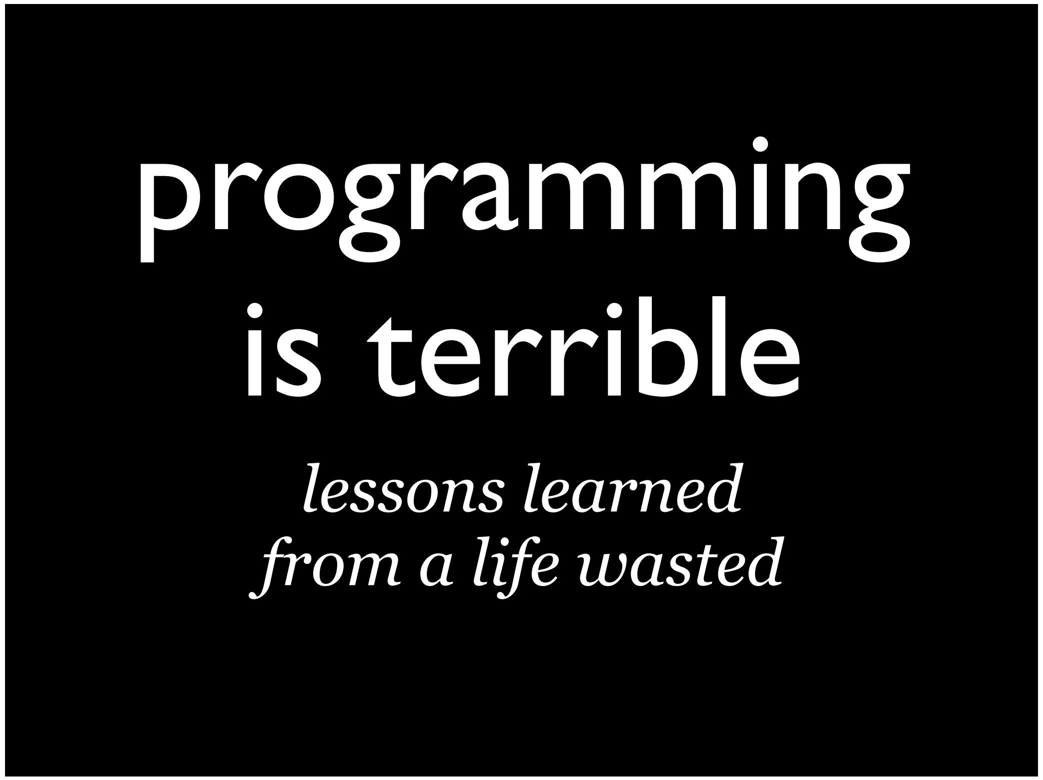
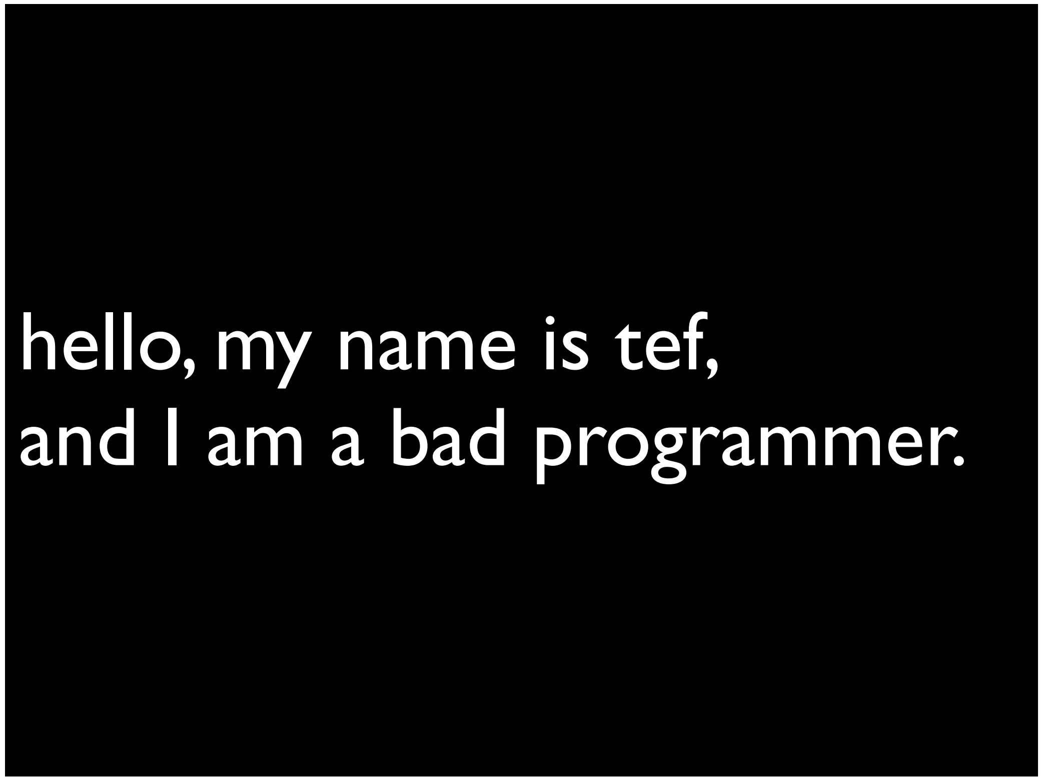
# Welcome



