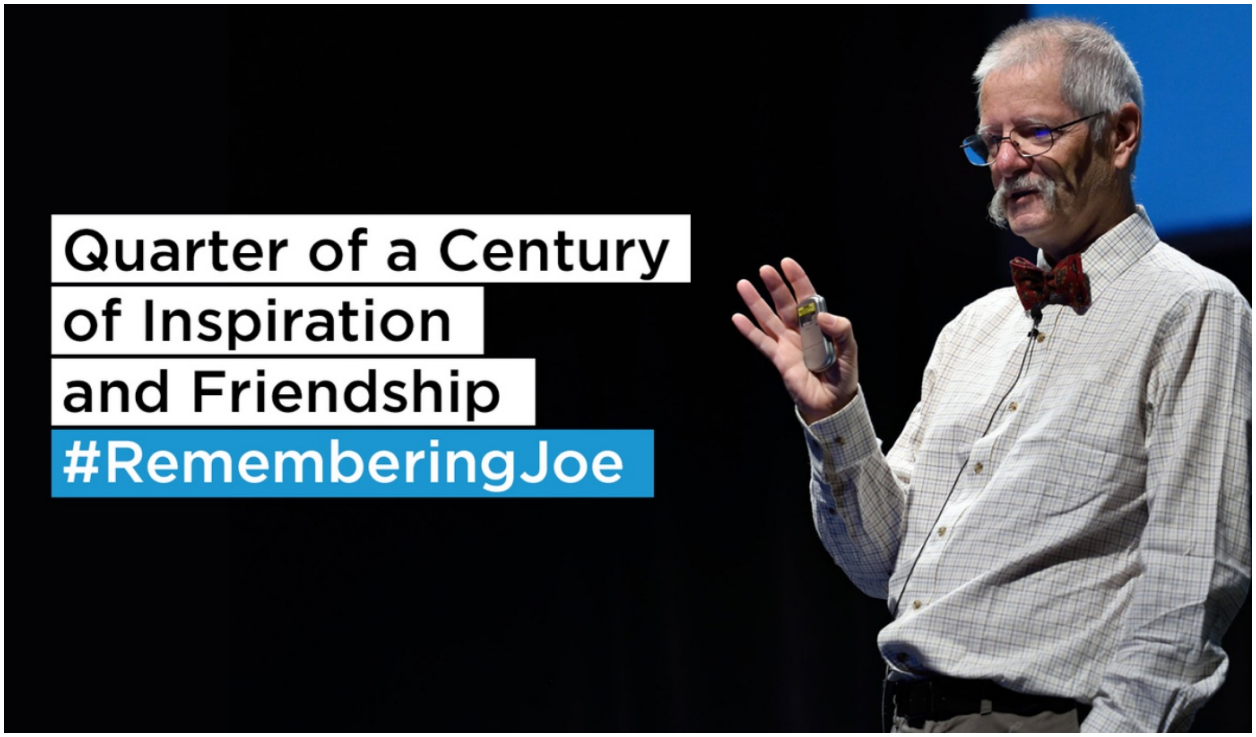


Remembering Joe, a Quarter of a Century of Inspiration and Friendship

2019-04-23 by Francesco Cesarini

I first came across the name Joe Armstrong in 1994, when I bought the first edition of Concurrent Programming in Erlang, a book he co-authored. Our first interaction happened in 1995, when I was looking for a company interested in sponsoring my Master's thesis. I dialled the number for Ellemtel Utvecklings AB, home of the Ericsson computer science laboratory, asking to be connected to Joe Armstrong. Getting a Hello, my unprepared opening line was Skall jag ta det på Engelska, or would you prefer if I took it in Swedish? A silent pause was followed by laughter, the same laughter many of us have come to associate with Joe.

Shouting at each other through the walls

Internships at the computer science lab were an immersion in enthusiasm and creativity. The lab had an academic, non conformist, almost anti-establishment, feel to it. Segregated to the corner of the building, pipe and cigarette smoke coming from some of the rooms. I am sure the cleaning staff were asked to "forget" the department existed, or maybe, they did not dare venture in there after hours. ISO 9000 reviews (and office moves) were a conspiracy to get Joe (and Robert) to clean their desks. But what mattered was not how neat your desk was, but the drive to innovate with an aim to further our

understanding in computer science. You did not use an existing web server, you wrote your own so as to understand how HTTP worked and suggest improvements and extensions. You did not take Ericsson's high security for granted, you found your way around firewalls and tried to outsmart the highly paid system administrators. When I got distributed Erlang working between my University and Ericsson UNIX accounts, I asked Joe if this was right. He walked in the hall and laughed loudly. It was his call for a show and tell, sharing knowledge with his curious colleagues who quickly make their way to his office. Sometimes, there was no need to make it there in person. Mike and Robert were always given the office next to his, so they could shout at each other through the walls.

Ahead of his time

Brainstorming sessions with Joe were the best. We once spent a good part of an afternoon finding a solution to a problem. I ended up working into the early hours of the morning on the implementation, and indeed, it worked. When Joe walked into the office the following day, he pops his head into my room and says You know the solution we came up with yesterday? Forget about it, it will not work. My jumping up and down saying it does work, it works, here it is, was fruitless. He replied, no, we were wrong, and walked off. He was often ahead of his time, in both thoughts and problems he was solving, and had probably discovered some edge and borderline cases to our solution which was too frivolous to share. This is where you learnt to filter his ideas, take those you understood or believed in and parked those you did not like or understand.

Solution to a problem

Sometimes, you would be walking outside his office, he would wave you in and share one of his daily epiphanies, hacks or articles and academic papers on a one-on-one basis. One of these hacks happened in conjunction with a major Erlang release in 1995. They had snuck in List Comprehensions and Funs in the language (product management had prioritized other features) and Joe showed me how they allowed you to implement clean and easy to maintain code, hiding recursive patterns and isolating side effects in a single location. Showing me the shortest implementation of QuickSort I had ever seen, he mentions you can solve the Eight Queens problem in six lines of code. I spent two nights trying to figure out a solution in my head, to no avail. Weary of a third sleepless night, I ask Joe for a solution only to be told: I have no idea, I never solved it myself. Look on the internet. WebCrawler and Lycos, whom had been around for a year, failed to provide a solution. Whilst I love Joe to bits, that particular day, I could have strangled him.

I think he was used to it (mentee students wanting to strangle him), as he claimed that we have always mistreated geniuses ahead of their time. Guttenberg was bankrupted, Galileo escaped the death penalty when put on trial, whilst Darwin's theories were mocked by

many. Fast forward to ten - fifteen years ago, governments and corporations alike were, instead of embracing peer to peer, persecuting those who built services on top of them. Ok, governments and corporations might not have strangled anyone, but they sure did sue them or try to put them in jail. Joe was disgusted by their treatment (he was the one who showed me how bittorrent worked), but he did smile when I told him one of the Pirate Bay founders claimed in an interview he was going to use his time in jail to learn Erlang. (I've looked everywhere for the article where I read about it, if anyone has a link, please send it to me francesco.cesarini@erlang-solutions.com)

Leaving Ericsson

In December 1998, a few days after Erlang was released as open source, Joe and some of his Computer Science Lab colleagues left to found Bluetail, later acquired by Alteon Websystems, who in turn got acquired by Nortel. We were both very busy building products these years, and met mainly at conferences and workshops. I do recall being told of the buzz at Nortel when Erlang Solutions launched its first website, and when the dotcom bubble burst, receiving a call from Pekka, a colleague of his, saying he, Joe, Robert, Jane and many others in the team had just been fired! It did not take long for Nortel, in 2003, to start advertising for Erlang developers with ten years of experience, not realising they had recently let go seven of the ten people who at the time fit the bill. Nortel would not have hired me, at the time I only had 8 years of Erlang. You've got to love random acts of management.

