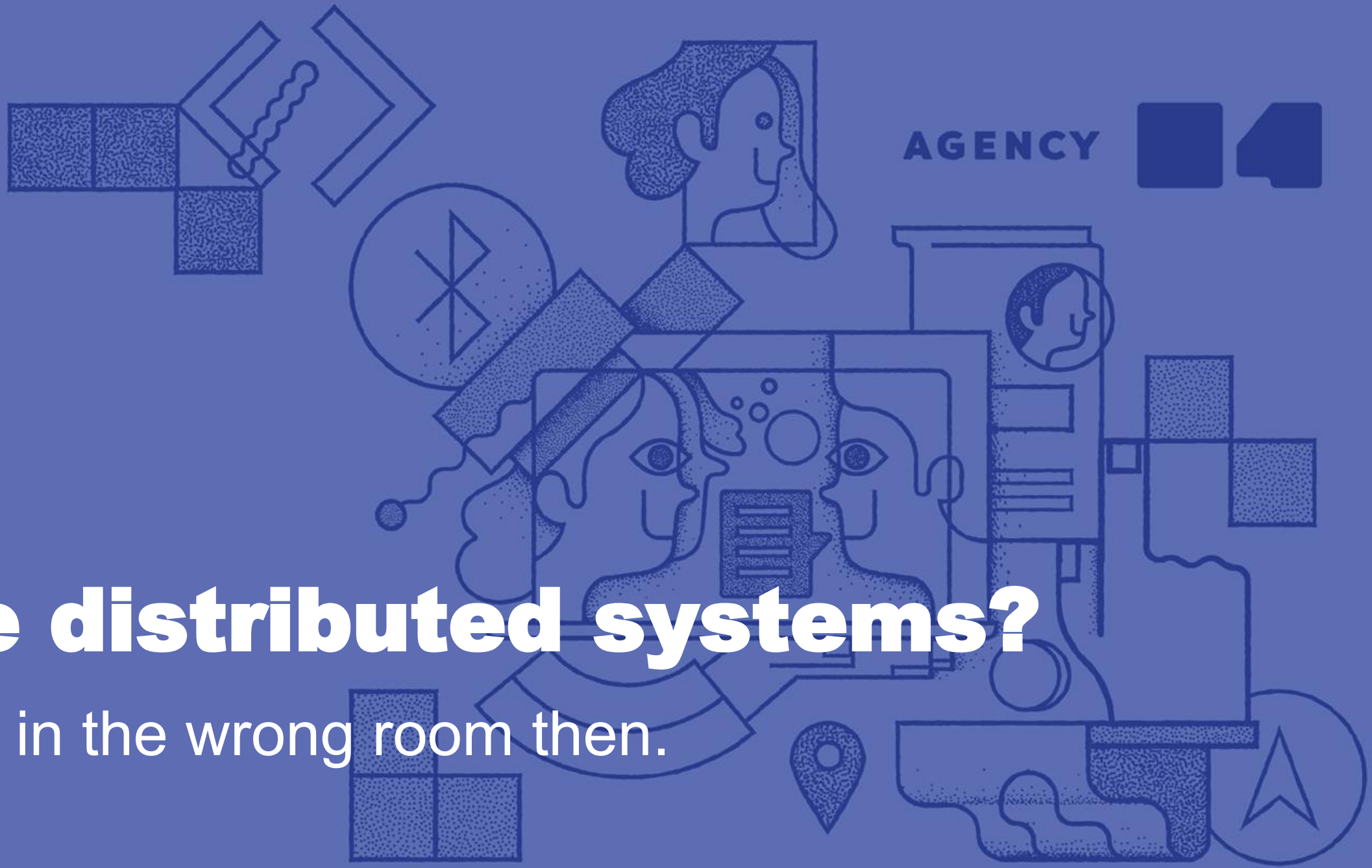