# programming is terrible

*lessons learned from a life wasted*

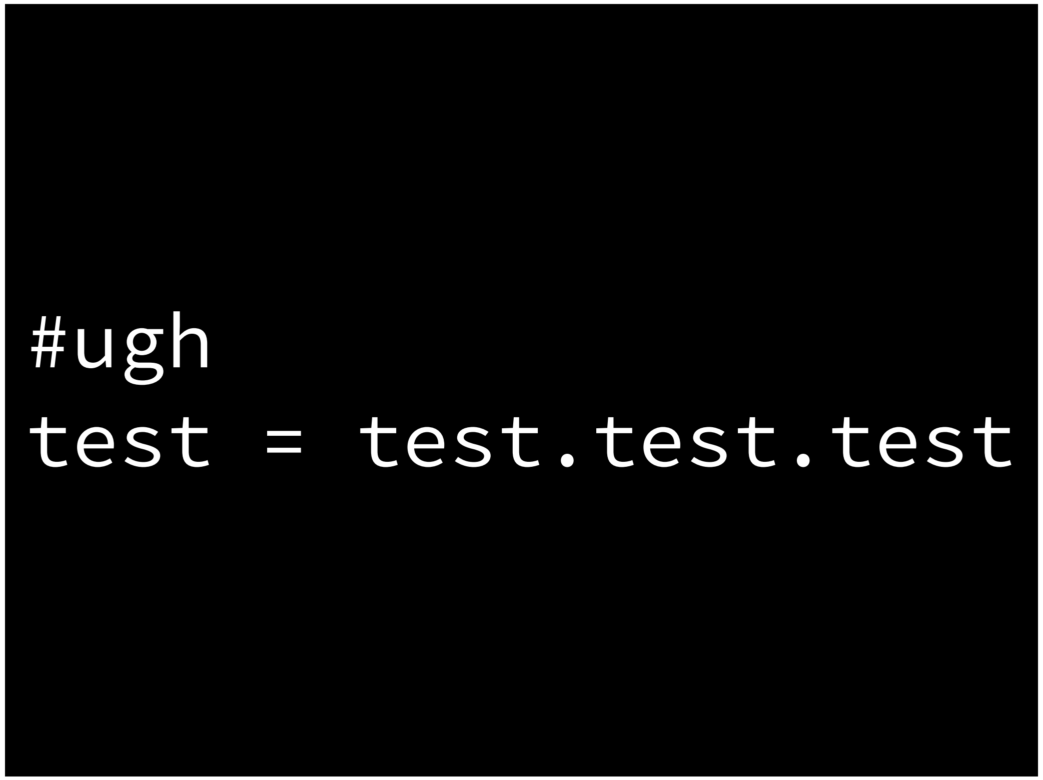
Before I start, let me give you the standard disclaimer. These are my opinons, they might not reflect reality. Although some have found them useful, they've often got me into trouble.



hello, my name is tef,
and I am a bad programmer.

Hi, my name is tef, and I am a bad programmer.  Before I make fun of other people, I should really start with myself:  I'm a bad programmer, and I've written a lot of bad code.

The next slide carries a health warning.



```
#ugh
test = test.test.test
```

This is an actual line of production code I wrote. I finally had the time to clear it up, a couple of months later, but when my co-worker saw this their first instinct was to make fun of me on the internet. Then they saw the comment, and realised that he couldn't make me feel any worse.

If I'm a bad programmer, what makes a good one? I’m not sure, but thankfully, loudmouth programmers on the internet have given us a few ideas.