Seeing his redundancy as an opportunity, he decided to make a stint in academia, working on his PhD through SICS, the Swedish Institute of Computer Science. Making reliable distributed systems in the presence of software errors is a must read for anyone trying to understand the rationale behind systems you write once and run forever. Folklore (Joe after a few beers) has it that after a couple of years at SICS, he walks into his supervisor's office and submits the full thesis. His supervisor looks up surprised and says, This is not how it is supposed to work. I should be giving you feedback as you write it. Oh, goes Joe, Let me know what you think. After SICS, Joe returns to Ericsson, completing the full circle. In 2014, becoming an Adjunct Professor at KTH, continuing to inspire students through his magical ability to pique their curiosity. Just like he inspired me back in 1995.

Quest to learn

Joe knew that programming languages were not about popularity or beauty contests. They were all about solving problems and progressing the industry. He was just as excited about Rich Hickey, Don Syme or Simon Peyton-Jones's success stories, and wished Haskell, F# and Clojure to do as well. More recently, he got all excited about Sylvan Clebsch's Ponylang. When we had dinner with the Go team, he enthusiastically

explained Erlang's concurrency error handling mechanisms to Ken Thompson. He was encouraging Ken to integrate similar semantics in Go. Erlang will not be around forever, he once told me in the 90s. Something better will come along. I don't think we realised back then, that whatever will come along, is going to be heavily influenced by his work!

There were obviously languages he did not like, C++, Java and JavaScript (read Node.js), backed by well motivated reasons. But despite not liking them, he wanted to meet the inventors behind these languages in his quest to learn more, understand their motivation and share his ideas. He asked me to try and get Brendan Eich to speak at the Erlang Factory in San Francisco and was hoping to meet Bjarne Stroustrup at Code Mesh in London. I once had to drag him out of Google in Mountain View when James Gosling was in his office and we happened to walk outside. Joe asked me if I thought it would be impolite for him to just go in and introduce himself, to which I suggested (for the sake of our host) that it might be better not to, and instead, get a proper introduction.

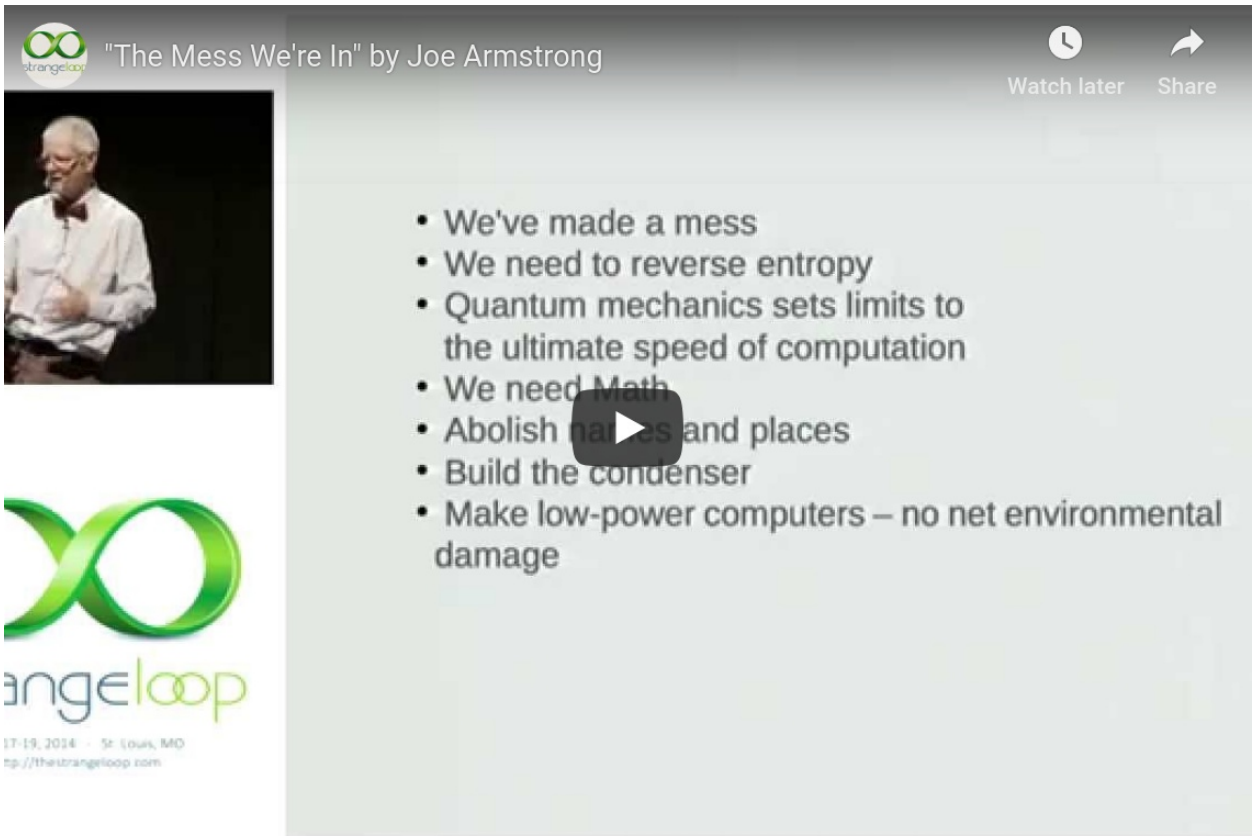
The Trio, or even Quartet!

We are talking about Joe, but let's not forget that for a good part of his career, he was part of a team together with Robert and Mike. They were led by a patient Bjarne, who gave them free rein in solving telecom-related and being interested in what, and not how. It was the space they needed to innovate. I am not sure any of them on their own have been able to create Erlang, but together, they were able to leverage each other's strengths and succeeded at creating both Erlang and OTP. I tried multiple times to get them on stage together to show the dynamics of the trio, but it was always scripted (even Joe's parts!). I

succeeded the third time, at the 2015 San Francisco Erlang Factory Keynote, From WhatsApp to Outer Space, where the last 15 minutes of the talk, they go off script and start asking each other questions, to the roaring laughter of the audience. It shows Joe to be the inventor, Robert to be the one interested in aesthetics and Mike the finisher and end user with industry experience. It was wonderful to see their team dynamics, friendship and bonds still in place and going strong, twenty years on.

