StackExchange

Stack Exchange Developer Compensation

June, 2011

This is a description of how developer salaries are determined at Stack Exchange.

Introduction

The development team at Stack Exchange is an amazing group of programmers who live up to our motto of "smart and get things done" every day.

We want to offer them compensation that is fair, easily understood, transparent, and competitive.

Fair means no games. Our compensation is not based on how well you negotiate or how often you ask for raises—it's based on a repeatable predictable system. There's no forced ranking, so other people don't have to do badly for you to do well. We don't have a *range* of possible salaries for every level, we have a single salary, so everything about the system is algorithmic.

Easily understood means that any developer can figure out what their salary should be according to this system. They can see what they need to do to move up in their career. And different managers can figure out how to pay their team members and get consistent and fair results.

Transparent reflects Stack Exchange's core beliefs about running our business in the open, without secrets. It means that if a list of everyone's salary suddenly appeared on Wikileaks, nobody would be surprised enough to be upset. Transparency is essential to insure fairness.

Despite our belief in transparency, we also believe in our employees' right to privacy. We publish everything about how the system works, but we won't reveal what a given individual earns to anyone but them and their manager.

Competitive means that you're earning at least as much at Stack Exchange as you would earn elsewhere. It's critical to being able to attract and retain the kind of developers we want working for us. If our compensation system isn't competitive, we won't be able to hire

the people we want without giving them an "exceptional" salary, and exceptions defeat fairness.

Components of a competitive salary

The salaries for top notch C# developers around the world can vary from around $30,000 to $200,000.

We've done extensive research on what other companies use to determine programmer salaries in order to make sure that our system is competitive. We've found that market salaries are a function of:

1. **Skills:** how good you are at what you do
2. **Scope:** how much stuff you're responsible for
3. **Experience:** how long you've been a programmer
4. **Company size:** small companies tend to pay less than large
5. **Location:** every geography has its own market salary
6. **Public artifacts:** how much you work in public and how well known you are in the community

Our system takes all of these into account.

Skills

Even when working with an enormously talented group of people like the team at Stack Exchange, there is always room to get more awesome. Skills are re-evaluated annually every summer by your direct manager. See the *2011 Be Even More Awesome Chart* at the end of this document.

Scope

The "scope" category reflects how much responsibility you have. In larger companies it's all about how many levels of managers you have under you.

At Stack Exchange, we don't have strong ownership in areas of the code, and there aren't a lot of "scope" levels, but we do have:

1. **Newbie**—not yet trusted to write code on their own, mostly given smaller tasks and working under close supervision. Tends to go no more than a couple of days at a time without needing more input from a manager. Expected to move up to "contributor" quickly.
2. **Contributor—**writes large chunks of code on their own. Tends to go a few weeks at a time without needing major direction from a manager.

3. **Architect—**designs major systems independently. Leads the design and development of large important pieces of code that take months to build. Proposes and advocates major new features and then drives them to completion. Leads several other developers, either as a manager or peer.

Experience

Programmers tend to acquire skills at radically different rates. There are plenty of 18 year old programmers who can run circles around 20-year veterans. That said, there are bands of experience inside of which the market does tend to compensate differently. For example, companies our size almost always have a single, fixed starting salary for new CS graduates, regardless of skill (probably because they can't really tell "skill" until the kid has been working for a while). After someone has been out of school for a year, skill matters much more in determining salary.

Our bands of experience are:

1. College student or intern
2. 0-1 years
3. 1-5 years
4. 5-15 years
5. 15 years+

We measure experience as *full time software development experience*.

- Time spent doing non-programming jobs doesn't count
- Time spent in college, in coops or internships, or before graduation (for someone who went to college) doesn't count

Company Size

We continuously benchmark our salaries against similar-size companies. Specifically, we look at salaries at other sub-100 employee, VC-funded, pre-profit US Internet startups. We specifically *don't* attempt to benchmark against Google, Microsoft, investment banks, or bootstrapped startups.

