

Common tasks range from pick-up and delivery of random things to grocery shopping, house cleaning and moving help with the most popular task being IKEA furniture assembly. The platform offers guidance for task posters on how to estimate appropriate task prices. A rating system as well as a thorough vetting process including a video interview and a federal background check for people who sign up as task runners ensures that task posters can count on their tasks being handled by someone reliable and trustworthy.

The variety of people serving as task runners is vast: College students, stay-at-home moms who are running their errands anyway and don't mind putting one or two more tasks on their to-do list, recent retirees who have a lot of free time and are happy to help others out while supplementing their pension – all these are very active demographic groups on TaskRabbit. There are actually quite a few people whose participation in TaskRabbit serves as their primary or even single source of income. According to Leah Busque, founder and CEO of TaskRabbit, the network's most active runners earn up to $5,000 a month[[3]](#footnote-3), making task running a feasible alternative or supplement to regular or low-paid employment.

TaskRabbit and other local service networks with similar business models make money by adding a service fee – 20% in the case of TaskRabbit – to the price that a bidder offers for performing a task. The company makes use of cutting-edge technologies puts a strong emphasis on mobile to make the process of connecting task posters and task runners as easy and hassle-free possible. Through dynamic location services and mobile alerting systems, potential task "TaskRabbits" can be alerted when a task pops up in their direct surrounding, creating possible scenarios like this: Someone posts a task that involves picking up a few items from IKEA. A task runner who is at the IKEA store at the very same moment of the task being posted gets notified. The task runner bids on the task and lets the task poster know that he or she is already at the store. The task poster gets notified about the incoming bid if it is accepted, the task runner is able to collect the items right way. According to Busque, the average time between a task being posted and the first bids rolling in is less than 10 minutes.

With over 4,000[[4]](#footnote-4) verified TaskRabbits, the platform has grown immensely in the last few years, branching out to more and more cities across the U.S. and eventually planning to launch internationally.

* 1. Available Technical Solutions
     1. Back-end technologies
        1. The LAMP stack and its variations

LAMP is the acronym for a solution stack based on the following components:

**L**inux – the operating system

**A**pache – the web server

**M**ySQL – the database management system

**P**HP, Perl or Python – the scripting language

These four layers combined form the basic architecture for a general purpose web server serving as the underlying back-end for many dynamic web sites and applications. LAMP has become increasingly popular throughout the last decade, with many large and traffic-intensive platforms such as Flickr, Facebook and Wikipedia using it as their base architecture, though particular components of the stack may be exchanged with other technologies to fit indvidual requirements.

The LAMP stack offers several significant benefits: The components are open-source and available for free, provided they are used in other open-source projects. Required hardware components and commercial licences for database systems and application servers are very inexpensive compared to other popular solutions such as the ASP.NET or J2EE frameworks. LAMP is considered reasonably secure, very flexible and, if configured correctly, can by highly performant and scalable.

Disadvantages of the LAMP approach are seen less commonly in the bundle as a whole, but rather in known weaknesses of its components in particular scenarios. The following discusses the individual layers in further detail and suggests viable alternatives for each component.

* + - * 1. Operating system solutions

As stated above, the L in LAMP refers to the operating system that the server is running on. Traditionally, this would be one of the many freely available Linux Versions: CentOS, Debian, RedHat Enterprise Linux, SuSE Linux or Ubuntu. However, the bottom layer is generally exchangeable with any arbitary OS – there are WAMP and MAMP configurations that run Windows or Mac OS as the underlying system respectively. Furthermore, there are a handful of UNIX-based operating systems that are less popular in commercial use but hold significant advantages over the traditional approach in back-end architecture.

Solaris, developed by Sun Microsystems, offers high scalability and a set of powerful features such as ZFS, a file system combined with volume management capabilites that is designed to be highly robust against data corruption through use of data snapshots, multiple copies, and data checksums. Due to its excellent support of multithreading, Solaris also shows better performance than Linux on most hardware platforms, especially on larger installations with multiple CPUs and cores. Therefore, Solaris distributions are certainly a viable alternative if maximum performance is a crucial business requirement.

Another solution that gained a lot of popularity, especially in environments in which security considerations are of high priority, is OpenBSD. It is designed to be outstandigly robust against attacks, for example by prohibiting simultaneous writing and execution processes on the same chunk of memory. A major disadvantage of OpenBSD is the systems relatively poor support of symmetric multiprocessing, making it significantly slower than most other operating systems. OpenBSD is a good solution for projects in which security is considered more important than performance.

