Developer 101

Types, Technologies & Tips

stack overflow TALENT

Table of Contents

stack overflow TALENT

01

Why Knowledge is Vital for Successful Tech Recruiting

The Increasing Demand for Developers Means Recruiters Need to Up Their Game.

The developers that you're recruiting don't expect you to have expertise in the skills that you're recruiting for. Otherwise, you'd be competing with them for their job. But you do need to know enough to speak to the skill set in the job requirements. Otherwise, your job postings and candidate communications will not reflect the requirement of the position and fail to attract the candidates you need.

But with a little research, you can get enough familiarity with the tech skills and the engineer mindset to talk to them about the roles you recruit for. And this guide will help you get going.

This guide has two goals. First, it provides a roadmap to gaining the knowledge about the technology that you recruit for so you can talk about what your company needs. Second, it introduces you to the developers themselves and the many specializations within the field. There's even a bonus section for advanced recruiters that talks a little turkey about the tech.

This guide is a starting point to understanding developers and how they operate. With this start, you should be able to fill out the gaps in your tech knowledge by talking directly to the technical managers with the open reqs to fill.

02

4 Tips for Building Knowledge in Tech Recruiting

1. **Know What You Don't Know**
   The first step to gaining knowledge is to know what knowledge you lack. At the beginning, it may seem like that includes everything. And that's ok! Everyone starts somewhere.

   The best thing that you can do is not try to hide your ignorance. The more comfortable you are in saying "I don't know," the better you'll be able to gain new information. Developers love to share information about the technology that they love, so not knowing something is an invitation for them to share.

2. **Ask specific questions**
   On your road to knowledge, you'll need to ask a lot of questions, especially "What is it?" But whether you're talking to a candidate about a role or to tech leads looking to round out their team, questions will help you understand the specific requirements and skill involved in the roles you're filling. Over time, you can make your questions more and more targeted—for example, by asking whether the mystery database they worked on is relational.

3. **Work closely with your in-house developers**
   Your company's developers are the experts in the skills that you're trying to hire for, so lean on them for information about those skills. Keep lines of communication open internally, even informally, and you'll be able to unravel any mystery tech that comes your way. If you have an internal chat system, lurk in #developers or #engineering to see what they're talking about.

4. **Keep reference material close by**
   You don't need to have every bit of information perfectly memorized and ready to recite on command. But you should have some handy reference materials around for those moments when you come up empty handed. And we even put together a Tech Glossary to help you out.

03

12 TYPES OF DEVELOPERS YOU SHOULD BE ABLE TO IDENTIFY

Before we talk about types of developers, let's establish what a developer is. The simplest version is: A software developer is someone who uses computer programming and code to solve business problems. Within this definition, there's a lot of leeway; some developers focus on front-end web paradigms that produce visual interfaces, some focus on the underlying logic that makes these work, and some use code and automated scripts to manage production infrastructure.

The key here is that all developers code, whether they learned formally at a university or on their own time using online resources. If you're hiring developers for roles where they don't get to write code, like Architect, Manager, or QA roles, make sure that this is something they are looking for. Otherwise, they may find themselves missing their days writing code, and you may find yourself looking for another candidate to fill their role.

We're using the terms 'engineer' and 'developer' synonymously. Some people will tell you that there is a difference, but that difference is not universally agreed on.

Here are some of the most common "types" of developers.

Backend Developer
Specializes in complex functional logic and performance of software systems. They create systems that process and store data, perform complex algorithmic calculations, and manage synchronous and asynchronous job schedules. Their skillset may include Java, C++, and other general purpose languages, database systems, and some cutting edge software engineering languages, like Rust or Go.

Frontend Developer
Creates the user-facing graphical interfaces that most of us interact with every day on the internet. This includes both the layout and design and the events that occur when you use the interface. They will be familiar with CSS, Javascript, and any number of front-end frameworks and libraries, such as React or JQuery.

Full Stack Web Developer

This developer can do both the frontend and backend development to create an end-to-end web application. This means they will have the skillsets of both types of developers. Why aren't all developers full stack? Specialization. While a full stack developer reduces the friction between system designs, they lack the deeper focus in one area. It's a trade-off.

If you are finding that the definition of what skills a full stack developer is expected to know, this post offers a more advanced dive into the discussion–which we are all part of.