Location

Because our team is geographically diverse, we could try to set salaries based on location (e.g. a developer in Kansas City receives a salary that is appropriate for Kansas City) but we don't see any justice in penalizing people who live in the boonies when they are doing the same work. Thus, our salaries are consistent across the world for

work-at-home developers even though this tends to benefit people living in very low cost-of-living places.

Due to the unusually high cost of living in New York City and the high market salaries in NY, and the fact that we don't want to accidentally *encourage* people to move out to the boonies and work remotely, there is a fixed cost of living bump up for New York.

Public Artifacts

One of the core values of Stack Exchange is the fact that we work in public, and creating public artifacts—an electronic, public bread crumb trail of the work we've done—is unusually important to us.

It's a part of compensation because the better well known you are in the industry, the higher a salary you will command.

There are a number of different ways you can create public artifacts. Nobody will do all of them, but doing at least some of these things is a core part of everyone's skill set. They are, in no particular order:

- Participating in Stack Exchange
- Writing blog posts on our blog
- Writing blog posts on your own blog
- Contributing to open source
- Public speaking
- Attending hackathons and meetups for developers
- Participating in panels
- Attending conferences
- Podcasting or being a guest on a podcast

The idea is that you're expected to create these public artifacts as a representative of the Stack Exchange team—speaking at a conference on marigolds without identifying yourself as being from Stack Exchange does nothing for us. The t-shirts are free. Wear 'em.

All of these things increase your market value, thus justifying our paying you more money, and when you do them as a representative of Stack Exchange, they reflect well on us, establishing our reputation as being the kind of place where the elite celebrity developers work, which, in turn, attracts more great developers.

2011 Be Even More Awesome Chart

AAA+++ Guru level of amazingness. Does *and* teaches. You are actually famous for this skill. This will be rare, even on our amazing team. Most people should have no more than one or two of these.

A+ Awesome. Totally, absolutely dominates this skill.

A Sufficient. Completely, utterly able to do this skill enough to accomplish current job.

B Be even more awesome. This is a good thing to work on being more awesome at over the next year.

If you have a lot of B's, don't despair. It just means you're closer to the beginning of your career than the middle and there's a lot of opportunity to grow.

We don't have C's because everyone here is awesome. If you were doing C, D, or F level work, we would have already had a stern talk with you and would be working closely with you to correct it.

Skill	Choose level			
Core programming Loops, subroutines, etc. The basics of programming	B	A	A+	AAA +++
Web programming HTML, CSS, JavaScript/JQuery, Ajax	B	A	A+	AAA +++
Our programming tools C#, .NET, LINQ, SQL	B	A	A+	AAA +++
Other engineering-type skills DVCS, bug fixing, automation, working on teams, etc. The kind of stuff you talk about on programmers.se	B	A	A+	AAA +++
Extracurricular programming learning Learns interesting new technologies for their own sake	B	A	A+	AAA +++
Creates public artifacts Blog posts, open source tools, books	B	A	A+	AAA +++
UI Design Good user experience skills—code is easy to learn and use	B	A	A+	AAA +++
Ships Good artists ship. Skill at getting code to users quickly	B	A	A+	AAA +++
Outreach Hackathons, volunteering, evangelism, public speaking	B	A	A+	AAA +++

Getting Things Done Executes on plan, makes things happen	B	A	A+	AAA +++
Performance and Optimization Consistently works on making the user experience fast	B	A	A+	AAA +++
Data Mining Looks at user data constantly to find ways to improve	B	A	A+	AAA +++
System Administration The important technologies used in our hosting environment	B	A	A+	AAA +++
Ideas Consistently coming up with new ideas	B	A	A+	AAA +++
Modem-whistling Able to emulate a 1200 baud modem using just your mouth (with error correction protocol) well enough to upload a GIF picture to Compuserve.	B	A	A+	AAA +++