* + - * 1. Web server solutions

The layer that contains the actual web server in the LAMP stack is usually a version of Apache. Apache was developed in 1995 as is still the most popular web server with a market share of over 60% on March 9, 2013, according to daily updated usage statistics on the W3Tech website[[5]](#footnote-5). Apache is already installed and pre-configured in common Linux distributions and therefore relatively easy to set up. It comes with a variety of built-in features and is highly configurable.

An alternative that sees continuous growth in popularity since its release in 2004 and made its way in the the top three of the most commonly used web servers is nginx. As opposed to Apache, it uses an asynchronous approach, resulting in significantly lower resource consumption. Being able to handle simultaneous requests in a single thread, it performs faster than Apache in most scenarios, especially for delivering static content.

If the main point of concern when chosing a web server is maximum scalability, Yaws (short for "Yet another web server) is a viable alternative. Written in Erlang and using a lightweight threading system, it is built to handle a high number of concurrent processes and therefore interesting for high-traffic websites and applications that face a high number of simultaneous requests.

For Java-based solutions, various open-source and commercial web servers are available, with Jetty, Tomcan and Glassfish being amongst the most popular ones.

* + - * 1. Database solutions

The third layer of the LAMP stack is formed by the database management system, traditionally this would be MySQL, a popular and well supported solution that is used by many high-traffic websited including Wikipedia, Facebook and Flickr. While MySQL is generally a robust and flexible database managment system that is easy to use and very fast for simple queries, it shows significant performance issues on more complex queries.

An alternative to MySQL that is optimized to perform much better on complex queries is PostgreSQL. PostgreSQL also has the advantage of coming with a permissive free licence that allows for free use even in commercial projects, while MySQL requires paying for a commercial licence when used in non-open source projects.

Another SQL-based solution with a completely free licence is SQLite. The main difference between SQLite and most other databases is that SQLite is not running on a server – it is merely represented by a flat file on a disk. This has both advantages and disadvantages. Making a complete back-up of the database for example could not be any simpler as it requires nothing more than copying one file. On the other hand, handling concurrent clients that try to access the database at the same time becomes a problem. SQLite is therefore mainly used in embedded applications, where only a single client will need to access the the stored data.

Besides the traditional SQL-based approach, a new wave of non-relational database solutions has drawn a lot of interest during the past few years. The so-called NoSQL systems use a different approach of storing and retrieving data, making it more scalable for large amounts of data. However, as most systems use simple key-value storage mechanisms which may be structured, but are lacking the possibility of modelling relationships between different fields, the use of a NoSQL database is only advantageous in certain scenarios. Where relationships and complex queries are of little interest, large chunks of data can be accessed and written significantly faster than with any SQL-based solution. Popular systems are MongoDB, which holds at least some SQL-like functionality such as queries and pre-defined indexes, CouchDB, which is optimized for high consistency and Redis, which is famous for being extremely fast.

* + - * 1. Server-side programming languages

The top layer of the LAMP is represented by the server-side programming language, typically a scripting language. Released in 1995, PHP is still by far the most popular and wide-spread choice when it comes to server-side languages – as of March 9, 2013, it is used in 78.7% of all websites recorded by the W3Techs survey[[6]](#footnote-6) and is still showing an upwards trend[[7]](#footnote-7). PHP's immense popularity can undoubtedly be seen as an advantage in itself when considering it for a new project – PHP developers are easy to find, learning the language is relatively straightforward and a big community of experienced developers offers great support for beginners. There are countless libraries and extensions available and it is widely supported by other frameworks. It is also highly polarizing. While it is often sharply criticized by blogging developers (such as Alex Munroe in his blog post "PHP: a fractal of bad design"[[8]](#footnote-8)) for being poorly designed and profoundly inconsistent, many others (such as [Fabien Potencier](http://fabien.potencier.org/) in his blog post "PHP is much better than you think"[[9]](#footnote-9)) argue that while it may not be the most perfect language in the world, it has come a long way in the past few years and evolved into a mature language that offers full support for OOP, is highly flexible and convenient to write.

