

A song of types and errors

or how I found myself leading teams of Scala
developers

A song of types and errors

or how I found myself leading teams of
developers

<shiny new tech>

Agenda

- ◆ « War stories »
- ◆ Some tips and tricks
- ◆ Really poor metaphors
- ◆ SPOILERS



The sunset Sea

Bay of Ice

BLADEWATER

IRON ISLANDS

SFIELD
ISLANDS

WESTEROS

The Shiverin

The Narrow Sea



WINTER
IS
COMING



The life and career of a
mainstream developer



A man with light brown hair, wearing ornate silver and gold armor, stands next to a dark horse. He is looking off to the side with a serious expression. In the background, another person in dark clothing is partially visible.

A Lannister
always pays
his debts.

A knight in ornate armor stands next to a dark horse, looking off to the side. The background is a textured wall.

A developer
always pays
his debts.

A knight in ornate armor, including a helmet and gauntlets, sits on a dark horse. He has long, light-colored hair and is looking slightly to his left. The background is dark and textured.

A developer
always pays
his debts

technical



A developer
always pays
~~others~~^{technical} debts

Pros and Cons of a mainstream job

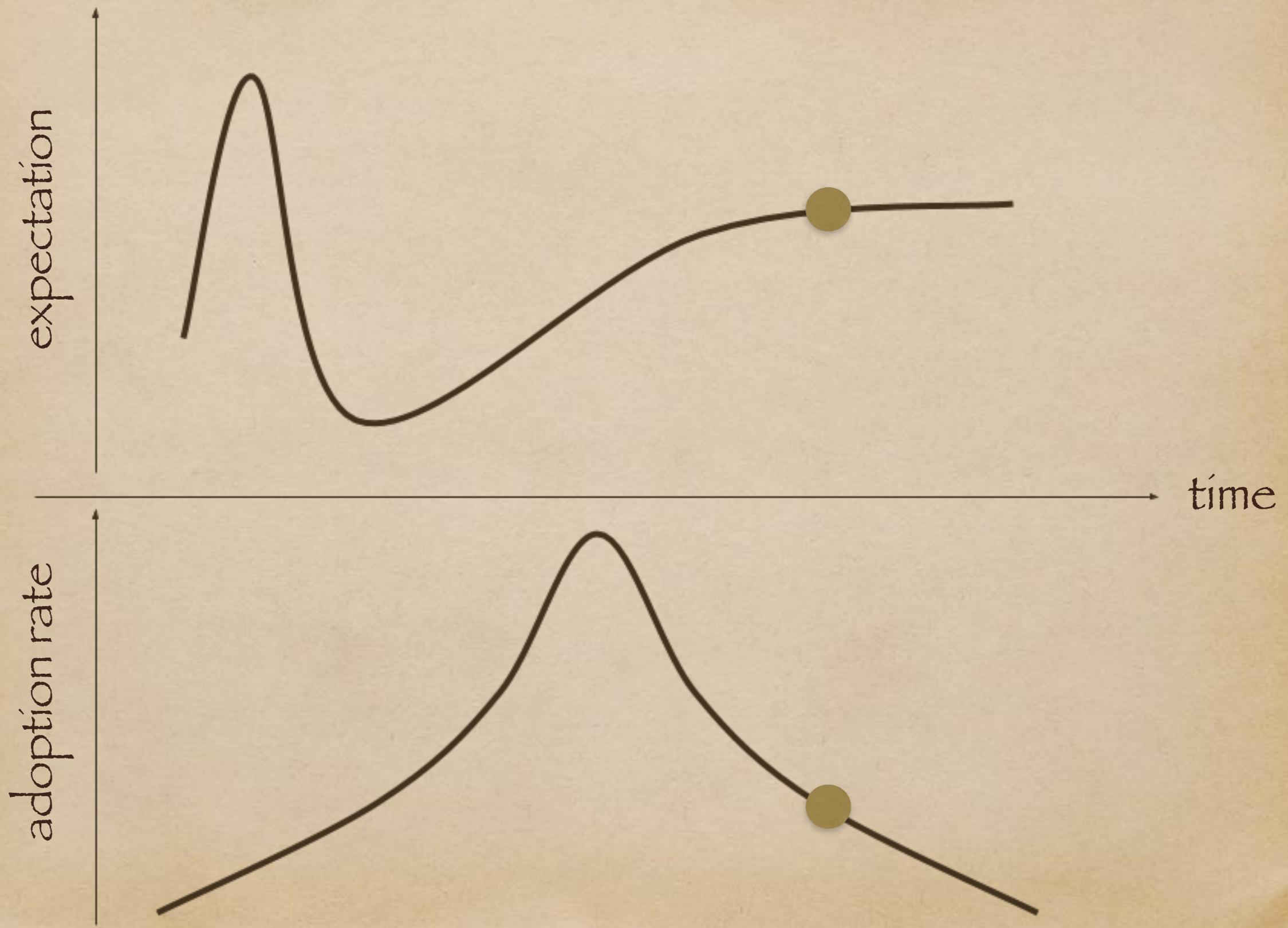
Pros

- ◆ easy to quit a boring job
- ◆ no big pressure to get better

Cons

- ◆ easy to fall into a boring job
- ◆ the « I've seen everything » feeling

Using a mainstream technology



Embracing the New

Searching for a <shiny new job>

- ◆ By definition, not many jobs available
- ◆ Create your own <shiny new tech> job
- ◆ Find or create your local user group
- ◆ Go to conferences, connect with people (gitter, IRC, ...)
- ◆ Be patient...



The
???
syndrome



A close-up portrait of a young man with light brown hair, wearing a detailed golden crown with multiple points and a textured, brownish-gold jacket over a dark blue shirt. He is looking slightly to his right with a neutral to slightly smiling expression. The background is blurred with warm autumnal colors.

The impostor syndrome

S*** getting real















What
have I
done ?

