**What it's like being a developer**

* A broad overview
* Q&As
* Types of the company (agencies vs SaaS vs eCommerce, and a note on Freelancing)
* What to expect from a tech company
* Continued Professional Development
* Tech culture (about events and participating in the tech community)
* Links to videos

**At a broad level**

A lot of people are under the impression that being a developer is antisocial work, where you wear a hoodie (accurate) and work in a basement (usually inaccurate) and don't talk to anyone (very inaccurate...).

At a high level, coding is:

* largely progressive, with conservative pockets
* different on a social level in the UK and the US
  + salaries are waaaay higher in the US (partly to account for healthcare costs/COL)
  + hierarchy, structure and culture is a bit different
* involves high communication
* more relaxed than other industries

**Q&As**

**How many hours a day are spent actually coding?**

This varies. I've seen companies budget 6 hours a day as productive hours (i.e. you're spending at least an hour/hour and a half getting drinks, being interrupted, etc), but that includes meetings for delivering features.

It varies by company. I've seen estimates between 2 hours and anything more than that (including working out of official hours).

**Is it a male-dominated industry?**

Yes, it still is. Is there a lot of sexism and bias in the industry? It's improving... but there is some; the industry is dominated by abled cishet white men.

Other biases:

* there's a bit of everything (racism, ageism, ableism, etc)
* this exists in all industries
* I've been told it's much better in tech than other industries

There is a LOT of commitment to change and to D&I in the industry, it's very vocal. We're an industry where a lot of prominent figures have their pronouns in their bio on Twitter, there are prominent WiT, disabled, PoC developers and techfluencers who are making change.

The biggest "trap" or stereotype for (particularly) women is falling into "glue" work (i.e. using soft skills to improve a dev team beyond just writing code): <https://noidea.dog/glue>

**Is it ever too late to go into coding?**

Ageism can go both ways in tech.

It's never too late to go into coding.

If you are older, people may try to push you into more senior interviews; just be really clear about your coding level. Although there is sometimes negative bias, generally companies really value the other life experience, skills and maturity you bring to the role (e.g. working to deadlines, working with teams, management, product lifecycles, general professionalism, etc).

**Types of company**

What a "day in the life" and week in the life of a developer looks like varies of course, but there are a lot of things that are similar at similar structures of the company - on a day-to-day level, and in terms of culture and benefits and responsibilities, etc.

**Agencies**

Agencies provide (and sometimes maintain) websites to a number of clients. Sometimes they will also provide developers to a client on a contract basis.

This ranges from:

* delivering static sites (i.e. HTML, CSS, a little JS, no backend)
* building sites on a CMS such as WordPress
* full end-to-end websites (i.e. a backend, a frontend, server stuff, and testing, etc)
* mobile/device applications
* to just providing extra developers for an existing codebase that needs more hands on deck

At an agency, often you are less likely to work in strict implementation of Agile. This is partly because you might receive a project specification, build it from start to finish, present it to the client, make some changes... and never see it again. That's Waterfall.

Sometimes, you will do ongoing work on a codebase, but the quality may be less of a concern than speed.

Often, agencies will care less about CPD as the priority is producing quantity rather than creating maintainable, modern code (if it looks modern to a user it doesn't matter if you built it in jQuery and used tables for positioning... but please don't do that).

Often, you are not client-facing; a sales person or project manager is. You are more likely to have a lead/senior developer who will be asked about timelines etc for a proposed project. You may meet clients; but you may be completely shielded from them.

Pros of working at an agency:

* lots of codebases, lots of tools to learn
* generally you're working fast
* they can be really laid back, and fun fast places to work Cons:
* you're often working to deadlines
* opportunities to learn how to improve code quality may not come as frequently Pro slash con:
* you are more likely to work independently rather than in a coherent team
* mixture of greenfield/brownfield codebases

**SaaS**

SaaS === software as a service. A company that sells/builds software.

Generally part of a larger/established team, or multiple teams. The code that you write is the code you're going to be stuck with for a while; so there's more of an emphasis on quality i.e. reducing technical debt. You will generally not be working with clients, but you will be dealing with a product owner/manager and internal requests.

With SaaS, working at small companies can be really different from working at very large companies.

Pros:

* more likely to provide opportunities to learn how to improve code quality
* more likely to have a mentor/someone to help train you up
* work/life balance is often better (retention is better/more valued at SaaS)

Cons:

* there's sometimes a ceiling of learning new things
* salary progression of staying in the same role/at the same company is lower than moving jobs

Pros slash Cons:

* you have the same stack and technologies for a long time

**Domain knowledge**

Knowledge of the subject of your code. i.e. if you did a website for investments, your domain would be investments.

It is as valuable sometimes as technical knowledge (i.e. the languages and tools you know).

**eCommerce and more**

Some industries have their own quirks.

If you work in eCommerce; you're likely to have to learn an in-house language/framework/library (e.g. Shopify has their own framework), or you may end up only doing minimal frontend work on top of an existing backend structure and existing template websites.

The same as many other industries; public sector often has a lower pay ceiling than private, but the teams are often valuing high-quality work and providing good work/life balance and great pension benefits.

**Freelancing**

You should not freelance for at least 2 years (in general). This is because you should get experience working in the industry and live codebases first, and more importantly, have exposure to working as part of a team with people who can mentor you and help you debug.

If you wanted to provide small static sites, go for it. Stuff like freelance React or frontend development takes more time for you to feel comfortable enough to do at other companies and work with clients.

Additionally, freelancing involves a lot of specific freelancing (non-technical, non-coder) skills to be able to find clients, charge/quote correctly, do the work, manage expectations, complete handover, etc. If you are working delivering static sites rather than working on a dev team, you will likely also be expected to provide design and advice on hosting, SEO, etc.