Some alternatives to using PHP as the server-side programming language are Perl, Python and Ruby. While Perl has shown a steady decline in popularity over the past decade (as indicated by a Google Trends query comparing the web search interest of keywords "Perl", "Python" and "Ruby"[[10]](#footnote-10)) it might still be a viable option where high flexibility is a key concern. Perl 6, a complete redesign of the Perl language, is currently under development and might have the potential of reviving Perl to a comeback.

Python, released in 1991, is widely appreciated for its elegance and readability. It is a powerful yet simple language that, in spite of its relatively small market share (0,2% of the websites recorded on March 9, 2012 by W3Tech[[11]](#footnote-11)), still remains a viable option for web and mobile development. The most commonly used web framework for Python is Django. Labeling itself the "web framework for perfectionists with deadlines"[[12]](#footnote-12), Django is designed to enforce clean design patterns and make developers produce well-organized, reusable code. Django is used in popular websites such as Pinterest and Instagram and also serves as the underlying framework of many mobile apps, web-based and native alike. In the latter case, Django would typically be combined with a RESTful API and serve responses in a format like JSON or XML.

Another increasingly popular solution is Ruby in combination with the Rails framework. It has drawn a lot of attention especially in the start-up world, where agile development methods and fast-changing requirements benefit from a flexible language that enables fast production of results and iterative approaches. Though Ruby is significantly slower its direct competitors PHP and Python, Ruby on Rails seeing vast growth in popularity as a framework for websites and mobile apps. It is used by platforms such as Hulu and GitHub. In January 2013, Rails released three urgent updates due to the discovery of highly critical vulnerabilities, putting 240,000 websites at risk. It remains to be seen whether those recent discoveries of security threats will cause the framework to lose some of its popularity.

Some other languages that are highly suitable for server-side programming are Scala, Erlang and Haskell. Of course, good old Java remains a viable option as well and still holds a market share of 4% (W3Tech website on March 10[[13]](#footnote-13)). Very recently, there has also been a lot of fuss in the developer community about Node.js, a lightweight and highly scalable platform that brought JavaScript to the back-end world.

* + - 1. Microsoft's ASP.NET framework

With a market share of over 20%[[14]](#footnote-14) on March 10, 2012, ASP.NET is the second most common server-side web application framework on the internet.

Applications based on the ASP.NET framework run on Microsoft's web server solution IIS (short for Internet Information System). IIS is simple to configure and maintain and monitor through a graphical user interface, however, it also has a reputation for poor security. Typically, ASP.NET application choose Microsoft's SQL Server as the underlying database, although third-party solutions, including NoSQL databases, are supported by the framework. On the programming side, the ASP.NET framework supports languages that conform to the Common Language Infrastructure (CLI) specifications, including C# and Visual Basic.

While the ASP.NET framework is a powerful and highly configurable solution for web development, it is important to note that is is underlying several restrictions: Micosoft as the underlying platform and IIS as the operating web server are without any alternative when choosing .NET. Furthermore, licencing costs for commercial applications as well as costs for add-ons and third-party tools can be high.

There are several extensions that make it possible to use the .NET for mobile development. While the regular ASP.NET framework is suitable for mobile websites, a particularly interesting solution for mobile apps is Xamarin (formerly called MonoTouch). Xamarin is a cross-platform implementation of the .NET framework, allowing to write mobile apps for iOS and Android Windows Phone using C#. Apps developed with Xamarin can share the same business logic, data access, and network communications code while native platform APIs can be called directly through the C# code, enabling full access to the device's specific features and therefore offering high performance paired with a native user experience.

* + - 1. Backend as a Service

In traditional software development lifecycles, building a back-end takes up a great chunk of time, often accounting for up to or even more than half of the total development time. With cloud computing becoming more and more ubiquitous throughout the last couple of years, a new trend in the world of software development emerged: Backend as a Service (BaaS). With a cloud-based out of the box back-end, providers of BaaS offer clients to exchange the traditional web server stack for a simple yet powerful solution that can be customly tailored to fit individual needs. Commonly used features that can be added include user management, location services, push notifcations and integration for social networks. Especially in the mobile sector, choosing BaaS may often be a advantageous business decision, saving developers a lot of time and enabling them to ship their apps faster then potential competitors. Amongst the many BaaS providers that are on the market today, the most popular ones (according a post[[15]](#footnote-15) by Antonio Martinez, published in November 2012 on the blog of my personal iOS guru Ray Wenderlich) include StackMob, Appcelerator, Parse, Applicasa and Kinvey.