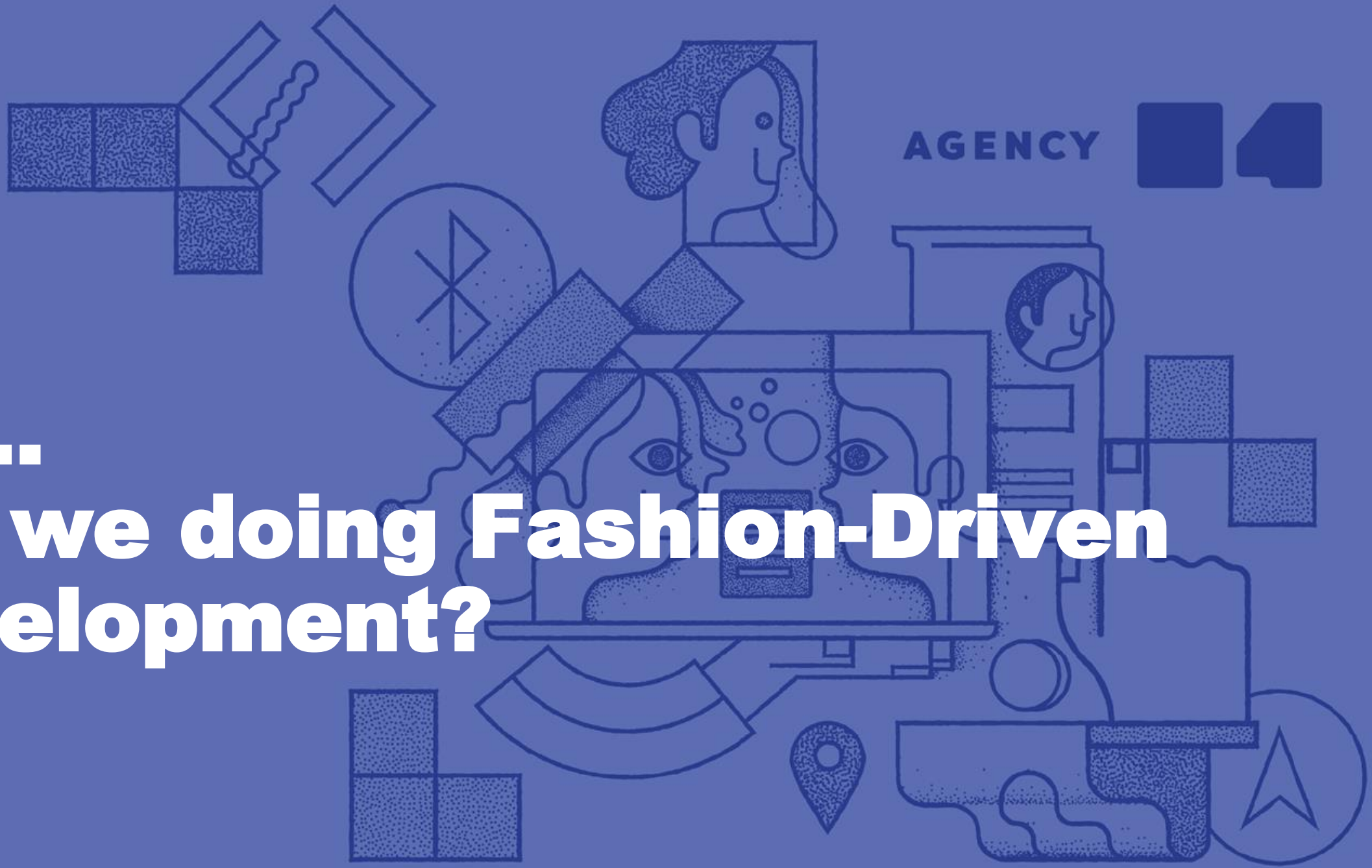