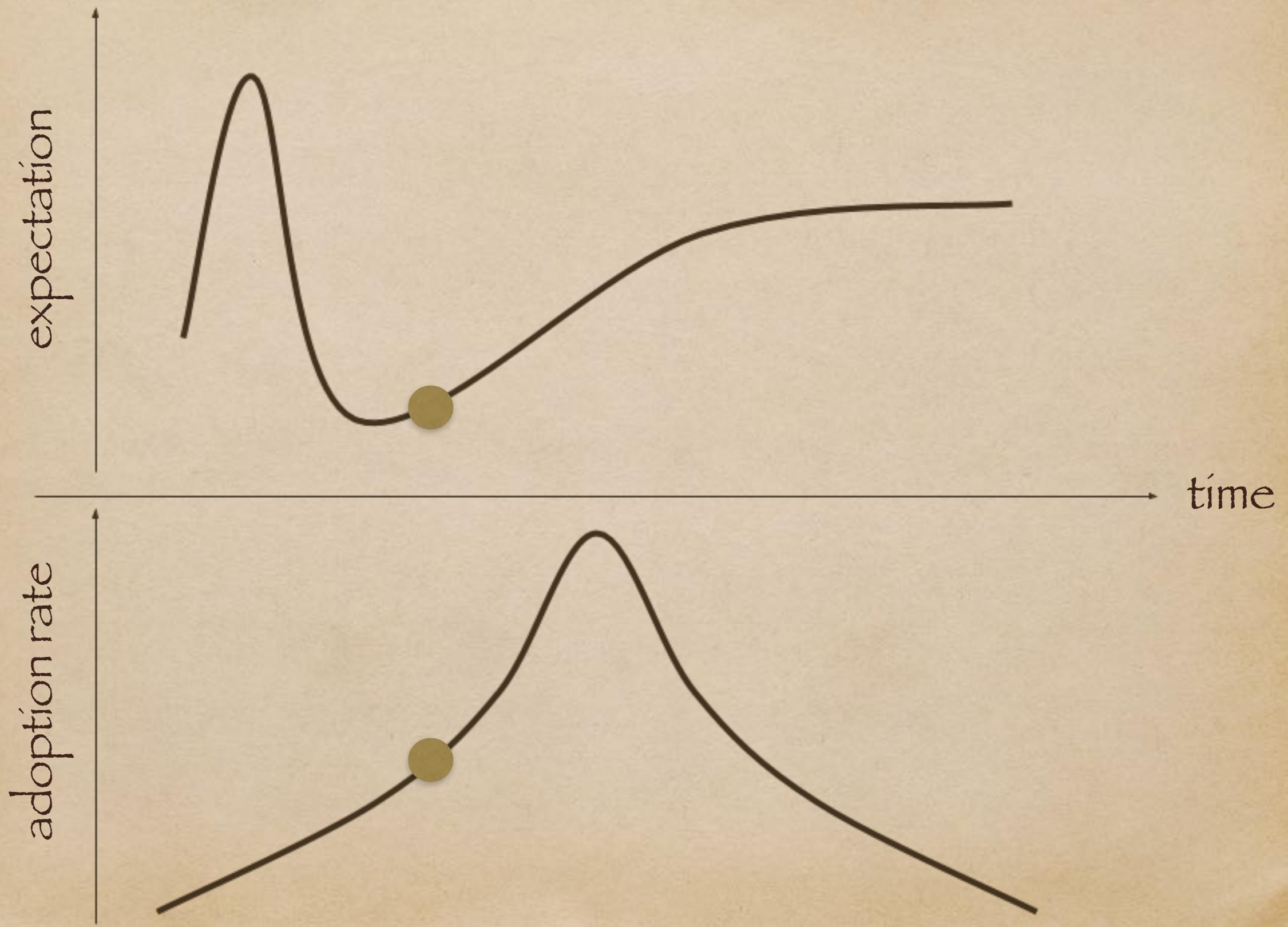


You
know
nothin'
<your name here>



Mistakes awaiting to be
made

adopting a <shiny new tech>





You
know
nothin'
Valentín



The noob paradox

- ◆ I know ~~nothing~~ just enough to glimpse at what I dunno
- ◆ The few I know already gives me tremendous power



Hic sunt dracones

A photograph of a dark-colored horse standing on top of a stone wall or castle rampart. The horse is looking down towards the camera. The background shows a bright sky with some clouds and another horse visible in the distance. The wall is made of large, rectangular stones.

Enemy #1
« overcleverness »

overcleverness

(that's not an actual word)

- ◆ « hubris » (not the framework) applied to programming
- ◆ abusing language features you don't master yet

Lesser enemies

- ◆ Lack of time to learn
- ◆ Lack of knowledge diffusion



Shameful code, please disregard

```
def handleStartSignUp = StackAction { implicit request =>
  startFormEnhanced.bindFromRequest.fold (
    errors => {
      BadRequest(startSignUpTemplate(errors, pharmacy, request))
    },
    userData => {
      val id = PatientIdentity.buildPatientId(PatientIdentity.buildPatientSocialId(userData.userName, pharmacyId))
      if(pharmacy.users.contains(id)){
        val f = startFormEnhanced.fill(userData).withError(FormError(NickName, Messages(UserNameAlreadyTaken)))