* + 1. Front-end technologies
       1. Web technologies
          1. HTML, CSS & JavaScript

HTML (Hyper Text Markup Language) was developed by Tim Berners-Lee around 1991 and is still actively evolving through the work of of the World Wide Web Consortion (W3C). HTML is the markup language that is used client-side for building web pages. It describes the content and structure of a page through a set of markup tags. Web browsers are designed to read HTML documents, interpret their content and display them to the viewer.

While HTML can also contain style rules, it is highly encouraged by the W3C to strictly seperate markup and style definitions and use CSS documents for everything concerning the visual design of a web page. CSS (Cascading Style Sheets) is a style sheet language that defines the rules of how HTML elements are displayed and formatted.

HTML5, the latest version of HTML, added several new features such as tags for video, audio and canvas elements as well as support for 3D graphics. In combination with CSS3 and JavaScript, it serves as the foundation of a new dimension of rich, interactive user experience in the world wide web. Supplementary JavaScript libraries such as jQuery offer additional features such as animationed effects and support of AJAX techniques for asynchronous server communication.

* + - * 1. Flash

Flash is a technology released by Macromedia in 1995. After Adobe acquired Marcomedia in 2005, it was developed and distributed as Adobe Flash. With the use of ActionScript, a fully object-oriented programming language for developing Flash applications, it is well suitable for creating complex, feature-rich internet applications such as interactive, animated websites.

For a long time, Flash was the standard technology for multimedia-driven websites. However, with the emergence of HTML5 and its large spectrum of new features, the use of Flash in websites is seeing a steady decline. This may also partly be accounted for by the fact that Apple's mobile devices iPad and iPhone, which have become increasingly popular over the past few years, do not support Flash. Today, Flash is mostly used for video playback and the development of online games. Adobe has addressed the shift in technologies for interactive web development by announcing to set its focus on specific market segments. In its "Roadmap for the Flash runtimes", a white paper initially published in February 2012, Adobe draws a clear outline for the future of Flash:

*"With the growth of competition in the browser market, browser vendors are increasingly innovating and providing functionality that makes it possible to deploy rich motion graphics directly via browser technologies, a role once served primarily by Flash Player. Increasingly, rich motion graphics are being deployed directly via the browser using HTML5, CSS3, JavaScript and other modern web technologies. Adobe expects that this trend will continue and accelerate, and Adobe will continue to play an active role in this space.*

*Adobe believes that the Flash runtimes are particularly and uniquely suited for two primary use cases: creating and deploying rich, expressive games with console-quality graphics and deploying premium video.*

*This focus does not mean that existing content will no longer run, or that Flash cannot be used for content other than gaming and premium video. However, it does mean that when prioritizing work, gaming and premium video use cases will take priority."[[16]](#footnote-16)*

* + - 1. Mobile technologies
         1. Native apps

The two big players in the world of mobile devices are Google and Apple. In the 4th quarter of 2012, a ranking published by the Intenational Data Corporation (IDC) amounts the total worldwide market share of both platforms combined at over 90%[[17]](#footnote-17), with Google's Android OS being the clear leader, running on 70.1% of all smartphones shipped in late 2012, while Apple's iPhone makes up 21% of sold devices in the same time frame.

Android applications are natively written in Java using the Android SDK and can be developed on any platform including Linux, Windows and Mac OS. Development tools are available for free and there is no special licence required to create and publish apps.

iOS apps are written in Objective-C using the iOS SDK. To create native apps for iOS platforms that can be published in the Apple App Store, developers must compile their app on the official iOS SDK running on Mac OS X. While development tools such as the IDE Xcode are available for free, a paid licence (currently $99/year) is required to test apps on real devices and publish them for App Store release.

Native apps, as opposed to web apps (see 1.3.2.2.2.), have the advantage of having full access to the specific device's capabilities such as the camera or accelerometer. They are downloaded and installed directly onto the device, therefore they can be used without necessarily requiring the user to be connected to the internet, provided the app itself does not need to access remote servers to work properly.

The obvious disadvantage of a native app is that it will only work on the respective platform that it was developed for. This means that if a business wants to release an app and has decided for a native approach, it will either have to target just one specific platform, or develop the same app in several different languages for all desired target platforms. An alternative to this is the use of a hyprid approach (see 1.3.2.2.3).