## Good:
## use my favourite language.

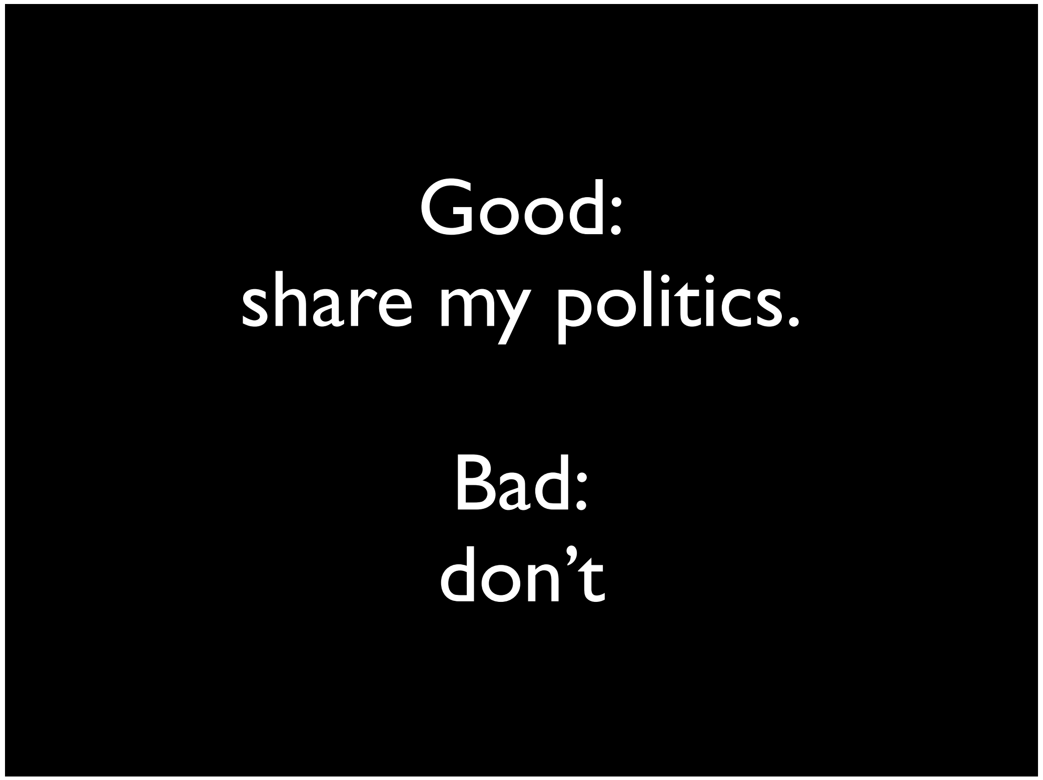
## Bad:
## don’t

One famous lisp programmer wrote an essay asking why other people aren't as smart or as clever as they are, and termed it the blub paradox. He argued that programming languages are a form of 1984's newspeak, a limiter to thought. Sure enough, some programming languages are bad, but then again, some programming languages let you write your own bad language inside, at the expense of everyone else.

Meanwhile, the company who bought his lisp code asked "Why can't we find anyone to maintain this". His code was very clever, so clever in fact that they couldn't find anyone else to understand it, and decided to re-write it in perl to make it cleaner and maintainble by mere mortals.

Then again, the person who came up with this argued that terrorist attacks wouldn't happen if planes were more like lisp, which at once demonstrated an ignorance of terrorism and computer security.

(You can still find it on his website, he's hidden it away, but his immediate reaction upon seeing the tragedy of september 11, 2001 was "I better tell people to use lisp!".)

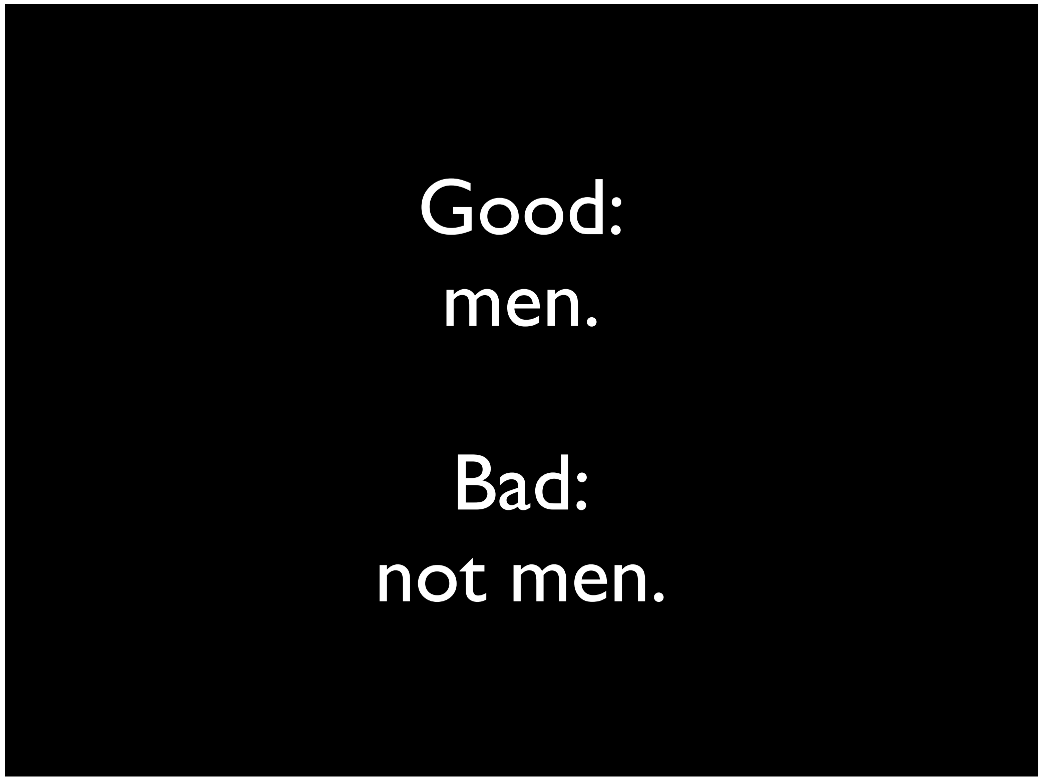


Good:
share my politics.

Bad:
don’t

Another famous loudmouth on the internet said "Good programmers are liberal, bad ones are conservative", or the other way around. He decided that the best way to talk about code style was to use terms even less people understand, and transformed tabs verses spaces into a political flame war.

Why do we do this? Everyone likes a simple answer to a hard question, and when it's emotionally charged too, you get far more attention.