        BadRequest(startSignUpTemplate(f,pharmacy, request))
      } else {
        // check if there is already an account for this email address
        use[InMemoryUserService].findInPharmaByEmailAndProvider(
          pharmacy,
          userData.mail,
          providerId) match {
          case Some(user) =>
            // user signed up already, send an email offering to login/recover password
            mailerUtil.sendAlreadyRegisteredEmail(pharmacy, user)
          }
          case None =>
            //Here we want to create the account and lock it thanks to a token
            val patientSID = PatientIdentity.buildPatientSocialId(userData.userName, pharmacyId)
            val patientId = PatientIdentity.buildPatientId(patientSID)
            val pw: PasswordInfo = Registry.hashers.currentHasher.hash(userData.password)
            val e1 = PatientSignedUp(pharmacyId, userData.userName, userData.firstName, userData.lastName, userData.email, pw)
            val e2 = PasswordChanged(pharmacyId, patientId, PasswordInfoForEvent(pw.hasher, pw.password, pw.salt))
            val e3 = PatientTokenProvided(pharmacyId, patientId, Token(UUID.randomUUID().toString), TokenDuration(10, TimeUnit.DAYS))
            EventService.send(pharmacyId, PatientCreated(Seq(e1, e2, e3)))
        }
      Redirect(onHandleStartSignUpGoTo).flashing(Success → Messages(ThankYouCheckEmail), Email → userData.mail)
    }
  )
}
```