Desktop Developer

Develops software that installs and runs natively on individual desktop and laptop computers within an operating system, such as Windows, Apple OS X, or Linux. They will use many of the same languages as backend developers, but with SDK and graphics libraries, such as XAML, Cocoa, or .NET. The applications that they develop will compile to a binary file that can be executed on the user's system.

Mobile Developer

Develops apps for various handheld, mobile operating systems, like iPhone or Android. May be a specialized type of frontend developer who uses mobile technologies instead of web ones, or they may create standalone apps like a desktop developer.

Graphics Programming

Within video game and special effects productions, graphics programmers handle the complexities around rendering models, lighting, shading, particle effects, visual occlusion, and sometimes physics effects on the rendered models. Because of the complexity of this rendering, these developers will have experience integrating and optimising technologies with very specialized purposes.

Database Engineer/Administrator

Database Engineers and Administrators manage the database system that stores and retrieves the data that an application uses. For some data-driven organizations, they may also deal with the data pipeline that transforms production data into analytics data. These engineers will deal with data table design, performance optimization, and data replication. Technologies used include SQL implementations like MySQL or PostgreSQL, more exotic NoSQL databases like MongoDB and Cassandra, and some programming languages like Python.

Data Scientist

Data scientists use programming to generate insights from large data sets. As such, they often come from statistics and other non-computer-science backgrounds. They use machine learning, statistical analyses, and predictive modelling to harness the power of the enormous amounts of data generated in networked applications. For these tasks, they'll use R, Python, Spark, and other specialized tools.

DevOps/Site Reliability Engineer (SRE)

As compared to most developers who write code to create production applications, DevOps/SREs write code to make sure that an application makes it safely into production and stays available. They focus on infrastructure and automation, both making other developers' lives easier and ensuring that the application's users have access. This includes things like build processes, continuous integration/delivery, resource provisioning, and infrastructure resiliency. The technologies for these tasks are ever changing, but include cloud provides like AWS and Azure, gradle, Spinnaker, Git implementations, and log management systems like Splunk.

Quality Assurance Developer

These developers ensure that the code other developers create doesn't break the application and performs as expected. While QA folks used to manually use the software to see if something went wrong (and many still do), modern QA Engineers often try to automate their tests. They code processes that simulate usage, both for individual features (unit tests) and for a more holistic view of that feature (integration tests). QA is the last line of defense, so they are detail oriented and methodical.

Developer for Client Relationship Management (CRM)

These developers customize and configure complex enterprise software that store client and overall business information. These systems include CRMs like Salesforce, Enterprise Resource Planning (ERP) systems like SAP, and collaborative document storage like Sharepoint. Their code optimizes sales and product processes, leading to improved sales and customer satisfaction.

Embedded Developer

While the developers discussed above generally work on software, the embedded developer writes for various hardware systems, programming microcontroller chip behavior, embedded firmware, and other devices, sometimes even coding on bare metal—systems without an operating system between the hardware and the code. These developers primarily use C/C++. As microprocessors and other chips with embedded logic become a part of more and more daily appliances, the demand for this role will only increase.

04

Bonus chapter for advanced readers: Talking Tech with Candidates

Getting the Names Right

Now you should have a sense of who developers are and what they do. But if you think developers vary wildly, wait until you see the technologies that they use. Different developers have different preferences, and the tools for one job may not have all the features needed for another. Plus, if a developer doesn't like an off-the-shelf tool, they can always build their own.

You may hear some on your team refer to a tech stack. Tech stacks are what a specific developer or organization uses in conjunction (or, if you prefer, "stacked" on each other).