Beyond Erlang

In more recent years, Joe started talking about topics beyond Erlang. His Strangeloop keynote, The Mess We're In, focused on one of his pet peeves in the software industry. As a result of computers becoming faster, software seems to become more complex, and hence, slower. He refers to the laws of physics, something the software industry has tried to defy for decades. Joe (a physicist by training) applied the laws of physics to computer science and distributed systems. Synchronously passing messages, shared memory or attempting to share data faster than the speed of light. If you hear a programmer say you can't do this, it defies the laws of physics, you now know where their quote comes from.



The Mess We're In

One of the many projects he was planning on doing after retirement was to interview his heroes, and if he got enough interviews, publish the results in a book. He suggested interviewing Alan Kay on stage at Code Mesh in London in 2016. I have never seen a conference audience so mesmerised. A similar reaction happened in the Let's #TalkConcurrency panel discussion in November 2018, where we were able to get Sir Tony Hoare, Carl Hewitt and Joe Armstrong to discuss the past, present and future of concurrency models. Joe was originally supposed to run the interview, but as many of us felt he had just as much to say as Tony and Carl, we got him on the panel instead. I am glad we did, as we covered three different, but overlapping approaches to concurrency, each created to solve a different problem. Travelling to Cambridge with Joe for the recording, it was obvious he was not well. His lungs were at 60% capacity, and he often ran out of breath. The pulmonary fibrosis was evident. But we all hoped they would be able to keep it under control.

Lung Research

On Saturday April 20th, 2019, I get the dreaded WhatsApp message that Joe had just passed away. Just the day before, the news was more positive, they had narrowed down the diagnosis and had adapted the treatment accordingly. Unfortunately, it was too late. Joe leaves behind his wife, Helen, his children Claire and Thomas and two cats, Zorro

and Daisy, who figure in various programs. Joe had named a previous generation of cats Wittgenstein and Schopenhauer but reality, in the form of Helen, renamed them; they became known as Tubby and Tiger. He also leaves behind many friends, colleagues, students and followers who will continue his work, spreading his ideas and ensuring they evolve and keep on getting embedded in mainstream programming practices.

We are all glad Joe got to see how his work has impacted the world around him, and how Erlang Style Concurrency and OTP are being adopted in the realm of distributed, fault tolerant systems which have to scale on multi-core architectures. Basically, the immediate future.

Thank you Joe for being yourself. Thank you Helen for supporting him in doing what he loved the most. And thank you Claire and Thomas for helping bring up an older brother called Erlang. As the old saying goes, no one truly understands concurrency until they have their second child (or cat).

Helen and Claire kindly requested donations to be made to lung research, with Claire starting a fundraiser on Facebook:

https://www.facebook.com/donate/312270252802913/312270286136243?sfns=mo

In the US there's the American Lung Association:

https://www.lung.org/get-involved/ways-to-give/

In the UK British Lung Foundation:

https://www.blf.org.uk/donate

In Sweden Hjärt-lungfonden:

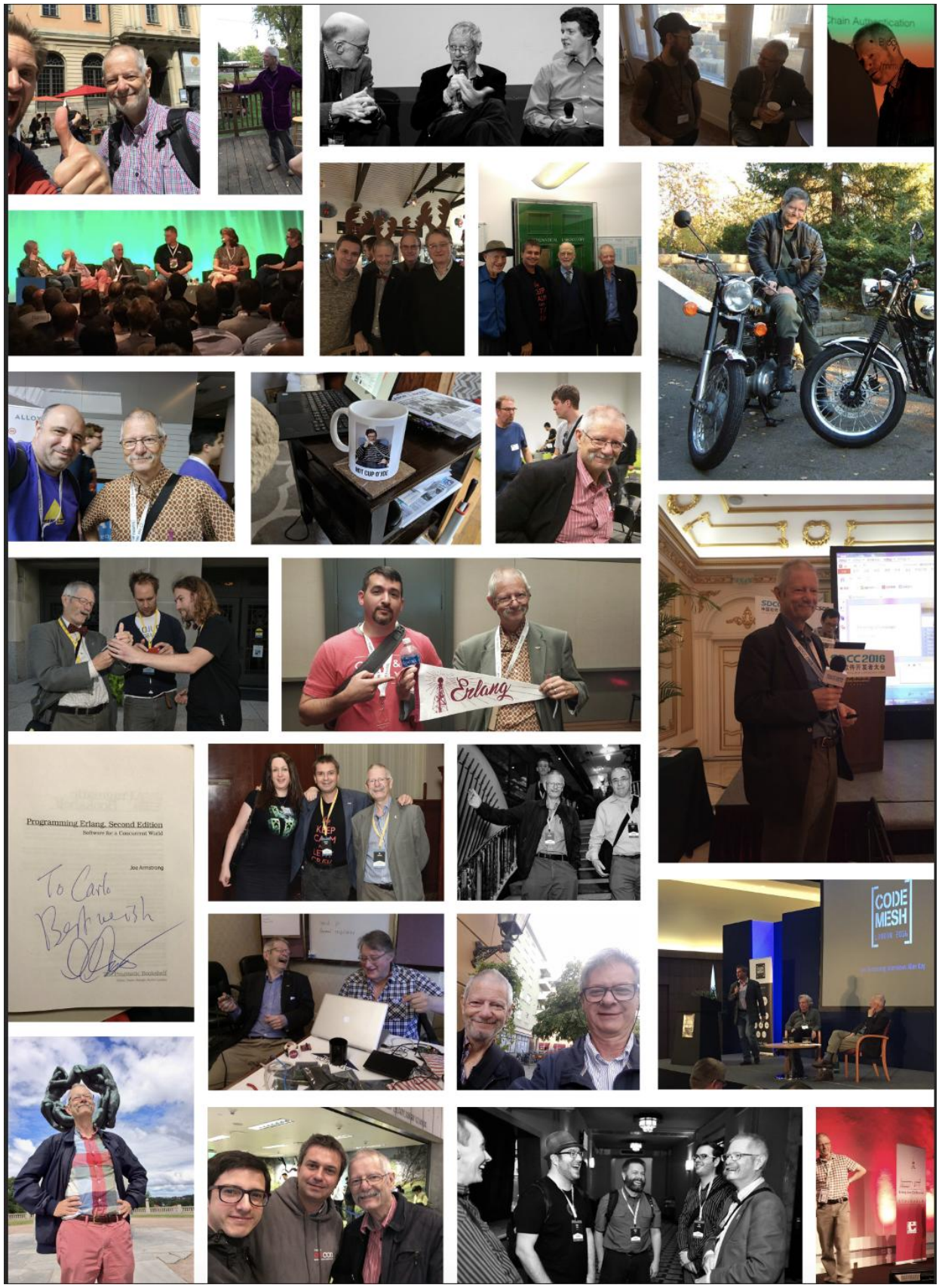
https://www.hjart-lungfonden.se/Stod-oss/

Remembering Joe <3

#RememberingJoe with credits to the wonderful community - many thanks for sharing <3

Many thanks and photo credits to: @samaaron; @puredanger; @rolodato; @bentanweihao; @MachinesAreUs; @bltroutwine; @danielozcps; @christian_fei; @bryan_hunter; @EuenLopez; @cacorriere; @t_sloughter; @acscherp; @strangeloop_stl; @janjiss; @zhanghu; @MarioAquino; @MakisOtman; @perlhack;

@colrack; @aodanne; @gsarwate; @scrogson; @gar1t; @RogerSuffling; @gturitto; @lelff Special thanks to @michaelbridge for the feature photograph.