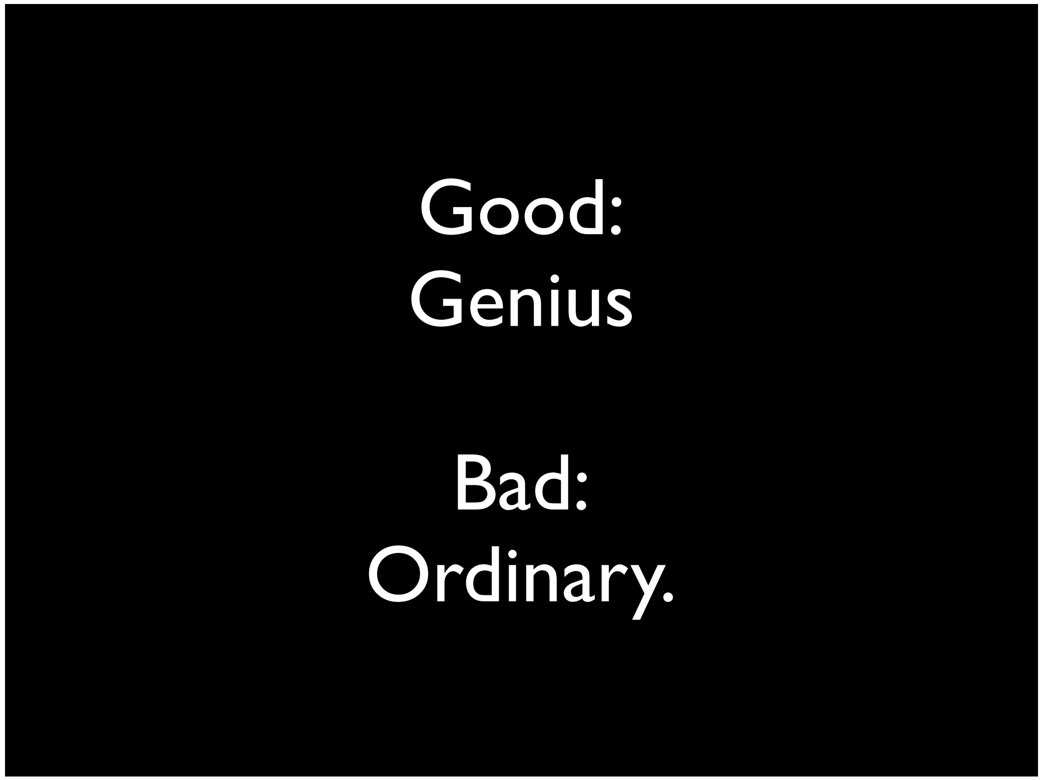
Good:
men.

Bad:
not men.

More recently, another loud mouth, Dave Winer, asked "Why aren't there more women in computing", and instead of doing some basic research, he went for awful psychology. He suggested that because cavemen had gender divisions, thus programming did.

The sheer ignorance of history in his statement was astonishing. The first computers were women, rooms of them doing calculations by hand. The first programmers were women too, because it was seen as a low status job. It also turns out the myths about cavemen "men did hunting, women looked after children" might be more due to our sexist cultural assumptions. Ruby itself owes a dbet of gratitude to Adele Goldberg and Barbra Lipkov, pioneers of earlier languages.

Although they opened with "I don't really know what I'm talking about", winer made no effort to learn, and is currently saying "I WAS JUST OPENING A DEBATE, YOU KNOW, ASKING IF WOMEN ARE PEOPLE, NO HARM DONE EH". If you find yourself saying "I'm just starting a debate", maybe you should probably do your research, and even more likely apologise.