Like distributed systems?

You’re in the wrong room then.



Or ...
Are we doing Fashion-Driven Development?

Or …
Has the world gone crazy or is it me?

Disclaimer

- This is just a subjective view of one developer
- Even my (ex) coworkers have different views
- There’s a good chance that I will change my view over time

My experience (1)

- 18 years of professional work
- Mostly enterprise market
- 40+ projects
  - few products in production for more than 10 years, mutliple major upgrades
- Never worked on:
  - High traffic stuff
  - Heavy analytics
  - AI, machine learning etc...
  - Huge platforms that comprise big portion of enterprise infrastructure (many big services/applications...)

AGENCY 

My experience (2)

- Development team size: 1 to 8, usually 2-4
- Project duration: 2 months to 1 year, usually 4-6 months
- Project size: 150 to 1,500 classes, usually around 600 classes
- Traffic requirements:
  - usually less than 10 req/sec
  - sometimes few tens of req/sec
  - rarely hundreds of req/sec
    - what the hell is „request” anyway?
- Don’t know what „serious” project mean to anyone, but lot of them not considered simple
  - Price tag: from few 10,000s to few 100,000s euros
- **At least 80-90% of projects out there fit this description**?

AGENCY 

AGENCY

Initial years (up to 2010)

Development description

- Usually two full-stack developers, sometimes solo
- Monolithic, 3-tier architecture (web, service and DB layer)
- Single SQL database, using just basics (table, view, index, sequence)
- Mostly synchronous logic
- Server-side web UI
- Deployed as 2 node setup (install Java, install Tomcat, copy application *.war* file, single log file)

AGENCY

AGENCY

Advanced years

New fashion

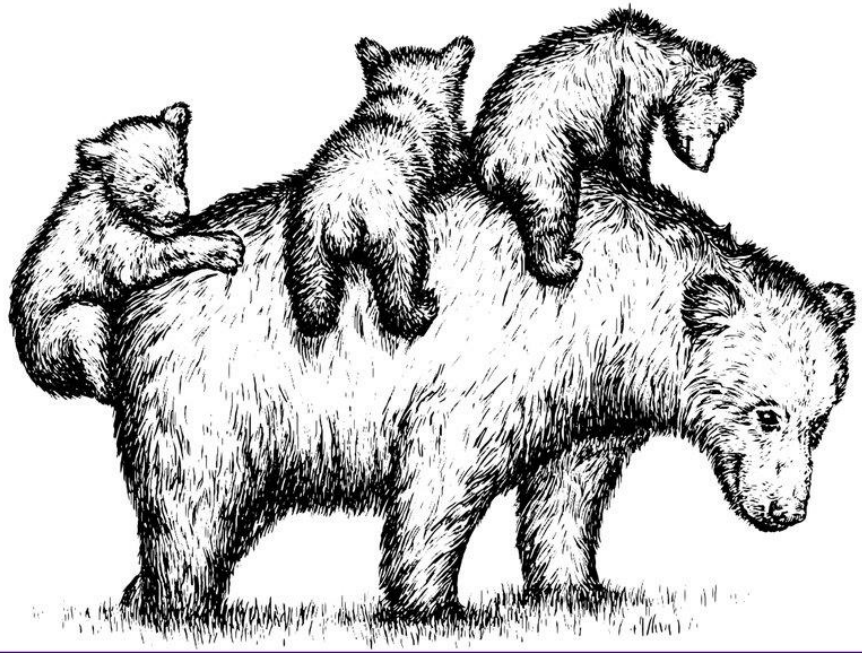
- Breaking the monolith
  A.K.A. Distributed architectures
  A.K.A. „Microservices”
- Asynchrony
- Big data
- Message-driven communication
- Cloud
- Horizontal scalability
- JavaScript front-ends

AGENCY

As far as I've seen, it **mostly** wasn't the customer that mandated change!

- Functional requirements not more complex than before
- Traffic requirements are bigger than before, but not significantly
  - BTW, performance of hardware and software stack continuosly improved over the years!

Getting the wrong idea from that conference talk you attended



Solving Imaginary Scaling Issues

At Scale

O RLY?

@ThePracticalDev

AGENCY

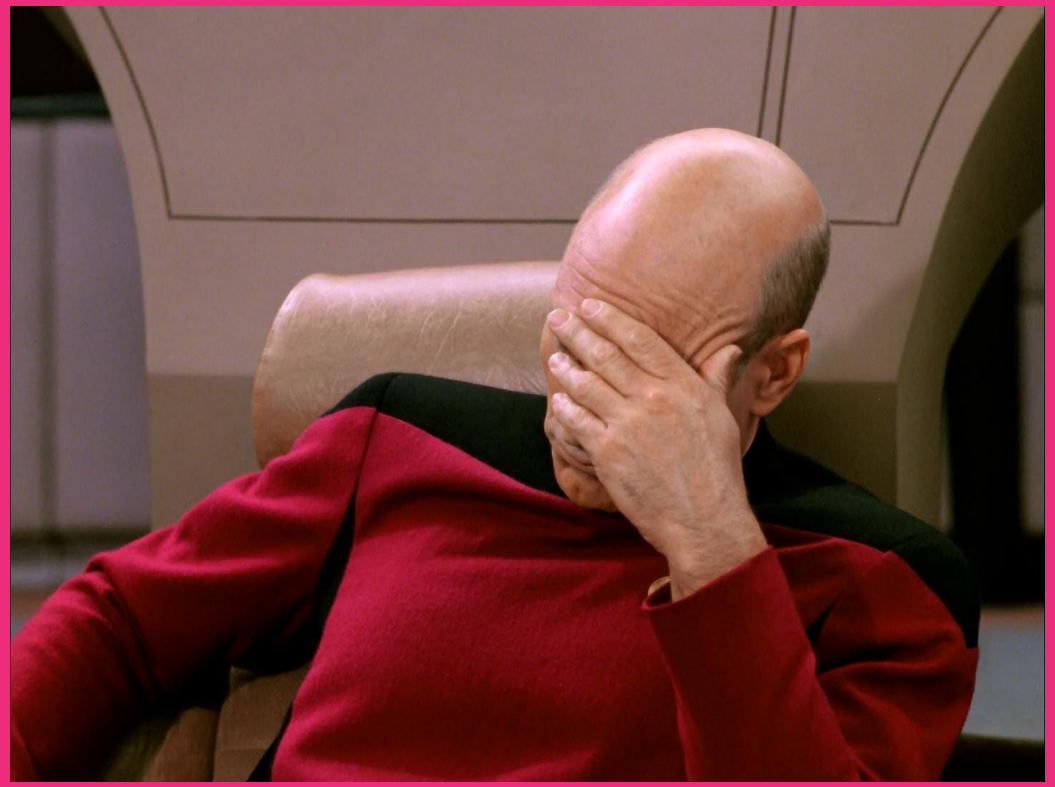
End result

- No, the projects haven’t failed!
- The projects delivered within deadline
- The projects had sufficient quality. Or not.
- But, what about:
  - Amount of resources required?
  - Amount of testing required?
  - Amount of knowledge required?
  - Development moral?
- **But your CVs will look awesome!** ☺

AGENCY

AGENCY

Microservices



First rule of distributed systems:

Don’t distribute!