Goodbye Joe

2019-04-20 by Fred Hebert

Joe Armstrong is mainly known as the father of Erlang, and the Erlang family has always been relatively small and closely knit. Anyone whose first Erlang conference (usually Erlang Factory, Erlang User Conference, or CodeBEAM) had Joe in the attendance would have a similar reaction. There was a feeling of awe about how accessible the community was. Here you were, and big names like Joe and Robert—who everyone knew by their first names—were right around the same room, friendly, and willing to talk to anybody. You'd feel like you were welcome no matter who you were.

Today, we've learned of Joe's passing away. I wasn't a super close friend of Joe, but I have known him and talked with him at various conferences over the last ten years or so. He's unsurprisingly been a huge influence in my career, and so I thought I should write this little post about him and his impact. My words can't give justice to the man he was, but I felt I needed to write this up.

Around 10 years ago, I used to sling PHP for a dating site, fresh out of a multimedia program in college. I had done a bit of C, Python, Scheme, Javascript, and a few other languages for fun by then. I had been told to figure out how to renew our chat system, and since back then Facebook had written their first one in Erlang (before rewriting it in C++, then buying WhatsApp), my boss threw a copy of Joe Armstrong's Programming Erlang (the first edition) on my desk.

This started my whole story with Erlang, which still goes on today. Joe's book was approachable, the same way he was. He could explain like no other what the principles of Erlang are, what led to its design, and the core approach to fault tolerance that it embodies. It's one of the few language-specific books that is not content with getting you to write code, but lets you understand why you should write it that way. The language didn't just have features because they were cool, it had features because they were needed for fault tolerance. What Joe told you applied anywhere else.

I would end up reading almost everything I could that Joe wrote. Among them:

- Programming Erlang
- His Thesis, which really outlines all the principles in details, including more extensions to the language. It's a great unifying test that anyone aiming for fault tolerance should read.
- His latest blog
- His older blog

Joe and Fred at Chicago Erlang 2014.

- The old paper that showed the first version of Erlang implemented in Prolog
- Various posts on the Erlang mailing lists

One of the amazing things Joe mentioned in his texts that was out of the ordinary compared to everything I had read before is that developers would make mistakes and we could not prevent them all. Instead, we had to be able to cope with them. He did not just tell you about a language, he launched you on a trail that taught you how to write entire systems. But Joe's advice and writing went further than this. He was always a huge fan of systematic simplicity, self-contained straightforwardness, and stories from the older times of computing that gave a fresh perspective on everything.

Rather than thinking software is never finished, he wanted software that was so simple it could actually be completed. The strength of your program should be from how extensible and composable its usage would be, not your ability to never be done writing it. On top of that, he wanted code to keep working forever.

One of his blog posts, My favorite Erlang Program, perfectly embodies this. Joe was the kind of person who would take a simple idea, and push it as far as possible, uncovering some great insights on the way. Then he'd share it with everyone.

Joe was a man who would get a thousand ideas a day. Maybe a few of them would be good or practical. For each of these, you'd probably need a whole team to take the underlying principles and make them usable for the rest of the world. But Joe would go on undeterred, always excited, and happy as ever exploring ideas and new concepts all the time. Some would be bad, some would be great. You never really knew what you'd end up with, but you know you'd have passion and great discussions. Should PDFs be both the documentation for a piece of code and the compiled artifact? What if no piece of code ever bore a name and instead you'd just use hashes for all of them and had globally unique everything? It would be good if everyone aimed for inter-program composability rather than whatever it is they are doing right now!

If you were talking with people at a conference, all standing in a circle, and you suddenly got Joe's attention, he'd be inching closer and closer to you, entirely focused, eventually ending up face-to-face with you, the circle crushed by his interest. You'd be able to have his undivided attention for a while, between huge bursts of laughter. He carried this everywhere. You'd invite him to talk about something, and if he had discovered something he felt was more interesting than what he had initially planned, he'd talk about that instead, would go over time if he had to, and everyone was fine with it because, well, that's just Joe. And it was always fun stuff, too.

On a more personal level, Joe was super enthusiastic about some of my first contributions to the language. He wrote the foreword to Learn You Some Erlang, and was always up for a chat. He's had a major influence on my career, and I owe him much. He has unknowingly sent me down a path I am still gladly walking.

Outside of software, Joe really enjoyed music, typesetting, and creative writing, which he would gladly tell you about. He's not known for these to nearly the same extent, but he did it all with the same passion and enjoyment.

He was part of the Erlang landscape, always interested in what people had to say. His passion and enjoyment about the craft, even in his 60s, was still high up at levels I don't even know I ever had or will ever have, and I have to say I am envious of him for that. I don't know what it will be like to have this community without him around. He was humble. He was approachable. He was excited. He was creative. His legacy is not just in code, but in the communities in which he instantly became a central part. He will be missed.

Hello Mike,
Hello Robert,
Goodbye Joe.

Joe the Office Mate

2019-04-21 by Luke Gorrie

I was sad to hear that Joe Armstrong passed away this week. He was a kind, generous, witty, brilliant fellow. He was also a hero and a mentor to me personally. I had the privilege to be hired by Joe to work with him at Bluetail and I'd like to share a few recollections of him as a colleague and office mate.

Joe loved to talk. His thought process involved lots of animated discussion. He spent a lot of his work day bouncing up and down in his chair, waving his arms, and sketching on whiteboards. Each time he came upon an interesting idea he would shout loudly ("Ooooohhhhhhh!") until somebody would come and hear about it. He was fantastically accessible and he livened up the whole office.

Joe wrote amazingly simple programs and he did so in a peculiar way. First he wrote down the program any old way just to get it out of his head. Then once it worked he would then immediately create a new directory program2 and write it again. He would repeat this process five or six times (program5, program6, ...) and each time he would understand the problem a little better and sense which parts of the program were essential enough to re-type. He thought this was the most natural thing in the world: of course you throw away the first few implementations, you didn't understand the problem when you wrote those!

Joe's home directory was a treasure trove of new and old ideas. He had his latest experiments and also "classics" like decades old verisons of Erlang that were still hosted on Prolog. This was all openly shared over NFS and exciting to explore. (I hope that Joe's home directory ends up somewhere like the Computer History Museum.)

Joe could see the essense of problems. His off-the-cuff solutions often sounded hopelessly naieve and oversimplified. Come on Joe, that's too silly, there's much more to it than that. Often after a few weeks of hard work I would come up with an elegant solution of my own -- and on the way to tell him about it I would recognize it as exactly the same idea he gave me in the beginning.

Joe talked about Richard O'Keefe and Niklaus Wirth as the best programmers in the world. He would often quote engineering trade-offs from Project Oberon: yes, overlapping windows are better than tiled, but not better enough to justify the implementation complexity.

Joe would get wildly excited by one "big idea" for weeks at a time. This could be a new idea of his own or a "well known" idea of somebody else's: the Rsync algorithm; public

key cryptography; diff algorithms; parsing algorithms; etc. He would take an idea off the shelf, think (and talk!) about it very intensely for a while, and then put it back for a while and dive into the next topic that felt ripe.

