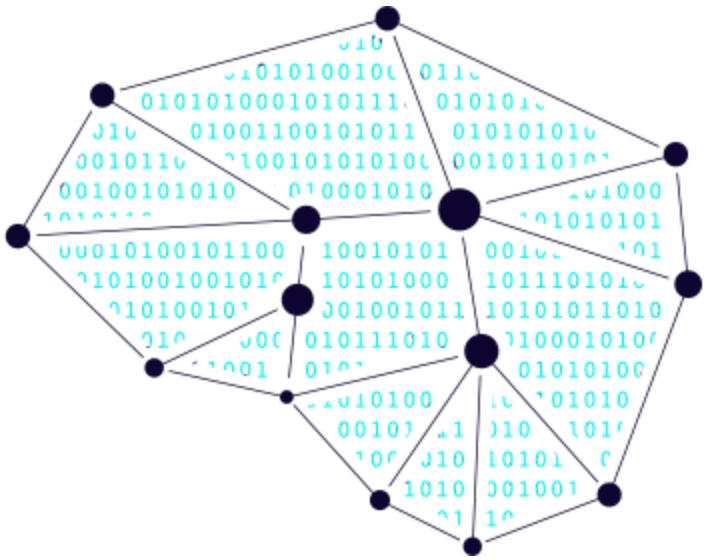

μtask

Welcome to the uTask guide to technical jargon and terminology!

The purpose of this guide is to teach you some concepts and jargon so you feel grounded and centered when dealing with coders via our platform. If you ever feel like you would like some help understanding, feel free to either consult this guide or reach out to one of our customer success reps. Really, we're here to help!

Table of Contents

Agile

There are basically two schools of software development. The old school was called “waterfall”.

A product team came up with a coherent vision for a product, with a complete, fixed specification for the entire product. Then, the team spent a year or so building it, and then shipped it by printing it onto a CD Rom and packaging it for distribution through distributors. As the years went on, the cycles became shorter, like the span of a few months. And instead of CDRom, software started being delivered over the Internet. But the bottom line is that it was slow, and startups decidedly *don’t* work this way.

The new school is called “agile”. Since companies can distribute their own software (through “software as a service”, or SaaS), either on their own infrastructure (server farms), through app stores (App Store, Google Play), or through shared server space (the cloud), they can deploy and make changes much faster. This has resulted in ethoses like “move fast and break things” (-Facebook) and the development of new engineering techniques for

continuously updating one's tech. These buzzwords are "continuous integration" and "continuous delivery".

Whereas a "Waterfall" team might have a year-long development cycles, "agile" teams' cycles are much shorter and tend to be two to four weeks long. This usually consists of a "sprint planning session" (where the team meets up to decide what tasks should be done), a work cycle, where team members indicate what work is in progress through the moving of "cards" in through a "kanban board", and then ideally, a "retrospective" at the end to discuss what went right and what went wrong.

Unfortunately, very few teams actually adhere to best practices and "retrospectives" are rare in practice.

Kanban board from the TV show, Silicon Valley

Standup Meetings

A key feature of an agile team is a daily "standup" meeting, wherein one person talks about:

1. What they did yesterday
2. Where they are "blocked" – i.e., relying on support from another member of the team
3. What they are working on today

Ideally, this is just a quick status report meeting. It's "standing up" so we get through it quickly. In practice, few managers know how to effectively run a standup meeting, so time is wasted as people go in depth on topics during the group's time. However, when standup works correctly, you can really keep your finger on the pulse of the group's output and whether you are going to complete all the goals you set out to during your sprint planning.

Agile Tools

Agile teams tend to use the following software programs to keep track of their tasks: Asana, Trello, Redmine, and Atlassian JIRA.

Agile Term: "Ship it"

"Release it to the public"

The Cloud

You have probably heard the term “the cloud” a lot and are wondering what it means. The answer is servers. Back in the day, companies would have to buy their own server to deploy their software services. This would be really expensive, an intensive capital cost, and created a barrier to entry: only funded startups could afford servers. Not only was there the cost of the hardware, but there was the cost of maintaining it (sysadmins, techs).

Nowadays, companies like Amazon, Microsoft and Google have gone ahead and bought tons and tons of servers and put them in their own data centers. In doing so, they have achieved Economies of Scale: they pay less per computer than we would. So, it makes sense to rent someone else’s servers rather than buy your own.