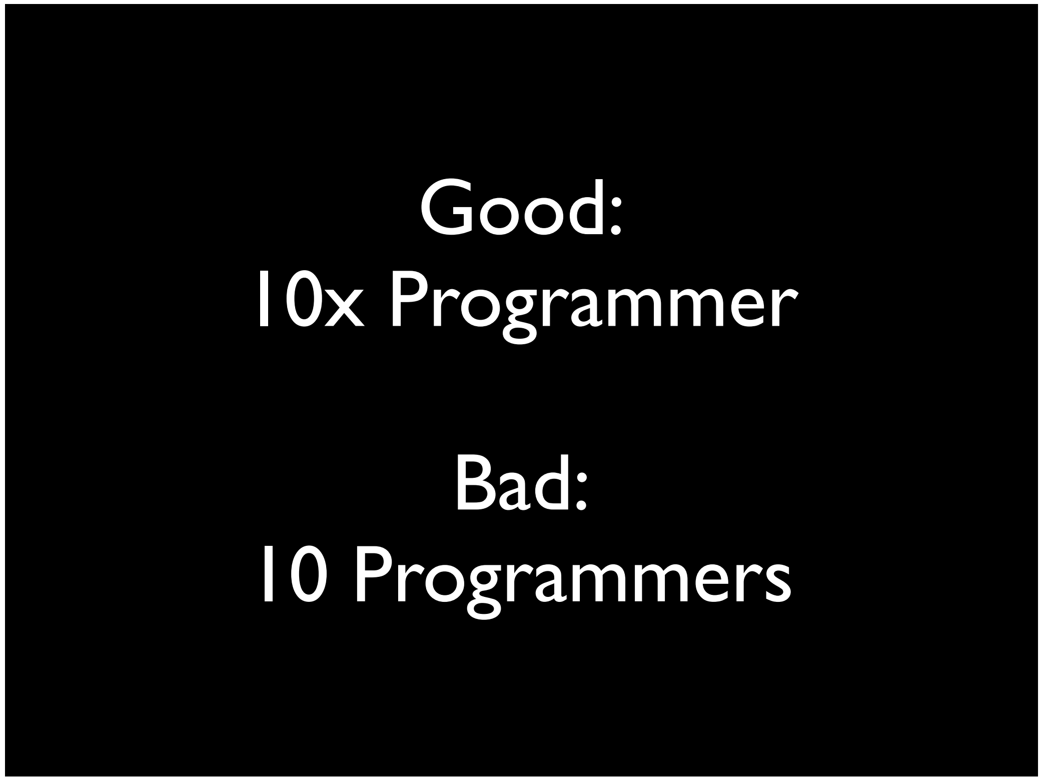

Good:
Genius

Bad:
Ordinary.

And you don't have to search far and wide to find actual research, for example "Is Math a Gift", dweck sets out to see if belief has an effect. Many teachers either say or assume that mathematics is a genetic ability, and so did the students.

What happened is that boys did better than girls. When they went to a school and changed the course work, to explain that it isn't something you're born with, it is something you work for. What happened is remarkable. The girls did just as well as the boys, and not only that, they both outperformed the original group.

Believing that ability is something you are born with, only encourages you not to try. For those stereotyped as being less capable than white men, it lets them run away from failure.



## Good:
## 10x Programmer

## Bad:
## 10 Programmers

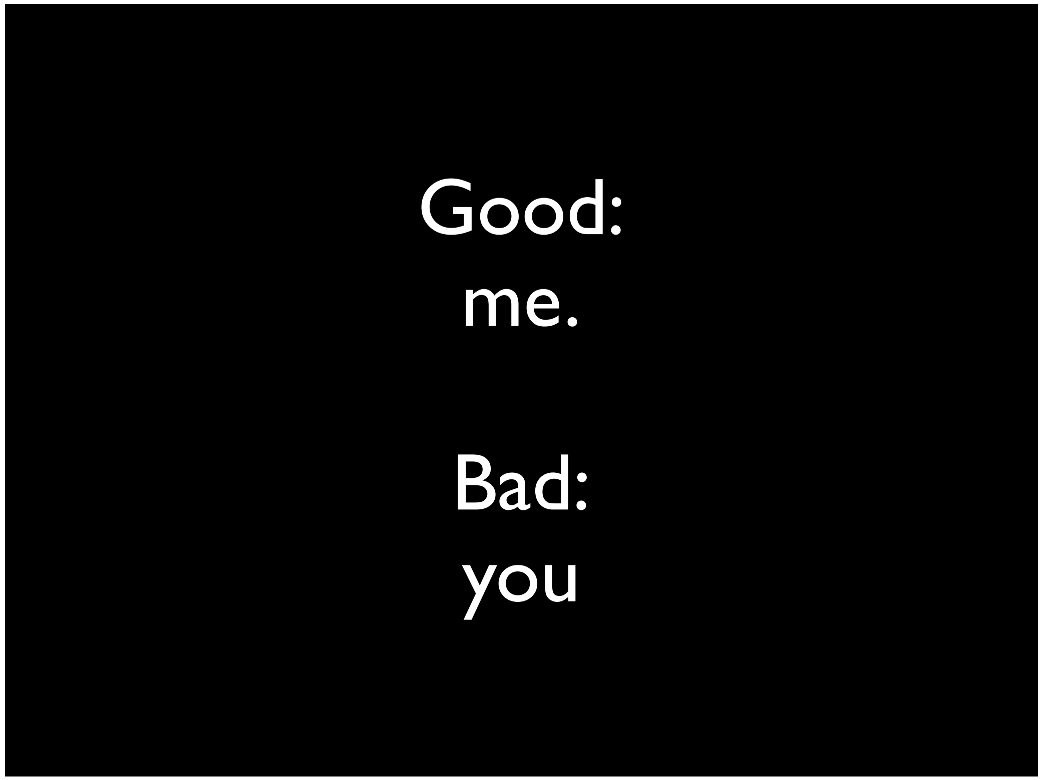
"Of course 10x programmers exist, I'm one of them". Sure enough there are some people like fabrice bellard, who seem to perform miracles, but the belief that some programmers are superhumans seems pervasive as ever.

Originating in a study on 12 people, for 30 minutes, working on punch cards versus working on interactive machines, it's been repeated over and over. Sometimes it takes the form that one programmer can outperform a team of 10 people, and turning into the myth of the lone genius.

Almost every line of code you use today has been written, edited and mantained by a group of people. Although a very small and focused team will have less communication overhead, they will never have the same breadth of experience as a larger, more diverse team.

Really what "10x programmer" means is this "We want to hire someone to do 80 hour weeks for almost no money". Penny Arcade recently advertised for a "Strong, Experienced Programmer" to do Four jobs, for under market rates for one.  They said "We don't want to give you more office, we want to make the office more fun to be in", which really means "Why should we pay you more? We'll never let you leave the office".

Really when a programmer talks about what makes a good programmer, and what makes a bad programmer, they really mean this:



Good:
me.

Bad:
you

"Repeat my mistakes and you will be as successful as I am", which may work if you're born into the same country, family, and wealth they are.