I am happy that Joe lived to see his life's work so well appreciated. I think that is a rare privilege even amongst very brilliant people. I always smiled when I saw him keynoting conferences and sharing his ideas with a receptive audience. Great work, Joe. Rest in peace.

{'EXIT', joe, goodbye}

2019-04-21 by Jesper Andersen

> *Jesper, I have this idea in which we'll connect all of the worlds Erlang systems to each other, imagine if every process could talk to every other process, world-wide!*
>
> —Joe Armstrong

Joe was never short on ideas when you spoke to him. At conferences, he would engage people with his newest idea, or he would find people across the room and connect them. This often resulted in new acquaintances, interesting conversations, and new insights. Joe acted as the fountain from which insights sprang. He always had a new project going, and was keen to tell about it.

Joe would speak to everyone. It didn't matter if they were new to the world of Erlang, or computers, or if you had 20 years of experience. He would quickly find your level, and then discuss at that level.

In my mind, three rules embodied the ideas of Joe:

- They were always practical and wanted to solve a grand problem with computers. Most often, the limitations of physics played a role.
- They always felt a little bit crazy, mainly due to the novelty of the idea.
- They were going to restructure your brain, and how you thought about stuff.

When I was writing a BitTorrent client for Erlang, Joe was quick to comment on the code "This part uses defensive code style, you can probably just let it crash." It took some years for that lesson to fully sink in, as it did with most of Joes stuff. He would gently nudge you in a direction, and by following his trajectory, you landed perfectly.

Most people reading this will know Joe for his work on Erlang. But he had lots of ideas, and not all of them pertained directly to Erlang itself, though it was often the vehicle. He had a keen interest in music and did collaboration on the Sonic Pi with Sam Aaron. Up until recently, Joe was working on improving wiki tooling. In particular, he'd realized how he needed a quine (a self-reproducing program)—the wiki should contain its own source code so it could reproduce itself. This would ensure the longevity of the wiki. This work was in April 2019, so he was working on stuff up until his untimely death.

Joe liked to find minimal solutions to problems. He would strip layer upon layer off a

problem until he had the core. He created a stack of universal binary formats, UBF. The ingenious part of this was that to transfer a term, one would transfer a program which when executed on a small stack-based virtual machine would yield said term. And the commands were chosen from the printable ASCII alphabet.

The higher layers in the stack was the basis for a lot of discussions I had with Joe. Both of us agreed that what happens “inside” an Erlang process wasn’t that interesting in the grand scale of things. It was the communication which was important, and it should have a contract system. Joe had certainly invented most of this before I even started looking at Erlang as a language. And I still—to this day—believe that this is future of protocol communication.

To see how visionary Joe was, his idea was to use such a contract system to get any Erlang process on any node to talk to another Erlang process, somewhere out in the universe. Essentially, this gives you “Serverless” operation, so he was a couple of years ahead of the pack on that.

Joe also wanted to mix this with a content addressable storage spanning the internet. He would speak fondly of IPFS which to a large extent was this vision. The main idea is to generate the reference key for data from the data itself, usually a cryptographic hash over the data. This in turn provides integrity: if the data changes, so does its key. Now, if you refer to data by its key, then you can verify you got the right data. Joe wanted to use this as a basis for software: “I can send you the hash, and you can fetch the library if you don’t have it.” he told me. He also wanted to use this idea to protect old software so “It could still run after many years.”--another reason Joe preferred minimalistic approaches to software. “You see, a package version is the hash of its source code, not a version number, that would be silly.”

He was adamant on making computers useful.

The greatest brain restructure, however, was always in the idea that your abstract logic of a program has to execute in the physical environment. This meant coping with failure when it happened as the only way to build robust software. Programming without this profound insight—software cannot proactively remove all error—is as silly as it is dangerous in hindsight.

Once you embrace fault tolerance, then you have the feeling of a burden released, and programs gets far easier to write. Of course, the more seasoned programmer knows when you can be proactive and when you have to be reactive. But if there one thought which has shaped the way I think about programming the most, it is this.

The second greatest brain restructure was Joe's dismissal of performance in software. In a post to the Erlang mailing list, Joe would come up with the following table:

```
This is what we do: (We = ///)

    1) hack it in erlang
    2) fast enough - ship it
    3) tweak the erlang
    4) fast enough? - ship it
    5) hack it in C?
    6) fast enough - ship it
    7) tweak the C
    8) fast enough? - ship it
    9) make an FPGA
    10) fast enough - ship it
    11) make an ASIC
    12) fast enough - ship it

As you go down this list things cost more and more - step
11 costs 1M$/try - to pass step 9 you need to have a high
volume product (10's to hundreds of thousands of units)
```

The core idea is that if you want to go really fast, you need to implement hardware specific to the problem, and you can only hope to tweak the software so far in performance. And he would make a point about setting up a specified target before starting the tweakery, so you'd know when to stop. Over the years, I've veered in the same general direction as Joe.

One thing I liked about Joe was his fearless approach to computing and to life. He'd never bow to authority. He'd would never be afraid to provoke if it was necessary. He never ever stopped R&D. He would tinker with things until he understood them, then find something new to look at. But only after he told you about his findings. Sharing was his modus operandi.

At conferences—in which Joe was an attendant—there would be this hallway track going on where people would talk with Joe. A slot would be skipped here and there. But some interesting conversation would be had, and people would convene around him in a circle that grew ever larger. I've watched more than once when a "innocent bystander" was drawn into his web of stories, findings, and insights.

Being a speaker with Joe in the audience was always a blast. Once he sensed he was allowed to interact, you could be sure he would “heckle” your talk in the most awesome way. It only took a smirk and you saying {hello, joe} and that would be his cue. We should have mic’ed him up at times.

When he did speak up, he would often frame a question such that it put a speaker in a position where they could really show their work. Never have I heard a bad question by Joe.

This was the Joe I knew. I think we owe him a lot with regard to pushing software and computer science ahead. The best we can do is to make sure his grand ideas are not lost upon us, and that they are implemented in the future to come.

And then perhaps, we will reach a point where Joe’s prediction from this Month will ring true:

One day computers might become useful.

—Joe Armstrong

Thanks, Joe

2019-04-23 by Jonas Bonér

A couple of days ago I was struck by the very sad and unexpected news of Joe Armstrong's early departure. Joe meant a lot to me personally and the Akka community at large, he has been a great inspiration and influence over the years, and I wanted to take some time to express my gratitude to him, both personally and on behalf of the Akka community.