After a few years, the amount you can charge for freelancing as a React or frontend developer is very very good. Currently, React contractors are being paid between £300-650/d (usually £400ish).

**What to expect from a tech company**

**Company culture**

On a cultural level, there are lots of different vibes at different companies:

* work hard/play hard
* "move fast and break things" - Facebook saying
* work/life balance is the big priority
* "we're all one big family"
* totally professional
* totally casual

You can ask about this kind of thing in an interview, but you're likely to be able to find some information about it on their website. There is no single fit, and you might find your preferences and priorities change throughout your career.

**Moving company**

Generally, people recommend staying in a role for about 2 years before moving, but you can move as early as six months or less in an emergency. You can move after a year few times, if needed. Just bear in mind that people may ask why if it becomes a pattern.

**Benefits in tech**

Most companies provide you with a laptop.

Some companies will let you use any kind of laptop; some use specifically Apple, or Windows, and often there's more flexibility with Linux.

Beyond that, there's a range of potential benefits:

* training budget (sometimes for conferences)
* social events
* healthcare/pension plans
* parental leaves
* cycle-to-work schemes
* sometimes unlimited holiday
  + if you find a company with this, and you get a role there, make sure to use that holiday
  + aim for a minimum of 20-25 holiday days/year
  + you can use it for things in addition to time off, such as conferences, doing volunteer work, for mental health days, etc
* subscriptions
  + training resources
  + medical/mental health apps
  + every so often other stuff
  + gym memberships
* discounts to clients/services (particularly at ecommerce companies)
* employee assistance programs (particularly at larger companies)
* flexible working
  + this usually means you still have to do a full week of work (i.e. 37.5 hours ish), but you only have to be available for meetings during core hours
* remote working
* 4 day weeks/4.5 day weeks/similar

**Career progression**

A good company will have multiple pipelines; it's not inevitable that promotion leads to management.

A lot of people are GREAT developers and abysmal managers, or vice versa. Some companies practice flat structures (i.e. there is no junior-, mid-, senior-, staff-, lead- developer etc titles, only "developer" or "engineer"). But often it will follow roughly that progression ^, potentially going as far as C-suite (CTO - Chief Technology Officer).

Don't worry about the absolute longterm stuff at this point. Although DO investigate and learn skills that you value (which may involve senior technical skills like architecture, or soft skills like giving feedback and mentoring).

**Staying up to date/learning**

Companies generally have a training budget (or other learning benefits). You can often spend this on books and conferences (sometimes they'll have a separate conference budget), and online courses and resources.

Some companies have 20% time; a percentage of your time (e.g. Friday afternoons) is set aside for learning and CPD (continued professional development), too.

Everyone's different; you might learn best from watching videos or from taking time to read the documentation and build things. I learn best from attending conferences and meetups, and trying new technologies as part of my work.

See links at the bottom for good videos of conference talks. I've been to most of the events linked at least once.

**Zeitgeist and events**

As well as CPD, events are good for building community and sense of identity and belonging around coding.

Conferences are often very general. I attended PHP UK for 3 years after I stopped using PHP, because it had such a diversity of topics (and the freebies were so good).

You get lots of talks on all subjects, and the quality of delivery is really good.

As well as the talks, you get:

* tons of freebies (including things like IDE licenses, socks, and Lego Millennium Falcons)
* often really good food
* social events before, during and after
* online communities sometimes
* the chance to take a free trip somewhere, on the company (provided they have conference/travel budget)

You can also use forums and groups and websites to build community:

* hackernoon
* dev.to
* stack overflow
* reddit
* slack groups and discord
* Twitter

**Conference/Meetup Videos**

A lot of conference and meetup talks are recorded! They're a great way to learn new things, and a great place to make friends in the industry.

When watching meetup/conf videos, don't force yourself to watch dry or complicated videos. Watch fun ones at a level you will understand; they don't even have to be technical to be useful.

Generally conferences have talks on all kinds of topics and languages, even if the conference is named and focussed on something specific (e.g. PHP language, JS, leadership, design).

Here are a few videos from different conferences and meetups which I enjoyed:

[How to crash a plane](https://youtu.be/099cHWSbAL8), and [how to make a sandwich (a talk on feedback)](https://youtu.be/P8sNSNkWFpc) ([Lead Developer Conference](https://leaddev.com/))

[Building binary circuitry, using minecraft](https://www.youtube.com/watch?v=ZaL96bIZYqQ) ([Dutch PHP Conference](https://www.phpconference.nl/))

[Learn about git "rebase" using comic books](https://youtu.be/BzITHlQa_VQ) ([PHP UK](https://www.phpconference.co.uk/))

**A couple more channels you may be interested in**

[React London](https://www.youtube.com/watch?v=E-50aGAZG38&list=PLW6ORi0XZU0BL3Up9mXpP75ilJBDOjMsQ&index=4&t=3636s). I enjoyed the third talk in that video - check out the speaker's website for some cool features!

[Bristol JS](https://www.youtube.com/channel/UCPOICck6I3zqGveOs6aFT5A/videos)

**Shorter videos**

If you're short on time, look up "lightning talks" to find 1-10 minute talks.

[This channel](https://www.youtube.com/user/Computerphile/videos?view=0&sort=p&shelf_id=0) has a lot of well-known videos between 5 and 20 minutes long. I really like [this one](https://www.youtube.com/watch?v=-5wpm-gesOY) about timezones. 10m

[Javascript Wat (a classic)](https://destroyallsoftware.com/talks/wat) video. <5m

[Here's one](https://youtu.be/RpcOzsWojNQ) my friend did on github tips and tricks! <10m