The reality is we don't know how to tell the difference between good and bad programmers, and you only have to look at how we interview to see it.

## Interviews

Some people advocate looking at github profiles (which tells you one thing: they can use github), or stack overflow (which tells you they're good at technical writing), or worse, brainteaser puzzles.

Can anyone guess what you should do if you get asked a puzzle question in an interview?

## LEAVE

That's right. Despite working in programming for almost a decade, I've yet had to move a mountain. The real problem with these is not that they are irrelivant, but they are wrong. A study called "The Wasson-Crick Selection Task" showed that problem solving ability is not an abstract ability, but often linked to the domain of the problem.

The selection task is an experiment to see if people are good at "contrapositives", or "We look for things that confirm our beliefs, but not for ones that show they are wrong"

They asked groups to solve a puzzle about playing cards, and another to solve the same problem, but this time about a social environment.  Most failed the playing cards, and most got the social problem right.

How you frame a problem impacts how you solve it. Asking people to solve puzzles outside of programming doesn't tell you if they'll write maintainable code.

If you get a programming puzzle, do ask "Have you ever used this in production", it'll embarass the interviewer, and it's fun. You might get told you're not a cultural fit for a company though.

We can't easily tell the difference between good and bad programmers. We also struggle with teaching people to be good programmers.

## Teaching our mistakes

If you ask any programmer how you should learn to program, I can tell you what they'll say. "Repeat my mistakes exactly.". Most teaching is concerned with nostalgia, learn what I did, how I did it, with no attempt at making it less awful for beginners.

Sometimes they will tell people to learn C, because it is hard, and character building. Even though they learned a bit of shell scripting before, they think it's best to start by teaching what is practical. Similarly, they'll tell you to learn Java or C#, because it will help you get a job.
Despite getting into programming as a hobby, programming to them is now just a career. With no room for experimentation, play, or fun. When you ask them how to learn, you should copy them exactly, but if you learn for the same curiosity they had, they get mad.

But to me, programming is more than just making a startup, or automating bureacracy, code is a medium for art, science, design too, and a way of not only communicating ideas, but exploring them.

## Mindstorms: Children, Computers, and Powerful Ideas

Other subjects suffer too: in mathematics, instead of being about problem solving, it's about memorising formulas and solutions other people have worked out. Learning isn't really fun when you don't get to be creative about it.

But maybe I'm also guilty of nostalgia too. I learned to program in LOGO, with turtle graphics. LOGO wasn't just about learning to code, but giving children a place to explore, and tools for thinking.

Back in the 60's one young coder was struggling with english, she couldn't work out why some words were nouns, and some were verbs. They set her a challenge "Write a setence generator", and so she sat down and hacked away at the problem.

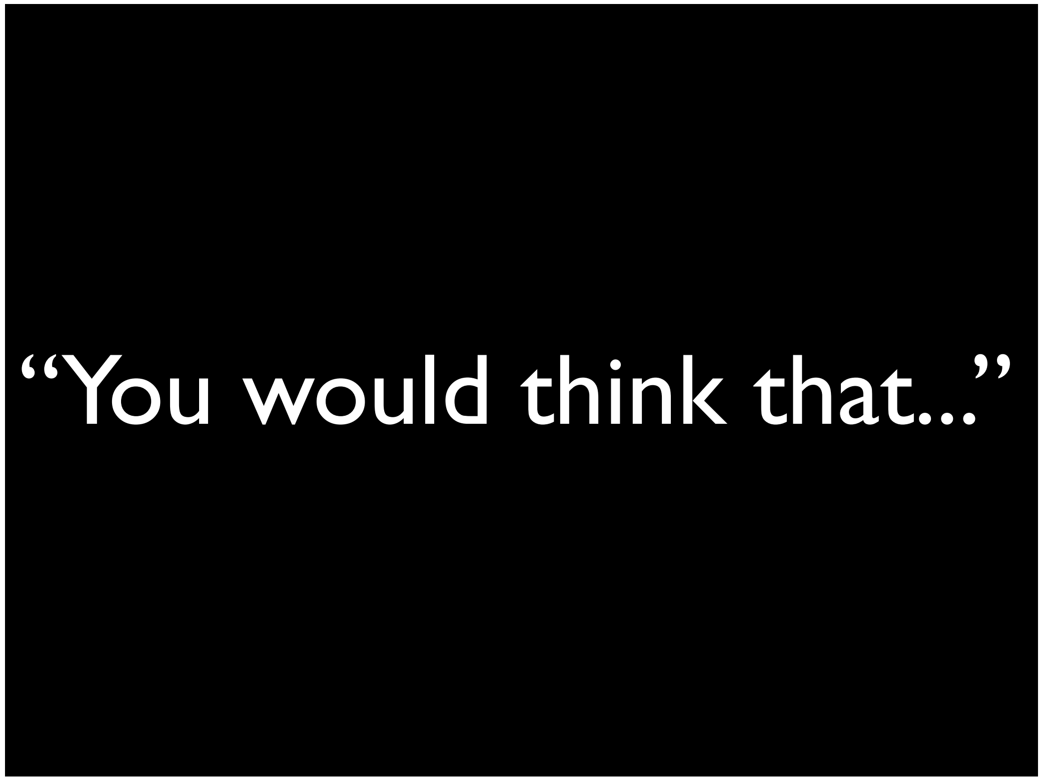
After making some typical programmer faces, she got her code to work and leaped around the classroom, yelling "I KNOW WHAT A VERB IS".