* + - * 1. Mobile web apps

Mobile web apps are apps that run in the web browser of a mobile device, instead of being downloaded and installed directly onto the device. They therefore require the device to be connected to the internet. Web apps have the clear advantage of being platform-independent, since all they need to be accessed is a mobile web browser, a core component of all web-enabled smartphones.

Since web apps are not published in app stores or marketplaces, developers can distribute and update them on their own terms without having to go through complicated approval processes that can sometimes hold back the release of an app for a considerable amount of time. On the other hand, not being listed in official stores may have the disadvantage of being difficult to discover – publishers of mobile web apps will have to find other ways of promoting their app to new users.

The new features of HTML5, CSS and JavaScript, along with additional frameworks if required, offer means for a fairly good imitation of platform-specific UI elements that can create an almost native look and feel of a well-designed web app. However, current mobile web browsers are only able to access a limited set of a device's specific capabilities. While properties such as orientation and geolocation may be available, a device's camera or accelerometer are currently not enabled in mobile web browsers.

* + - * 1. Hybrid apps

Hybrid apps combine the advantages of both native apps and web apps. A hybrid app uses a web view control as a native wrapper for a web-based application. Frameworks such as PhoneGap, Appcelerator Titanium (both JavaScript-based) and Adobe AIR (ActionScript-based) offer developers an abstraction layer that exposes native APIs (and therefore full access to the device's capabilities) to the app. Hybrid apps can also access the device's file system for caching data, making it possible to design the app in a way that it can be used when the device is offline.

Since hybrid apps are deployed natively for each target platform, they can be published to app stores and marketplaces just like any fully native app. This solves the problem of difficult discovery of mobile web apps.

Hybrid apps are a viable solution for cross-platform development and will likely see a sharp increase in popularity in the future. In February 2013, analyst house Garter predicted that "more than half of mobile apps deployed by enterprise by 2016 will be hybrid"[[18]](#footnote-18).

1. The Original Boatload of Knowledge Down the Ohio River: William Maclure's and Robert Owen's Transfer of Science and Education to the Midwest, 1825-1826, DONALD E. PlTZER, Department of History, University of Southern Indiana, Evansville, IN 47712 [↑](#footnote-ref-1)
2. The Community Exchange System - Reinventing Money, by Tim Jenkin, December 2011. Published in Digital Development Debates, December 2011

   http://www.digital-development-debates.org/issues/06-innovation/transparency/reinventing-money-a-community-exchange-system-from-south-africa-conquers-the-world/post/comment/list/ [↑](#footnote-ref-2)
3. "TEDxSoMa - Leah Busque - From Social Networking to Service Networking" – http://www.youtube.com/watch?v=r4P63-8uS58 [↑](#footnote-ref-3)
4. TaskRabbit info graphic - https://www.taskrabbit.com/core/assets/press/TaskRabbit\_Infographic.pdf [↑](#footnote-ref-4)
5. http://w3techs.com/technologies/overview/web\_server/all [↑](#footnote-ref-5)
6. http://w3techs.com/technologies/overview/programming\_language/all [↑](#footnote-ref-6)
7. http://w3techs.com/technologies/history\_overview/programming\_language/ms/y [↑](#footnote-ref-7)
8. http://me.veekun.com/blog/2012/04/09/php-a-fractal-of-bad-design/ [↑](#footnote-ref-8)
9. http://fabien.potencier.org/article/64/php-is-much-better-than-you-think [↑](#footnote-ref-9)
10. http://www.google.com/trends/explore#q=perl,python,ruby [↑](#footnote-ref-10)
11. http://w3techs.com/technologies/overview/programming\_language/all [↑](#footnote-ref-11)
12. https://www.djangoproject.com/ [↑](#footnote-ref-12)
13. http://w3techs.com/technologies/overview/programming\_language/all [↑](#footnote-ref-13)
14. http://w3techs.com/technologies/overview/programming\_language/all [↑](#footnote-ref-14)
15. http://www.raywenderlich.com/20482/how-to-choose-the-best-backend-provider-for-your-ios-app-parse-vs-stackmob-vs-appcelerator-cloud-and-more [↑](#footnote-ref-15)
16. http://www.adobe.com/devnet/flashplatform/whitepapers/roadmap.html [↑](#footnote-ref-16)
17. http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.UT0kADkqZos [↑](#footnote-ref-17)
18. http://www.techrepublic.com/blog/cio-insights/hybrid-mobile-apps-the-future-of-app-development/39749785 [↑](#footnote-ref-18)