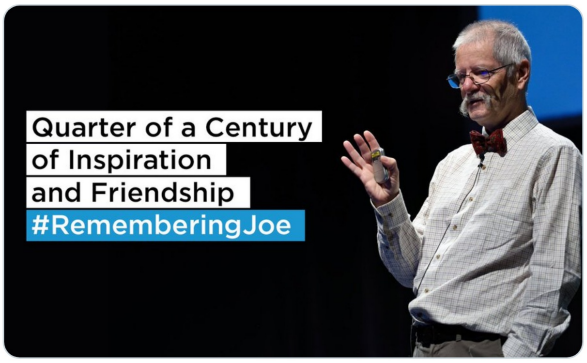
Joe was a rare visionary, and his groundbreaking work on Erlang and general distributed systems design can be traced back to the early '80s. He was way ahead of his time and it is not until the past 5-10 years that our industry at large has had to catch up with the architectural styles Joe has been promoting for decades—without "Let it Crash!", location transparency, async messaging, immutability, declarative programming, and a vision of systems with millions of loosely coupled isolated processes/actors working together as a single cohesive system, there would be no Akka, Reactive, or modern distributed systems.

Apart from being technically brilliant, Joe was very generous and approachable, always fun to hang out with, highly energetic and curious (of people as much as tech), always smiling, with a joke, unpredictable wit, or rant up his sleeve, and always eager for a fun debate, whatever the topic.

Thanks for everything, Joe. We will miss you a lot.

Erlang Solutions
@ErlangSolutions

To our dearest Friend, Collaborator and Mentor Joe - we will miss you dearly ❤️
All of us at Erlang Solutions send our love to Joe's Family, Friends and Everyone saddened by the news from last Saturday.
#RememberingJoe by @FrancescoC erlang-solutions.com/blog/rememberi...



Bryan Hunter
@bryan_hunter

A BEAMing @joeerl

#rememberingjoe

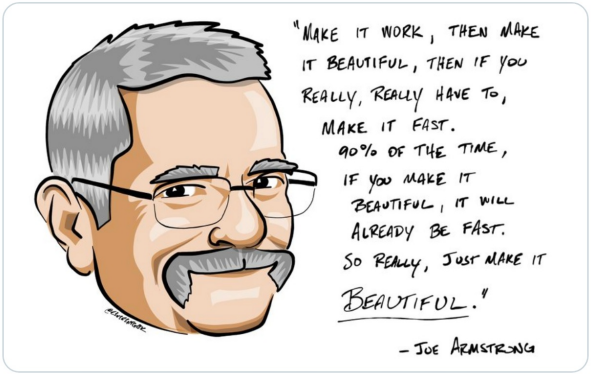
Peer Stritzinger
@peerstr

“Have fun - if it’s not fun it is pointless” /Joe. Learning Erlang was the best thing that happened in my programers life. Thank you for that and many inspiring chats.
#rememberingjoe

David Neal 🥓🥑
@reverentgeek

The software industry has lost one of its greatest champions. Rest In Peace, @joeerl.

#rememberingjoe #joearmstrong #erlang #programming #distributedcomputing



Ulf Wiger
@uwiger

Dear friend and mentor @joeerl shared my love of music. He and Helen visited one of the opera recitals we used to give at our home. Katarina recalls how they arrived on motorcycles, lighting up the place with their smiles.

#rememberingjoe

Katarina Pilotti/Ulf Wiger: Adriana Lecouvreur (Cile...
From Adriana Lecouvreur (Cilea) Glanshammars kyrka, 19 July 2014 Katarina Pilotti, soprano Ulf ...
youtube.com

Torbjörn Törnkvist
@kruskakli

At CSLab we suddenly hear Joe bursting into happy laughter. Turns out he had implemented the NNTP protocol and when listing the News group names, a large number of AT&T groups shows up with peculiar names, which when listed becomes a big ASCII art saying: AT&T
#rememberingjoe

UniKentComp
@UniKentComp

Replying to @ErlangSolutions @FrancescoC and @joeerl
So sorry so hear the news of Joe's passing. We are grateful for the time he gave us to film the Erlang MOOC #rememberingjoe

Brujo Benavides
@elbrujohalcon

#rememberingjoe: 2012. @chaddepue convinced me to give my first talk ever. I was SO SCARED but a very funny "old man" kept nodding in the audience (reassuring me that what I was saying made sense). In the end, he said "great talk!". Later on, I learned it was @joeerl :')

Tristan Sloughter 🌹
@t_sloughter

I learned the news while finishing my morning hot cup o'Joe :(#rememberingjoe

Francesco Cesarini
@FrancescoC

I am glad to see everyone sharing their @joeerl moments, making it not a day of sorrow, but one of celebration. He would approve. #rememberingjoe

Euen
@EuenLopez

Can I join? he said, and I had the possibility to met one of the biggest icons of the technology and the language in which I work on daily basis. Thanks @joeerl for your kindness and your given inspiration. #rememberingjoe

Joe Armstrong and Euen

11:00 PM · Apr 20, 2019 · Twitter for iPhone

2 Retweets **24** Likes

Garrett Smith @gar1t · 1h
Replying to @EuenLopez and @joeerl
This was one of Joe's finest qualities - I don't think I've met anyone as approachable or as enthusiastic about deep engagement, like, immediately. And this for an Englishman acclimated to living in Sweden!

Rudolf Hersén
@rhersen

1991. Joe, Mike and Robert in the basement in Älvsjö doing brilliant stuff with a team of 20 or so people. On the floors above them, hundreds of C++ developers on an "OO is the future" project (cancelled 4 yrs later). #rememberingjoe

Brujo Benavides
@elbrujohalcon

#rememberingjoe: 2012. @chaddepue convinced me to give my first talk ever. I was SO SCARED but a very funny "old man" kept nodding in the audience (reassuring me that what I was saying made sense). In the end, he said "great talk!". Later on, I learned it was @joeerl :')

half–duplex 10BaseT NATEwork
@perlhack

#rememberingjoe still feeling this loss - Joe's enthusiasm was infectious; his approaches to concurrency and reliability transformative. To meet him was to make a friend, he wouldn't take no for an answer :)

Elena Lindqvist
@elenalindq

@ericsson, let's share the charming "Inside Erlang" video (and others) from the internal video channel to the Ericsson youtube channel, in memory of Joe, to hear from him how Erlang helped CERN in their hunt for the Higgs bosom. #rememberingjoe

1:38 AM · Apr 23, 2019 · Twitter Web Client

2 Likes

Ericsson @ericsson · Apr 23
Replying to @elenalindq
A fantastic idea. Thank you for that; we've now shared--I hadn't heard of his passing until just now. ///Christine

Tonny Uhlin
@TUhlin

Our thoughts are with Joe Armstrong's friends and family as we learn of his passing. Joe joined Ericsson in 1985, and within a year he had developed an early version of Erlang. In this video from 2014 he tells his story. #rememberingjoe t.eric.sn/2W3rB54 #TeamEricsson

ERICSSON
Inside Erlang – creator Joe Armstrong tells his story
Erlang is a programming language designed by Ericsson and used by a number of companies such...
ericsson.com

Jaana B. Dogan
@rakyll

Joe Armstrong had a huge influence on me as a programmer. It is painful to see him go but his ideas will stay with us for a long time. #rememberingjoe

Jamison Dance
@jergason

If I hadn't watched his talk the day before I would have had no idea who he was. He was humble and curious despite all he knew and had accomplished, and he went out of his way to make me feel good as a fairly nervous speaker. #rememberingjoe

steveklabnik
@steveklabnik

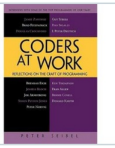
I only met @joeerl once; he was extremely kind, thoughtful, and humble. Someone asked a question while a group of us were riding the London Eye, and he pulled out his laptop to help explain. #rememberingjoe

Mario Aquino
@MarioAquino

A very happy memory of @joeerl from September, 2014. #rememberingjoe

Simon Fenton
@screamish

Joe Armstrong was clearly clever, but it was his incredible kindness in his interactions with people IRL and online that really stuck with me. I haven't seen it mentioned yet on this hashtag so I'll say it's well worth a read goodreads.com/book/show/6713... #rememberingjoe



Coders at Work: Reflections on the Craft of Progra...
Coders at Work book. Read 233 reviews from the world's largest community for readers. Peter Seibe...
goodreads.com

Guilherme de Maio
@nirev

I've never got the chance to meet Joe, and I so deeply wanted.