This is one of the core ideas around constructive learning: Don't give them answers, give them tools and let them think about problems. This is the main theme of Seymour Papert's book "Mindstorms: Children, Computers, and Powerful ideas", a wonderful manifesto for computer aided learning.

We shouldn't be teaching children to code, we should be teaching them to think, and software is one of many powerful tools to help them.

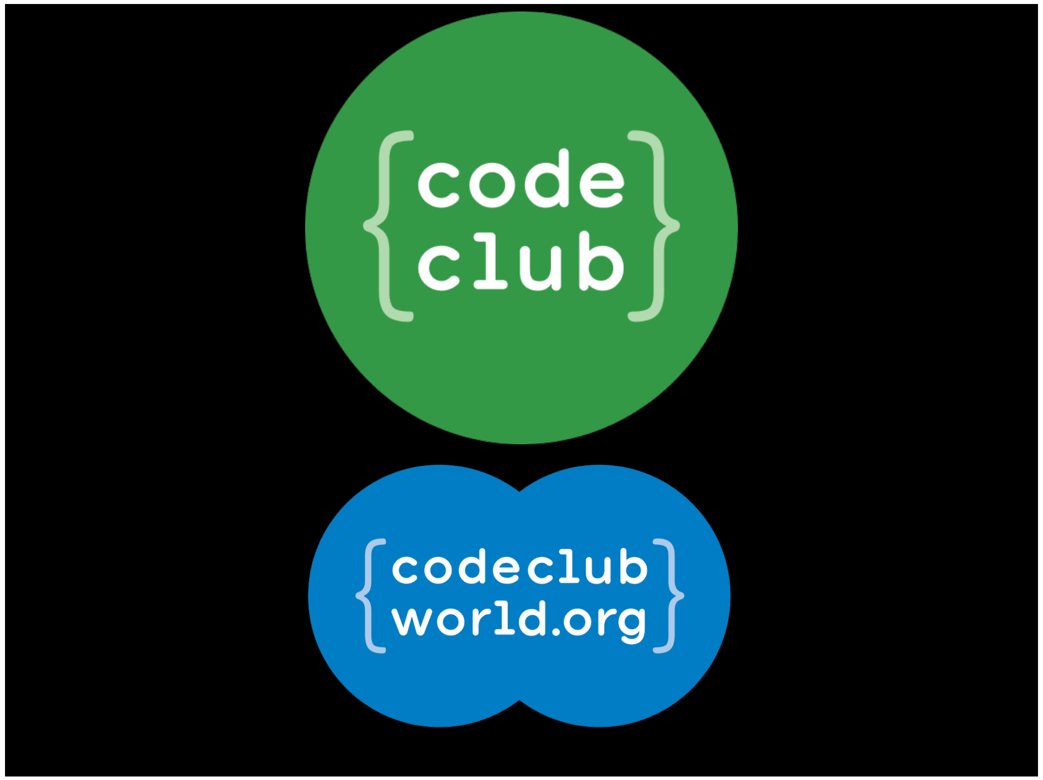
To me, Programming is the ultimate sandbox game, and the computer is a lever long enough to move the world.  I can't wait to put it in the hands of the next generation.

Despite promising you cynicism, I'm actually an optimist. You can tell because I say the following magic words.



## “You would think that...”

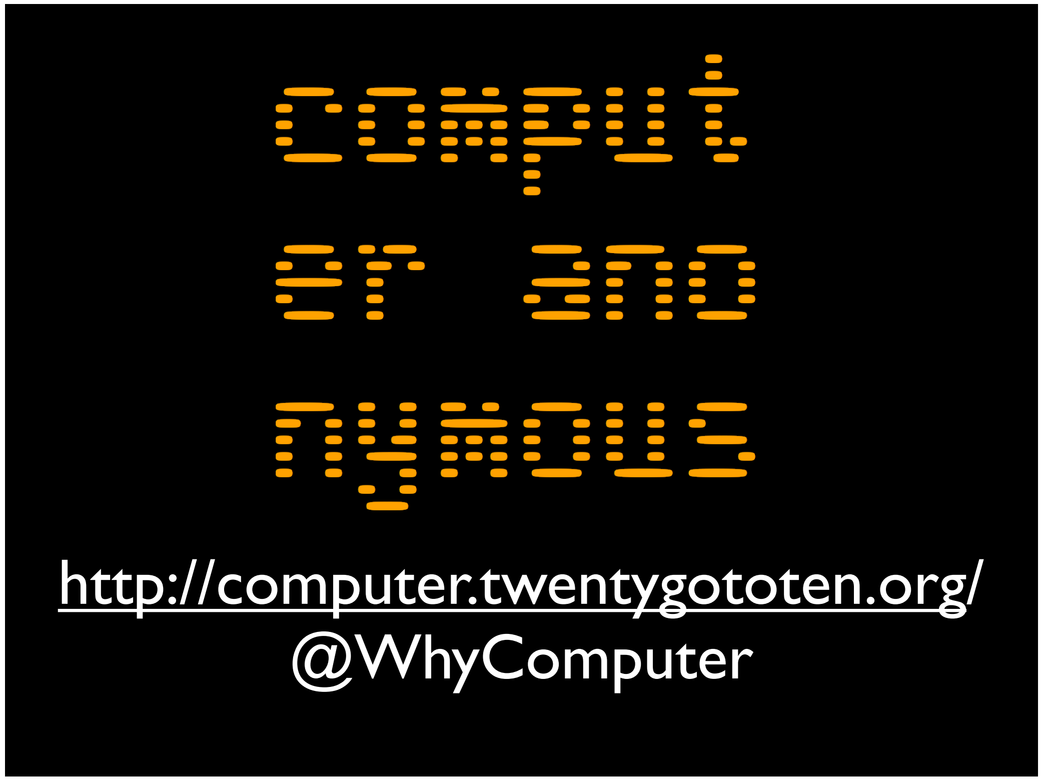
"You would think that", starts many sentences, and underlying it is a hope that technology could be better, implemented well, and mostly bug-free. As much as I'm bitter about how we code and how we teach, I know we can do better, and without some naievety and optimism, I wouldn't even begin to try changing the world.