Slightly less shameful code...

```
def confirmOrder = SecuredBasketAction {
  implicit request: SecuredBasketRequest[AnyContent] =>
    val ph = request.pharmacy
    val pid = ph.id
    val bid = request.basketId
    val result: SimpleResult \/ SimpleResult = for {
      u           ← request.user                                \/> Forbidden
      form        = acceptCGV.bindFromRequest()
      accept      ← form.value                                 \/> BadRequest(cgvTe
      basket      ← ph.orders.get(bid)                         \/> NotFound
      fees        ← ph.deliveryFees(basket)                   \> InternalServerEr
      payzenData  ← \/.fromEither(PayzenService.prepareData(basket2Payzen(request
      prices      = PricesSealed(pid, bid, Basket.retrievePrices(basket, ph.catal
      reg         = OrderRegistered(pid, bid, PatientIdentity.buildPatientId(u.id
      revision    ← \/.fromEither(EventService.send(ph.id, BasketCheckout(Seq(pric
    } yield RedirectAfterSend(revision, orderConfirmationRoute)
    result.toEither.merge
}
```



Lesson #1 :

Code review is not enough

- ◆ The notion of good code changes over time
- ◆ => today's acceptable code is tomorrow's bad code

Getting better
at
The Game

The aim of FP

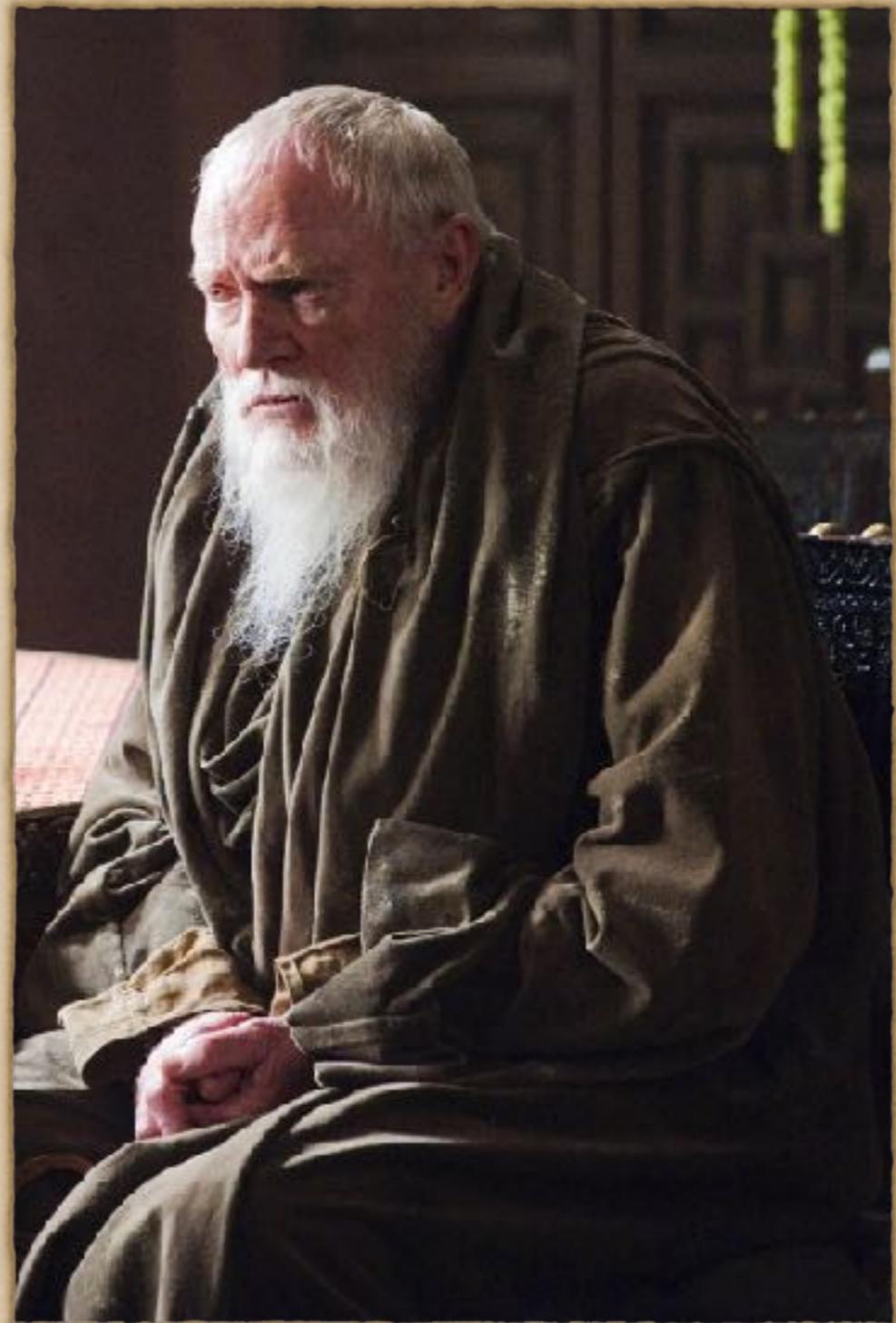
- ◆ Less boilerplate
- ◆ Less bugs
- ◆ Productivity +++



