FP for WebDev

advantages, approaches, problems and solutions

About me

- Alexey Kachayev
- CTO at KitApps Inc.
- Open source activist
- Functional programming advocate
- Erlang, Python, Scala, Clojure, Haskell, Go
- kachayev <$> twitter
- kachayev <$> gmail
- kachayev <$> github

Disclaimer #1

It's hard to tell "is this language FP"?
We will talk about: Erlang, Scala, Clojure, Haskell

Disclaimer #2

Web is only one way to communicate.
We will talk “mostly” about web.

“Popular web” Monolit

- PHP: Zend, Symphony
- Python: Django
- Ruby: Rails
- Java: Spring, Play
- ...

“Popular web”

Monolit

- Easy to start
- HTML-driven development
- Templates-driven development

Service-oriented

- "Microframeworks" era
- Python: Flask
- Ruby: Sinatra
- PHP: Silex

Service-oriented

Why?

- Much more functionality
- Much more users
- Bigger development teams
- Many dynamic requests from each user (ajax)
- Frequent changes

Service-oriented

- High modularity
- Pluggable architectures
- Independent parts
- Independent development
- API-driven, protocol-driven

Event-driven

- Python: Twisted, Tornado, Gevent, Eventlet, Tulip
- PHP: React, PhpDaemon
- Ruby: EventMachine, Celluloid
- JS: node.js

Event-driven

- layered architectures
- 3rd party integrations
- requests/response sealing
- ... but async is hard
- ... but "do not block" is hard
- ... but error handling is hard (i.e. exceptions are hard)
- ... but users still depend on each other

And now?

- Much more users (growing fast)
- New fields (SaaS, PaaS, IaaS) with new problems
- Sophisticated dynamic and desktop-like interfaces
- Not only HTTP (WS, RTC)
- Real-time communication (i.e. chats, games)
- Real-time collaboration (i.e. games, trello)
- Hyper-text? come on...

What the problem is?

- request-reply doesn't work (almost at all)
- concurrency is hard
- fault-tolerance is hard
- p2p scaling is hard
- vertical scaling is hard
- everything should be done yesterday

What do we need?

- easy and transparent concurrency
- easy way to write async code
- composability and maintainability in sophisticated systems
- fault-tolerance made right
- easy multi-core consuming
- easy support for new protocols
- etc...

Erlang

- message-passing concurrency
- multi-core support
- distributed systems without changes in code
- “super” stable VM
- “hard to broke” something
- interactive development (even on running production)

Erlang is good for...

- API servers (i.e. REST, RPC)
- SaaS/PaaS/IaaS cloud "without downtime" (SLAs etc)
- as infrastructure (2-level OS)
- streaming servers (i.e. Erlyvideo)
- real-time communication (i.e. Cowboy websockets)
- "super-light" client-side and shared contexts (i.e. Nitrogen)

Hard to do with other stack(s)

- SaaS/PaaS/IaaS cloud "without downtime" (SLAs etc)
- as infrastructure (2-level OS), load balancers etc
- streaming servers
- real-time (or near real-time) communication
- "super-light" client-side and shared contexts
- p2p

And more...

- ChicagoBoss (rails?)
- ErlyDTL (django?)

Cases

- Facebook (Chat, Chef)
- Amazon (SimpleDB)
- Yahoo! (Delicious)
- Whatsapp (Messaging)
- RabbitMQ (AMQP)
- Velti (SMS gateway)
- Heroku (Routing)
- Github (gh-pages)
- Bugsense (data analysis)
- Blizzard, Wooga (games)

Scala

- JVM based, high performance
- (mostly) functional paradigm
- expressive type system with ATD and type inference
- less code with more results
- message-passing concurrency (prev. Akka)
- ... XML is native Scala type

Scala is good for...

- DSLs
- easy-to-read async code
- API servers (i.e. with Scalatra, BlueEyes etc)
- service-oriented architecture (i.e. with Finagle)
- real-time communication
- “super-light” client-side (i.e. with Liftweb)
- if you already use Java...

And more...

- Play 2 (if you already work with Play)

Cases

- AOL
- Twitter
- LinkedIn
- Foursquare
- Meetup
- Remember the Milk
- Boundary
- Quora
- StackMob
- Simple
- Yammer

Clojure

- JVM based
- functional paradigm, lisp-family
- STM concurrency model, multi-core support
- modern language, active development, fast-growing community
- interactive development
- macroses
- JavaScript compiler (ClojureScript)

Clojure web stack

- Netty, Http-kit, Aleph, Mongrel2
- Ring
- Compojure
- Enlive, Laser, Hiccup
- ... ClojureScript
- ... Javelin, Domina
- Build your own framework from parts!
- Old: Noir, ClojureScript One

Clojure is good for...

- BigData
- Real-time collaboration (i.e. task boards)
- Concurrent documents editing
- Multiplayer games
- Experiments and rapid prototyping
- Client-side reactive programming (sophisticated interfaces)

Cases

- Twitter (Backtype, Storm)
- Factual
- KamaGames
- Prismatic
- Groupon
- Disqus
- Answers.com
- Simple

Haskell, OCaml

- functional languages, "ML-family"
- expressive type system with ATD and type inference
- high level of code modularity and composability
- compact declarative syntax
- green thread
- math basement, compile-time program verification

Haskell is good for...

- functional reactive programming (i.e. reactive-banana, Elm)
- data parsing and types validation
- compiled HTML (i.e. BlazzedHtml, Hamlet)
- fun and cool stuff :)

And more...

- Happstack, Snap, Yesod, Scotty (Haskell)
- ... if you already works with Haskell
- ... if you want something really fast but not in C
- ... if you want to make cool and fun things

Cases

- Echo (Erlang, OCaml)
- Selectel (Erlang, Haskell)

How to start?

- do something simple
- do something interesting
- find problem (pain?) and solve it
- do not try to rewrite your project line-by-line

Links

- https://github.com/extend/cowboy
- http://nitrogenproject.com/
- http://liftweb.net/
- https://github.com/twitter/finagle
- http://www.scalatra.org/
- https://github.com/ring-clojure/ring
- https://github.com/weavejester/compojure
- http://pedestal.io/
- http://himera.herokuapp.com/synonym.html
- http://snapframework.com/

IWBYP

http://iwbyp.chris-granger.com/

Questions?

* other talks: https://kachayev.github.com/talks
** me: http://twitter.com/kachayev