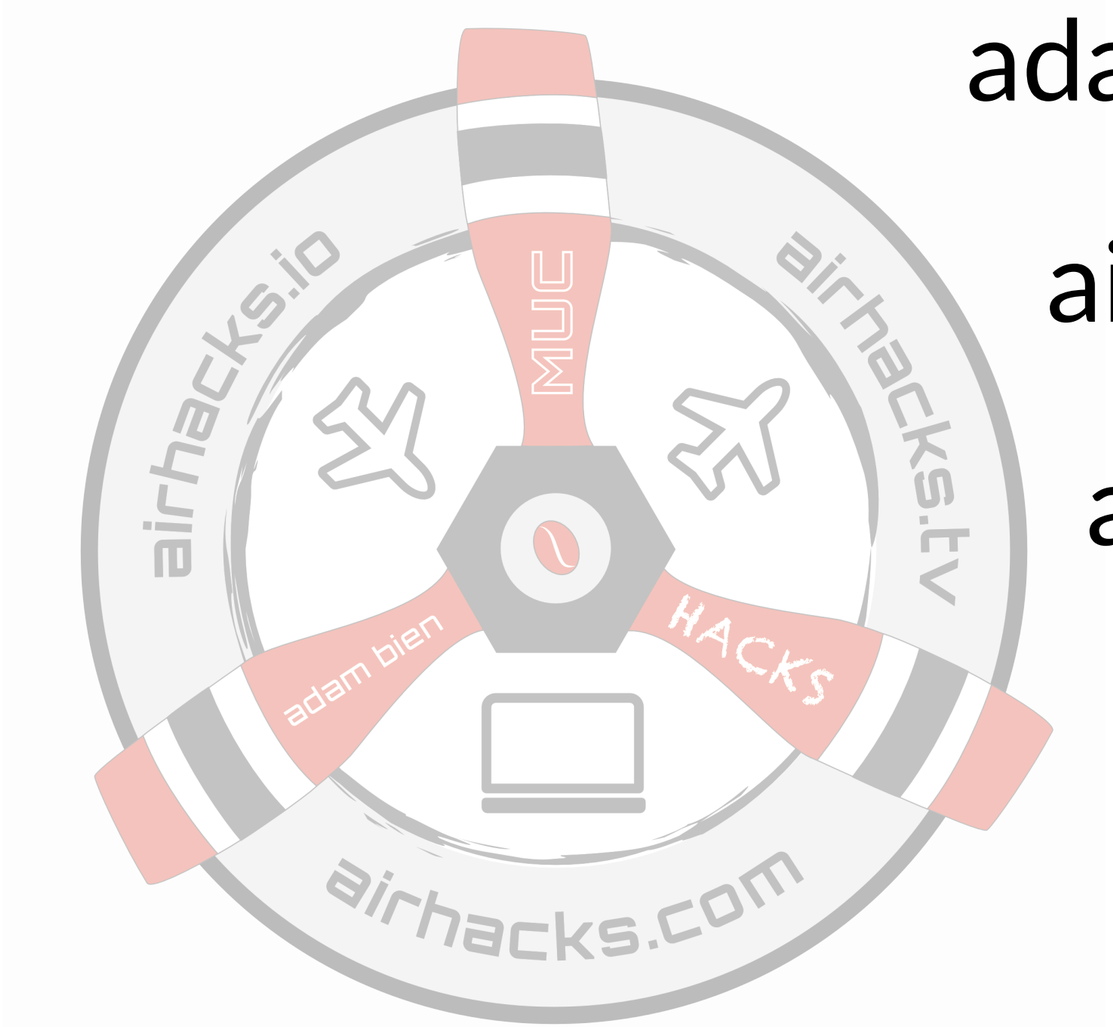
What Should Happen in 2021

…with Java Backends and Web Frontends

"It's not work if you like it"
...so I never worked. #java



adambien.blog

airhacks.com

airhacks.live

airhacks.io

airhacks.fm

Brief History of Web Frontends

1. Fat Clients
2. Editor + HTML (...XHTML) on incompatible browsers
3. Rich Clients
4. Web 2.0, then HTML 5
5. jQuery, Backbone + Co.
6. Web Standards
7. ES 2015 with builds and transpilations
8. Dependency Injection, code generation ...in a browser
9. Evergreen, compatible browsers with powerful APIs