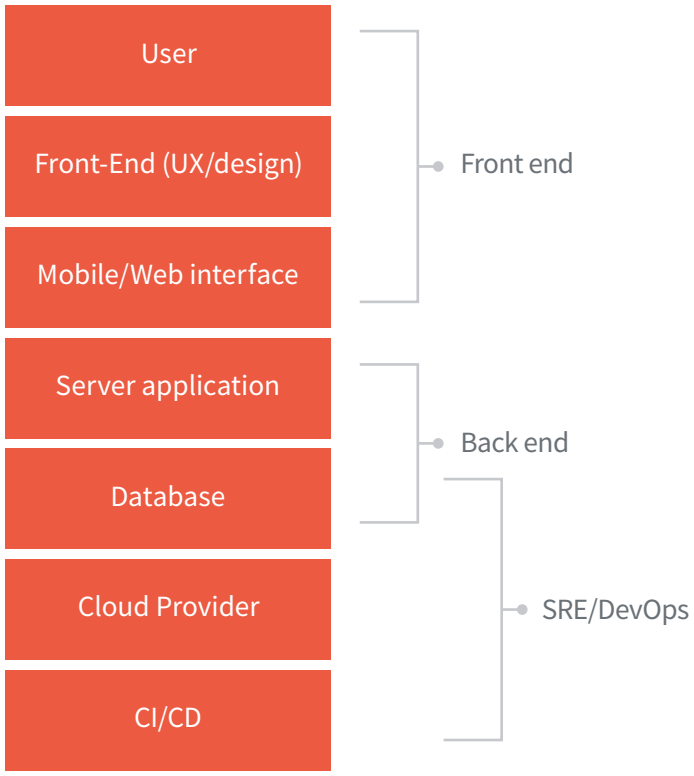
For those with limited technical knowledge, this can all be confusing. How do programming languages and operating systems interact? What are relational vs. non-relational databases? And what's with all the frameworks and libraries? Don't worry, this section will walk you through a few ecosystems and some of the more commonly combined stacks.

Tech Stack: The Team's Toolbox

When a team or organization talks about their tech stack, what they're really talking about are the technologies that work together to create their product. For most networked applications, the technologies in a stack will include a backend language, a frontend framework and/or mobile technology, a database, and possibly a cloud provider, web server, and/or content delivery network (CDN). For installed applications, they'll use an operating system and associated SDK, a programming language, a graphics library, and possibly a database.

There are a lot of technologies in each of those categories, and the lists change all the time as the landscape shifts. And this doesn't include all the smaller tools, like analytics, testing, and CRMs. As a company grows, so does their tech stack, especially if their technology org allows developers to choose their own tools.

Tech Stack



Types of technologies

Programming languages

Literally, the text that determines how the developer constructs their code to create an application. Some languages compile to a binary file; that is, they are processed into a machine readable state and made directly executable on a host computer. Some compiled languages are cross-platform and can be run through a virtual machine on any operating system. Some are interpreted; that is, the code is executed in real time without compilation.

- C/C++
- C#
- Java
- Python
- Go
- Scala
- JavaScript

Operating Systems (OS)

The basic software interface between the hardware and the applications that you use daily. Most OS provide an immersive graphical user interface (GUI) to make configuration and management easier. An OS will handle things like file storage, input/output, and memory/processor management.

- OS X
- Microsoft Windows
- Linux
- FreeBSD

REST APIs (Application Programming Interfaces)

URLs that accept and return data, which can be called and incorporated into a software program. In networked applications, APIs allow you to incorporate specialized functions into your feature set without writing the code yourself. Each API is connection to a program that you can call to retrieve or process data.

Software/Native Development Kits (SDK/NDK)

Specialized binary files that can be incorporated into an application intended to run natively on an operating system. They typically provide the building blocks of a development environment (e.g. a compiler) or implement specific features (e.g. performance monitoring). NDKs are a special subset that allows an application to run on a specific operating system, particularly used for mobile platforms. The difference between an SDK and an API is that an SDK must be incorporated wholesale, but doesn't require a network connection to function so runs much faster.

Frontend Frameworks

Provide extra power and convenience for complex behaviors of web interfaces. They build on an existing web language—JavaScript, Ruby, PHP, C#, Python—and make it easier to develop a robust web application. The downside to a framework could be performance issues and the fact that the landscape for frameworks is shifting rapidly.

- Angular
- React
- Vue.js
- Ember.js
- Backbone.js

Databases

Store and retrieve the data needed and gather by an application. Data can be structured—have specific field-value pairs—or unstructured. The database itself can be relational—a series of two-dimensional tables linked through relational cells—or non-relational, also called NoSQL. SQL is primary a language to query databases, not a database itself.

- Oracle
- MySQL
- Microsoft SQL Server
- Apache Cassandra
- MongoDB

Cloud Providers

Provide scalable computing resources in a networked environment. These are referred to as serverless computing or "the cloud" because their location isn't tied to a specific machine; in fact, a single cloud may operate in multiple data centers simultaneously. Computing resources are provisioned as needed, which makes a lot of sense for large internet applications that have spikes in traffic over a day or year.