A woman with long blonde hair, wearing a blue and gold robe, is riding a large, dark brown dragon with a spiny back and a wide-open mouth showing sharp teeth. The dragon is standing on a rocky, grassy hillside under a clear sky.

Taming dragons

Step 1
Thinking
hard



Writing down the problem

- ◆ validate JSON payload \Rightarrow JsResult[X]
- ◆ validate form \Rightarrow Form[X]
- ◆ service method \Rightarrow Future[Option[X]],
Future[Either[Err, X]], Future[X], ...
- ◆ in the end we want Future[Result]

Step 2

Searching the Web



Learning with no time to learn

- ◆ Gather « knowledge sources »
- ◆ Map concepts/libraries to notions/usages
- ◆ When a problem arises reverse the mapping
- ◆ Learn in depth only when needed

Epiphany : Monad Transformers

- ◆ What I needed to learn was monad transformers
- ◆ All the previous types can be shovelled down to a EitherT[Future, Result, X]
- ◆ With a unique monad transformer, I can finally get my single for-comprehension

Monad-transformers FTW

```
def addOrUpdateQuote(providerId: BSONObjectID, jobId: BSONObjectID) = P
implicit request =>
for {
    quote      ← request.body.read[Quote]          ?
    user       ← userService.findById(providerId) ?
    _           ← user.individual.flatMap(_.birthDate) ?
    _           ← handle(AddOrUpdateQuote(user, quote)) ?
} yield NoContent
}
```

<https://github.com/Kanaka-io/play-monadic-actions>

A dark, atmospheric scene of a flooded city at night. The water reflects the surrounding environment, creating a greenish glow. In the background, there are silhouettes of buildings and trees. The overall mood is mysterious and somber.

Step 3
Refactor
everywhere

Step 4 Profit



A photograph of a woman with shoulder-length brown hair, wearing a dark green t-shirt, standing on a beach. She is looking off to the right of the frame with a neutral expression. The background shows a calm sea with distant lights from what appears to be a city or pier under a cloudy sky.

Living
like
impostors

Plot twist



I am
not
dead

BTW, you know nothin'



The leader's tools

- ◆ Improved tooling
- ◆ Project skeletons
- ◆ Pair programming



Improving on code review

1. Define a set of written (best) practices
2. When a file is modified, enforce practices during review (in the whole file)
3. Revise practices with the team as everyone gets better
4. Repeat from 1.

Brace
yourselves



<shiny new tech>
is coming

Keeping in touch

- ◆ @ValentínKasas / @ParísScalaUG
- ◆ <https://github.com/vill>

Q & A