Microservices - Pros

- Justification – „decoupling”
- But there are many types of decoupling – process decoupling is just one (heavyweight) type
- Pros:
  - Stronger code decoupling (no „big ball of mud”)
  - Execution/memory isolation - autonomy
  - Separate scaling strategy
  - DB or language flexibility

Microservices – Cons (1)

- Remote instead of local call
  - Code complexity
  - Rubustness?
  - Speed?
  - Compile safety?
  - Tool support?
- Global data consistency lowered → no strong references between multiple service models (foreign keys, object references...)
- APIs frequently batch oriented, not fine grained

Microservices – Cons (2)

- Redesigning much harder due to more people/components involved →premature architecture freeze
- Service versioning
- End-to-end testing much harder
- No single-thread debuggability (single stack trace etc...)
- Various little things, such as jumping between multiple log outputs, multiple IDE projects ...

AGENCY

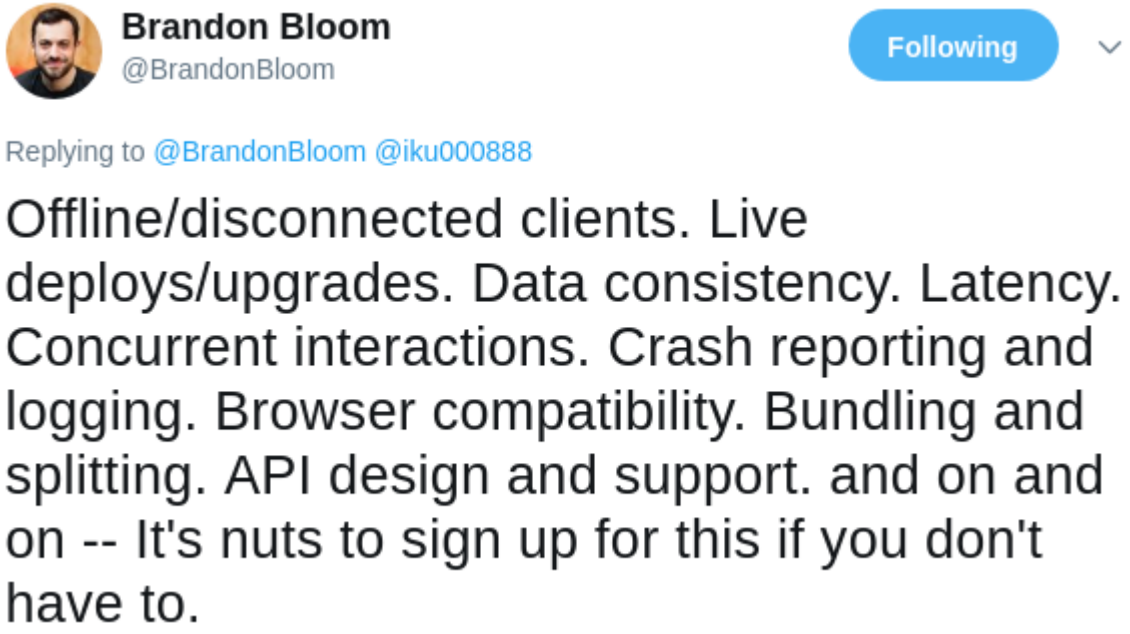
JavaScript front-ends

JavaScript front-end - Pros

- Much more powerful than server-side - more attractive/responsive UI controls
- Faster (no need for server request/response for each action)
- Client-side caching
- No messing with:
  - „Back” button
  - Much easier to implement complex form (server-side „wizards” require session state)

JavaScript front-end - Cons

- Burden of additional language
  - Human resource management increase
  - Harder knowledge sharing
  - Language is easy, whole ecosystem is required:
    - frameworks, libraries, build tools
- Front-end and back-end are now distributed system!



AGENCY

- Strive for simplicity; good code should be obvious and boring, not adventure
- If you can improve something by removing, not adding more stuff, that’s awesome!
- New tech’s drawbacks are rarely visible at first – be cautious with evaluation
- Every project is exploration, there are multiple ways to approach the problem – so explore!
- Don’t drown yourself in low level problems, question also high level decisions - there are usually tremendous improvements waiting to be discovered