He always seemed like the most humble and enthusiastic person. Erlang (and Elixir) completely changed the way I think of software, and I'll always be thankful for it. #rememberingjoe

Anders Danne
@aodanne

I will always miss our long coffeebreak discussions during our Erlang projects at Ericsson. I have learned a lot from him during more than 30 years. #rememberingjoe

Ayanda Z D
@dube_aya

@joeerl made my codemesh 2015 the greatest ever. Thank You for creating something special, that restructured & shaped global telecommunications for generations to come! we'll miss you, your wisdom & all the fun you brought, so much! #rememberingjoe

Anthony Doan
@AnthonyDoan20

@joeerl You've contributed to who I am as a programmer as of today. Your talks and blog posts are the reason why I am better. The energy, passion, and kindness you give in those talks and blog posts is something I will dearly miss. You're my inspiration. RIP. #rememberingjoe

zurab
@zurab

Joe was one of the most intelligent people. But when he talked, you never felt out of touch. His attention to simplicity was inspiring. This is very rare in software development. Try #Erlang to see why. RIP #rememberingjoe

Tonći Galić
@tuxified

I always look up to folks smarter than me, @joeerl however made me feel at ease. Must have been the way he radiated kindness, curiosity and always seemed to enjoy the moment. Thank you, you will be missed #rememberingjoe

jail flerken stark pikachu
@jailandrade

Gracias por Erlang @joeerl #rememberingjoe

Iain Duncan
@iain_xornot

There are way too many people in the tech biz who call themselves gurus, and mentors, and thought leaders, and it's all bullshit. But Joe Armstrong was all of those things, for real. At least I got to know him in book form. #rememberingjoe

Roger Suffling
@RogerSuffling

#joearmstrong and #fredherbert having an interesting discussion on #erlang tools at #EUC2016

Pavel West
@pavelw

He was always a huge fan of systematic simplicity, self-contained straightforwardness, and stories from the older times of computing that gave a fresh perspective on everything. @joeerl R.I.P. #rememberingjoe

Safwan Kamarrudin
@SafwanDotErl

Replying to @mrsjoeerl @FrancescoC and @joeerl

My condolences to you and family. It's hard to overstate the influence @joeerl had on my career, going from an Erlang fanboy to someone who gets paid to work on the platform he helped create. His fun and humorous Twitter ramblings will be sorely missed. #rememberingjoe

Sonny Scroggin
@scrogson

Meeting @joeerl for the first time. September 2014 at Chicago Erlang. He told me that Elixir and Erlang were the same thing. #rememberingjoe

{:arif, :yayalar}
@ayayalar

#rememberingjoe it is sad to see you go. You will always be remembered. Thanks for everything.

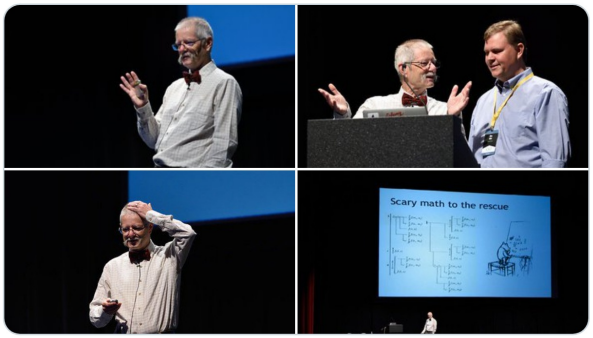
Tim Heaney
@oylenshpeegul

I had the privilege of seeing @joeerl speak at @abstractionscon a couple of years ago. I ran into him at the Pittsburgh airport on the way home and we had a lovely chat while waiting for our planes. #rememberingjoe

Strange Loop Conf
@strangeloop_stl

Joe Armstrong - Strange Loop 2014 keynote
#rememberingjoe

Translate Tweet



Claudio Ortolina
@cloud8421

In 2014 I nervously gave a talk at Elixirconf, Joe was in the audience. At the end of the talk, he came to me and asked me a few questions. I felt privileged to have an exchange with him as a peer. #rememberingjoe

Garrett Smith
@gar1t

The first time I saw Joe Armstrong I was sitting in the audience at EUC in Stockholm and there above us, standing among the rafters on some hidden platform, was a man in silhouette looking down. Too small to be Darth Vader, to big to be Yoda, I knew instantly who it was.

5:05 AM · Apr 21, 2019 · Twitter for Android

View Tweet activity

1 Retweet **17** Likes

Garrett Smith @gar1t · Apr 21
Replying to @gar1t
He looked on, surveying us, the Erlang faithful. I would return later to the Chicago user group to recount what I had learned at my first EUC, the holy see. But at that moment, all I could think was, that's the man. That's the fucking man. That's Joe Armstrong.

1 9

Garrett Smith @gar1t · Apr 21
In short order I would learn what everyone learned when they met Joe. Brilliant, curious, focused, insatiably creative. But more than anything was his generosity of spirit. I just felt *good* being around Joe. I felt *good* taking to him. He gave more than he received.

Antonio Nikishaev
@lelff

Replying to @FrancescoC and @joeerl
#rememberingjoe
web.archive.org/web/2012012815...

Felipe Ripoll
@FelipeRipoll84

5 years ago I was exhausted of Java, thanks to @candresbolanos I found “programming Erlang” and suddenly my career pointed to a different direction. I only meet you once, thanks for #erlang #rememberingjoe

Bryan Hunter
@bryan_hunter

Replying to @scrogson and @joeerl
Dining philosophers

Olympen, Grønlandsleiret 15, 0190 Oslo, Norway
June 2016

#rememberingjoe

Translate Tweet

Garrett Smith
@gar1t

Joe casting his typically big shadow.

Image by @bltroutwine

#rememberingjoe

Mathias Verraes
@mathiasverraes

I once had a conversation with @joeerl standing in line for lunch at a distsys conf. He answered my newbie questions with the same passion as he'd answered the more interesting panel questions the day before. #rememberingjoe

Fernando Alonso
@krakatoa1987

Love and light to @joeerl family.
#rememberingjoe

> **Francesco Cesarini** @FrancescoC · Apr 20
> It is with great sadness that I share news of Joe Armstrong's passing away earlier today. Whilst he may no longer be with us, his work has laid the foundation which will be used by generations to come. RIP @joeerl, thank you for inspiring us all.