Frontends now

1. Libraries like jQuery, XHR-wrappers became built-in browser APIs
2. Frameworks become less relevant
3. Web Components are available everywhere
4. CSS layout was fixed
5. Proprietary CSS frameworks became optional
6. Compatibility is a "cool" feature
7. JavaScript becomes more and more similar to Java
8. Java FX is an interesting option for iOS and Android

Brief History of Java Backends

1. Sun Java Web Server, W3C's Jigsaw, JWSDK
2. approx. 20-50 different application servers with incompatible APIs
3. J2EE vs. Tomcat
4. Lightweight runtimes (aka: 5 MB Tomcat with 50+ MB WAR :-))
5. More XML than Java code
6. Java 5 and annotations, Java EE 5, Configuration by Exception, Convention over Configuration
7. @Inject and CDI => Java EE 6
8. ThinWARs, kB-WARs and the end of shared deployments
9. CNCF, and the raise of MicroProfile
10. Micro runtimes on standard APIs
11. Standard APIs (JDBC, JNDI, JMS, Servlets) are available for 20+ years

nothing compares to ...code

Fallacies

1. Vanilla Web Components are not productive - you need a framework
2. You need npm, minifcations, uglifications, transpilations and code generation to be an effective web developer
3. Public clouds are cheaper than on-premise infrastructure
4. Kubernetes + micro services are the only way to go
5. Functions are a general-purpose programming model
6. Java-based serverside rendering is pointless
7. API stability doesn't matter
8. Throughput and performance don't matter any more
9. Memory footprint is everything

What should (will) happen

1. Learn fundamentals once, apply everywhere
2. API: stability, SPI added value
3. Fatigue with esoterics
4. Write less code

What should (will) happen with backends

1. GraalVM: polyglot + next generation micro runtimes
2. CLIs and functions / serverless: with Java and GraalVM
3. Event-driven runtimes make reactive programming viable
4. Clear separation between the runtime and the business code (container image layering). Uber-jar / fat-jars are problematic on containers
5. Java becomes more and more like ...JavaScript (destructuring, text blocks, lambdas, var)
6. Competition around MicroProfile: Helidon vs. Quarkus vs. Payara vs. OpenLiberty vs. Piranha Cloud vs. KumuluzEE (...)
7. Faster cadence dictated by external forces like e.g. CNCF
8. Focus on developer joy, developer experience and productivity
9. Meaningful observability matters

What should (will) happen with frontends

1. Vanilla Web Components / raise of libraries / frameworks are less relevant
2. More and more framework features will move to the browser APIs
3. Browsers get more desktop features
4. Less browsers (=bad) <-> more compatibility (=good)
5. AR, VR could resurface again with new devices
6. Infrastructural JavaScript frameworks are nice to have
7. Less tooling on developer machines, back to the roots

airhacks.live

NEW online, live virtual workshops

"like airhacks.com, without leaving your home"

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Live, Virtual Online Workshops, Winter 2020:

MicroProfile with Quarkus, December 10th, 2020

Micro Frontends with Web Components, lit-html and redux, December 11th, 2020

Live, Virtual Online Workshops, Spring 2021:

Building Event-Driven Applications with Streams, Logs and Messages, March 23rd, 2021

by and with adam-bien.com


live stream
Q&A
airhacks.tv

Thank You!

and see you at:

workshops.adam-bien.com

meetup.com/airhacks

airhacks.eventbrite.com