code club

codeclub world.org

Which is my current job: Changing the world, one child at a time. We're trying to take Paperts Dream and make it happen. Clare and Linda founded code club to get volunteers into schools to help childen to learn about code through play. It's been eighteen months and they've gotten over one thousand four hundred clubs in the UK, and our coursework has been translated into many languages and used around the world.

We're still working on antartica. Linda will be talking about CodeClub tomorrow, so I don't want to spoil her talk, but I'd like to mention something else I've tried to change things.



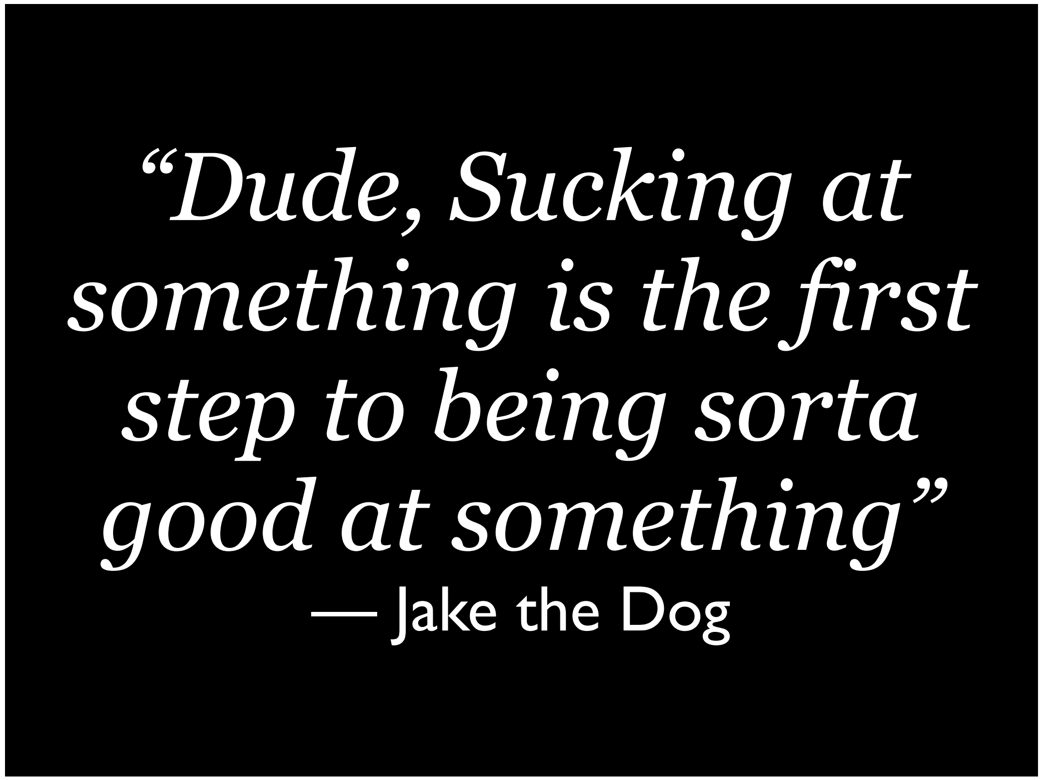
comput
er ano
nymous

http://computer.twentygototen.org/
@WhyComputer

Two months ago, a friend felt rejected from her local ruby group. Too much dudebros, and a some elitism and arrogance going unchecked.

So we said we'd go and make our own meetup, with a code of conduct, and a manifesto. We wanted to make an inclusive meeting, and especially for those who felt alienated by technology, or that they just weren't good enough to join in.

Within a week we had other meetings in germany and america. Before we'd even had our first meeting we'd had another 5. Today we have three different london meetups, 7 in north america, and a number around europe, uk and canada.



*“Dude, Sucking at something is the first step to being sorta good at something”*

— Jake the Dog

I know I'm a bad programmer, but that hasn't stopped me. I'm probably not the best at teaching kids, but I'm giving it a shot anyway.

My only hope is not making the world a better place, but being an example to others. I don't want people to repeat my mistakes, I want the next generation to make their own.

Thank you

programmingisterrible.com