David Schainker
@schainks

An all time favorite talk. Thanks for posting again @FrancescoC #rememberingjoe

> **Francesco Cesarini** @FrancescoC · Apr 23
> In the office lift (elevator for my US chums), someone I did not know heading to a different floor was watching 'The Mess We're In', @joeerl's Strangeloop Keynote: youtube.com/watch?v=lKXe3H... ... #erlang #rememberingjoe

leffel @devilherdue · Apr 5
my strongest skill as a software eng is my willingness to interrupt an entire meeting to announce "I have no fucking idea what those words mean, please explain"

there are *always* other folks who pipe up about being confused once someone else admits it first

126 1.4K 8.7K

Joe Armstrong
@joeerl

Replying to @devilherdue
I always had a rubber duck in my bag when I didn't understand anything I'd pull out the rubber duck and say "well I don't understand could you explain it to my rubber duck" and a model of the 🇬🇧 queen " explain it to her majesty" (for tricky stuff) - true

Lars Albertsson
@lalleal

I'm sad to hear of Joe Armstrong's passing. I remember many entertaining and intelligent coffee room discussions. He'd mix statements of crisp clarity with things I'd violently disagree with in a unique way. Always educational, vivid, and presented with a smile.

Garrett Smith
@gar1t

That laugh! You know what I mean. You could hear it across the room. I can hear it now.

Roger Suffling
@RogerSuffling

Joe @joeerl and the gang (including @JaneWalerud) in #Stockholm #EUC2016 discussing events that led to the historic open sourcing of #Erlang by #Ericsson #rememberingjoe

Ericsson Italia
@EricssonItalia

È venuto a mancare Joe Armstrong. Una lunga carriera in Ericsson, nel 1986 inventò #Erlang, ancora oggi alla base del funzionamento di Whatsapp, sistemi tlc e molte app. Due anni fa venne a trovare molti suoi ex colleghi nel campus Ericsson di Roma #rememberingjoe

Translate Tweet

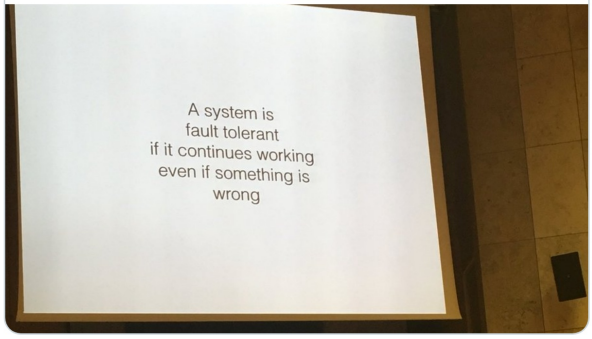
Ganesh
@gsarwate

Replying to @FrancescoC and @joeerl
Remembering Joe at @erlexsf meetup (March 2015) It was a great privilege to see him speak in person. #rememberingjoe

Michal Slaski
@michalslaski

In 2008 Joe spoke in Kraków about Concurrent Software for a Concurrent World and inspired a generation of graduates #rememberingjoe

> **Michal Slaski** @michalslaski · Apr 6, 2018
> 10 years ago @joeerl visited Krakow to present at the @sfikrakow conference and it inspired many to study Erlang. He is back today with another inspiring talk :)



Awsaf Rahman
@Awsaf174

I was a newbie in Erlang when I posted on the Erlang forum asking for some help. To my surprise, the father of Erlang Joe Armstrong himself responded and solved the problem.

He leaves a legacy unmatched. Rest in peace my friend @joeerl #rememberingjoe

Joe Armstrong <erlang@gmail.com>
to Erlang, me

I'll do this with a 4 element list to show the principle (you have a 100K list but it works the same way)

```
1> L = [1234,45,67,89].
[1234,45,67,89]
2> file:write_file("foo", term_to_binary(L)).
ok
3> {ok, B} = file:read_file("foo").
{ok,<<131,108,0,0,0,4,98,0,0,4,210,97,45,97,67,97,89,106>>}
4> binary_to_term(B).
[1234,45,67,89]
```

term_to_binary turns anything into a binary which you save in a file
binary_to_term is the inverse

Cheers
...

Matt Weagle
@mweagle

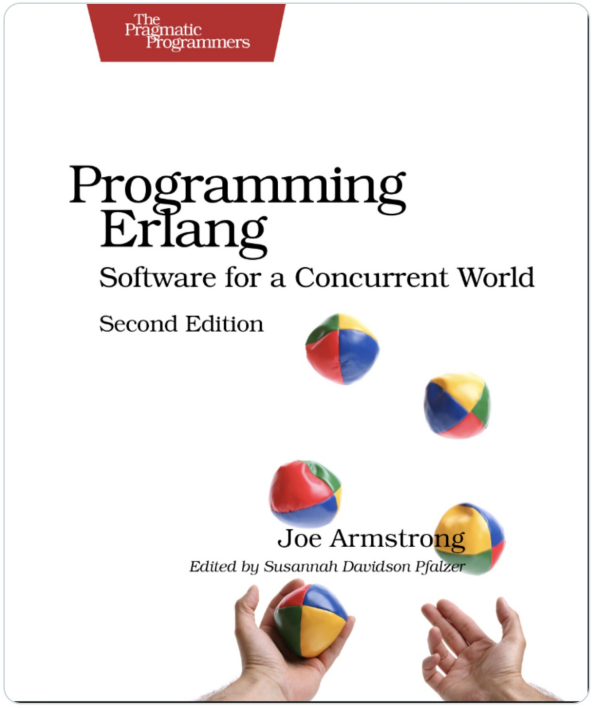
I never met @joeerl but he had a huge impact on me and many others. His voice and work was humble, curious, and brilliant. Sending thoughts to his family. #rememberingjoe

Omer Kilic
@OmerK

Can't exactly remember what the context is but we did talk a lot about hardware and he always had a good story from back when computing was not all about the cloud etc. Here's Joe and @rvirding "hacking" a cupcake at the Erlang Embedded workshop in SF, 2013.

Shuaib Afegbua
@afegbuas

Never met @joeerl, but Joe Armstrong made a huge influence on me as a software developer. Erlang/OTP/The Beam changed everything for me. THANK YOU JOE for your contributions, amazing talks and being Joe. We will keep "Letting it Crash"#rememberingjoe



Ser Brienne of Tarth
@Adeola

#rememberingjoe

> **Joe Armstrong** @joeerl · Apr 4
> One on the disadvantages of having a PhD in computer science is that I get asked really difficult questions.
>
> Like - "In gmail on my iPhone I press archive - can I get my mail back?"
> ...

Fabian van 't Hooft
@FabianHooft

Replying to @mrsjoeerl @FrancescoC and @joeerl
My condolences to you and your family. I never had the pleasure of meeting him. He was a source of inspiration and he will be sorely missed by the entire community. #rememberingjoe

Garrett Smith
@gar1t

Joe Armstrong and Alan Kay. As the story goes, Joe and team were waiting for a Smalltalk computer but as it was taking forever to arrive they started using Prolog for their initial work on Erlang. @rvirding did I get that right?

Image by @bltroutwine

Albin Stigö
@albinstigo

Hello Mike,
Hello Robert,
Goodbye Joe.