“Cloud” is just an idea of reducing costs by renting other people’s computers rather than your own. If you are “deploying in the cloud”, you are hosting your website on rented computers (probably from Amazon, Microsoft or Google). If

you are “running analytics in the cloud”, you are borrowing computer time (probably from Amazon, Microsoft or Google) in order to do your computation – you don’t actually own the computers.

Cloud = other people’s hardware.

Java vs. JavaScript

These are two different languages. You will permanently lose credibility in a developer's eyes if you refer to Java as JavaScript or vice versa. It's petty, but we're here to help you avoid the faux pas.

Java is mostly used for enterprise software (big companies), Android app development, and server software.

JavaScript is used for website development (controlling the front-end logic and animations of a website), server development (the post popular framework is Node.js, but JavaScript is frequently used in "serverless" apps that run entirely on the cloud), and mobile app development. JavaScript is frequently combined with the framework React Native in order to build mobile apps that work on both iPhone and Android, without having to involve two separate code bases.

Clients & Servers

In computing there are two important concepts: the “client” and the “server”. The server is the computer that stores all the information and the database and does processing. The client is the computer that interacts with the user.

A web browser is a “client”. An iPhone app is a “client”. These “clients” talk to servers.

Front End Developers

A “front-end” developer is someone who writes code for web browsers, such as Chrome, Safari, Firefox, Internet Explorer, and the browsers (Safari or Chrome) on people’s mobile phones. Frontend developers will know JavaScript, CSS, and HTML. Sometimes they will also know frameworks such as SASS, haml, LESS, Backbone.js, Angular, React, Vue.

Back End Developers

A “back-end” developer is someone who writes code for servers. These languages include: JavaScript (yes, also can be used as a backend language), Java, C#, PHP, Python, C++, C, and Ruby.

Most often, back-end developers write code in a framework. For Ruby, the most popular framework is Ruby on Rails. For PHP, it’s WordPress, Laravel, Symfony and CodeIgniter. For JavaScript, it’s Node.js. For C#, it’s .NET. For python, it’s Django.

Frameworks

A framework is a set of tools that have already solved problems that constantly come up in development. It's like baking a cake from a mix instead of from scratch. But this metaphor falls short because nobody ever looks down upon a developer who doesn't "bake from scratch"; however, inventing a new framework is akin to "baking a cake from scratch". But making a new framework catch on is much rarer than successfully making a tasty cake!

Database Developers

A “database” developer knows how to talk to databases. This person will speak both the languages SQL (using solutions such as MS SQL Server, PostgreSQL, MySQL) and “noSQL” (using solutions such as MongoDB). Many backend developers are also decent database developers.

Mobile Developers

A “mobile” developer is someone who writes code for people’s phones. Mobile developers usually write Java (Android), .NET, Objective-C or Swift. These days, it is also possible to use JavaScript to write mobile code, by leveraging frameworks such as Ionic and React Native.

QA/QC

There are two types of QA, or Quality Assurance, developers. There are "testers", who catch bugs by testing the website in a "*Staging*" environment (the version of the company's tech that's rolled out only to the company itself) before finding them in a "*Production*" environment (the version of the tech that everybody uses).

A good QA will thoroughly document bugs (deviations from expected behavior) with lots of screenshots and information about how the bug was encountered (steps needed to reproduce).

There is a second type of QA. These QAs will build software and write tests to catch bugs in an automated fashion. These are called "QA Engineers". In many software teams, all engineers are expected to write their own automated tests as well.

QAs like to use the software “Selenium” for automated testing of the user’s experience.

DevOps

Devops is a developer who deploys your app, either to an App Store or onto the Cloud. In the hierarchy of developer skill, “DevOps” is generally perceived to be above “QA” but lower than “Front End”/”Backend”/”Full Stack” engineers. This does vary somewhat from company to company, as a Principal QA Engineer will generally be considered higher, in corporate ranking and in pay, than a junior front end developer.

Full Stack Developer

Sometimes developers are both “front-end” and “back-end”. If they can do this, and they can also do basic deployment (“devops”), then they are considered a “full stack” developer because they can build applications soup to nuts.