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Alibaba Cloud

Tip: Don't underestimate the impact your technology has on general job satisfaction. Out of the 79,000+ developers questioned in our developer survey, 54% stated that the technologies that they'd work with would be the most important factor in their job satisfaction.

The 10 Most Popular Technologies You Meet in Development

The following are the 10 most popular technologies as ranked by our 2019 developer survey. These are in order from most to least popular.

JavaScript

A specialized programming language for the web and, along with HTML and CSS, one of the core technologies that comprise the web. It's an interpreted language, so code is directly executed within a user's browser. Most web applications now use a framework that uses JavaScript to perform web application specific functionality. Applications without a framework are referred to as Vanilla JS. JavaScript has become so pervasive that some backend systems use it, like Node.js.

HTML/CSS

The other two foundational web technologies. HTML is a markup language, where effects are created in a browser by placing tags around other text. CSS adds powerful styling within HTML. Anyone creating anything on the internet will eventually use one or both of these technologies, often in conjunction with others. Think of HTML as the content of the page, while CSS determines how that content will be presented. CSS tools—called CSS preprocessors—make it easier to write CSS. Popular CSS preprocessors include Sass, Less, and Stylus.

SQL (pronounced "sequel")

A database query language; it stands for Structured Query Language. While it is primarily used to search structured databases, SQL implementations like Presto allow querying large NoSQL datasets as well. SQL is the dominant database query language.

Python

An open-source interpreted language used in a large number of contexts, from web applications to data science. Because of its versatility and ease of use, Python is constantly growing in popularity.

Java

A compiled programming language that can run on any platform that has a Java Virtual Machine (JVM). While Java has a syntax similar to C++, it's become more popular thanks to its use in client-server web applications. Java applications can be embedded directly in web pages, while Java itself is often used as a backend web language.

Bash/Shell/PowerShell

OS-specific types of command-line scripting languages. Many developers love to work on a command line interface, especially with tools like Git and curl. Bash and the other two let developers create scripts of multiple commands to run in a single go, either manually typed, on a chronological schedule, or triggered by external events.

C# (pronounced C Sharp)

An object-oriented language developed at Microsoft as part of their .NET initiative. It's general purpose and easy to get into for developers who know Java or C++. Because it is a modern update on C programming methodologies, it is very popular for Windows desktop applications, especially video games. The popular Unity game engine uses C# as its scripting language.

PHP

A programming language initially designed as a server-side (or backend) language. Short for PHP: Hypertext Processor, PHP code is processed by an interpreter before a web page loads and replaced by the processed HTML, CSS, and Javascript. While PHP is less popular than it was a few years ago, it's still very popular for the ability to embed PHP code directly into HTML and have it rendered based on the context.

C++

For a language created in 1979, C++ remains pretty popular today. It's a compiled language with a focus on performance, efficiency, and flexibility. Even though there have been many other languages developed to improve it, C++ itself is still in active development and used for a number of applications compiled to binary files and executed natively.

TypeScript

An open source superset—that is, it's JavaScript plus additional features and syntax—of JavaScript developed and maintained by Microsoft. It transcompiles to JavaScript; instead of the result being an executable binary file, it's still JavaScript code. Originally designed to overcome the shortcomings of JavaScript, Typescript provides features like type checking that help catch errors and more easily develop robust applications.

C

The progenitor of modern programming languages. It originally provided a cross-platform way of interacting with low-level memory and processor functions. Where many more contemporary functions have a lot of helper functions built into the compiler, such as garbage collection, C has few safety features and maps pretty closely to assembly language instructions. As such, it's often very fast, but has no in-built safety measures. C is popular in embedded systems programming, programming for firmware on hardware.

Access to the Developer Community and Tech Trends

As the largest developer community in the world, we're the first place most developers look for answers to their programming questions.

Over 50 million developers visit our site every month. So we have the pulse of what technologies are trending now and how employers can connect to the right candidates.

Stack Overflow Talent offers comprehensive recruiting tools, from job postings to job searches to employer branding strategies as well as personal support that will, along with these tools, help you achieve your recruiting goals. This guide is only one of many insights that will help you on your path to recruiting expertise.

Talk to an expert ▷