



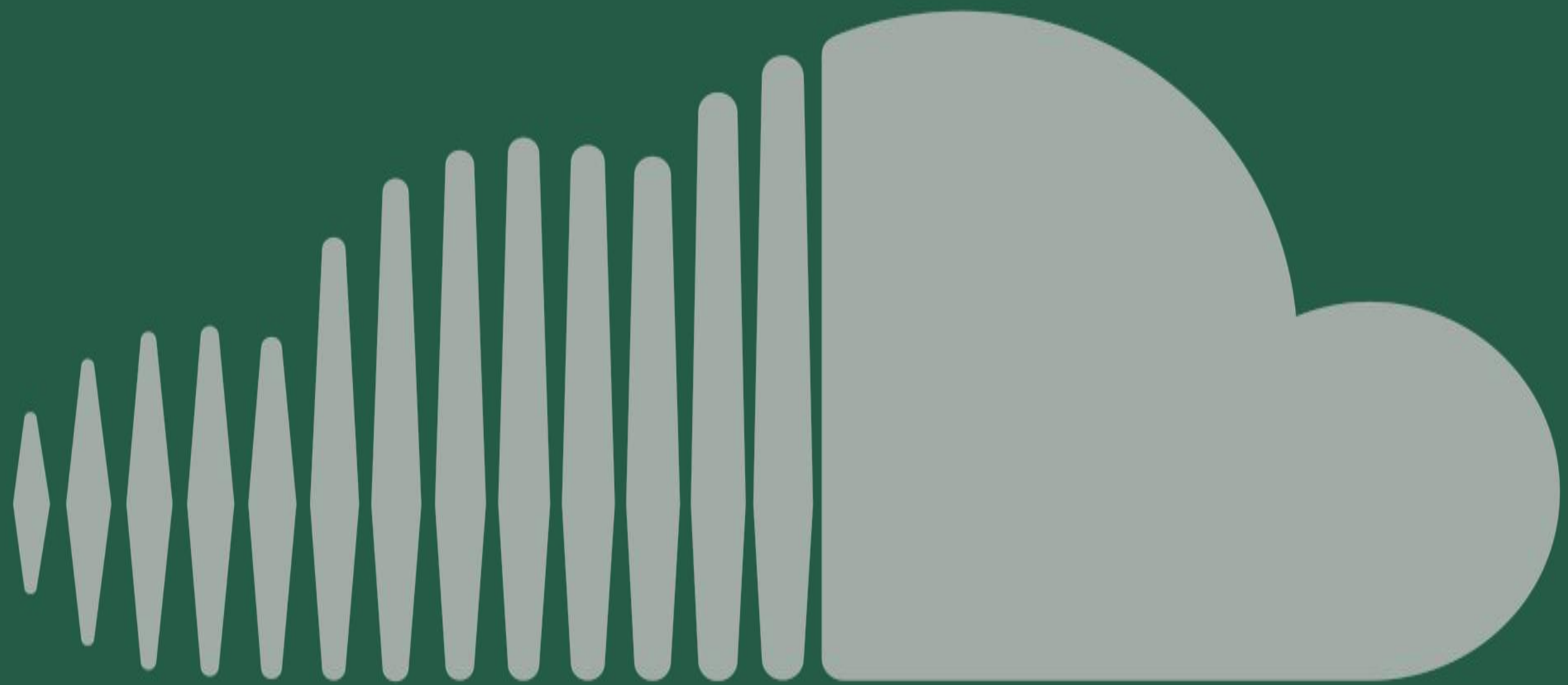
CASE STUDY:

TYPE-LEVEL PROGRAMMING IN THE REAL WORLD

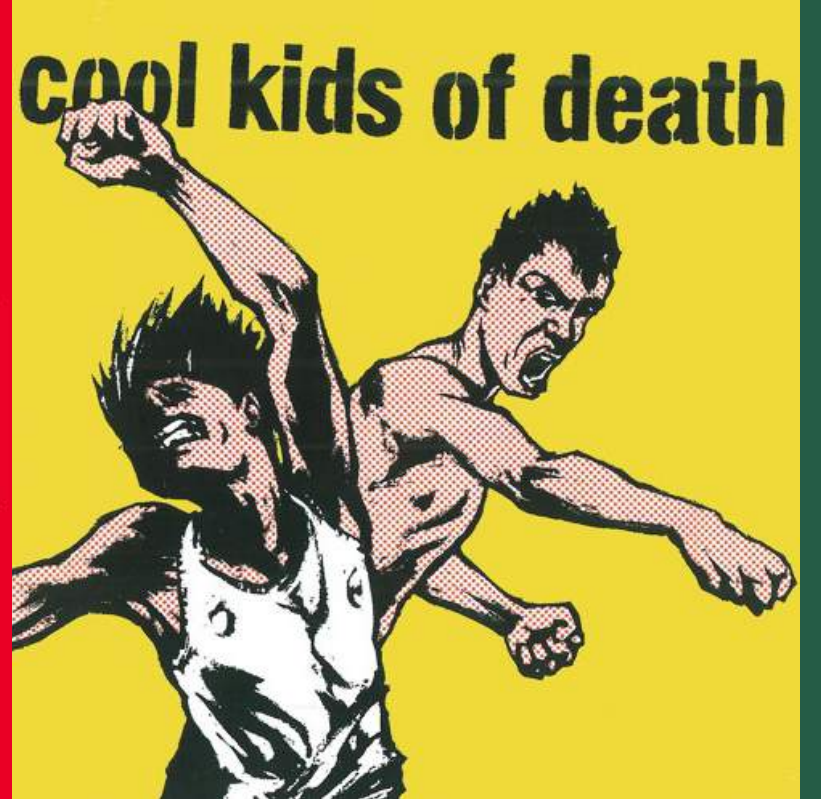
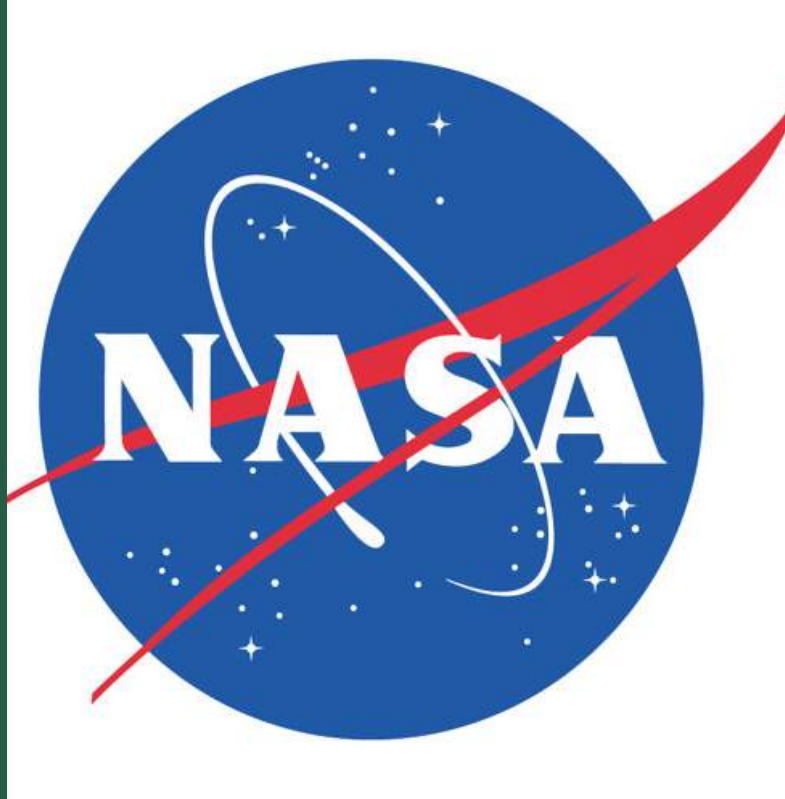
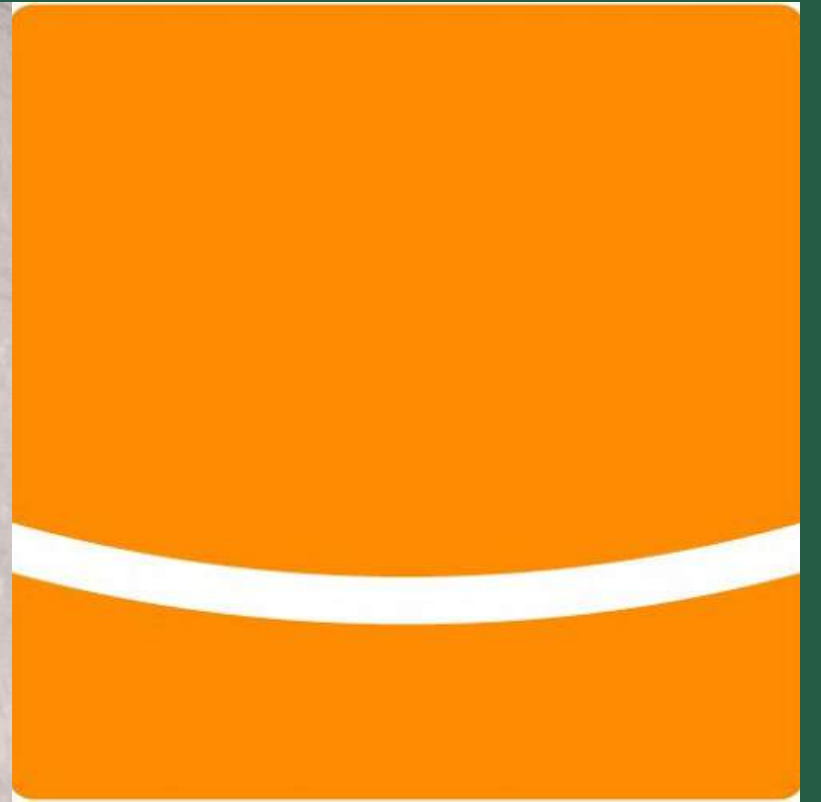
MY NAME IS @FOLONE

[HTTPS://GITHUB.COM/FOLONE/SCALAR-CONF](https://github.com/folone/scalar-conf)



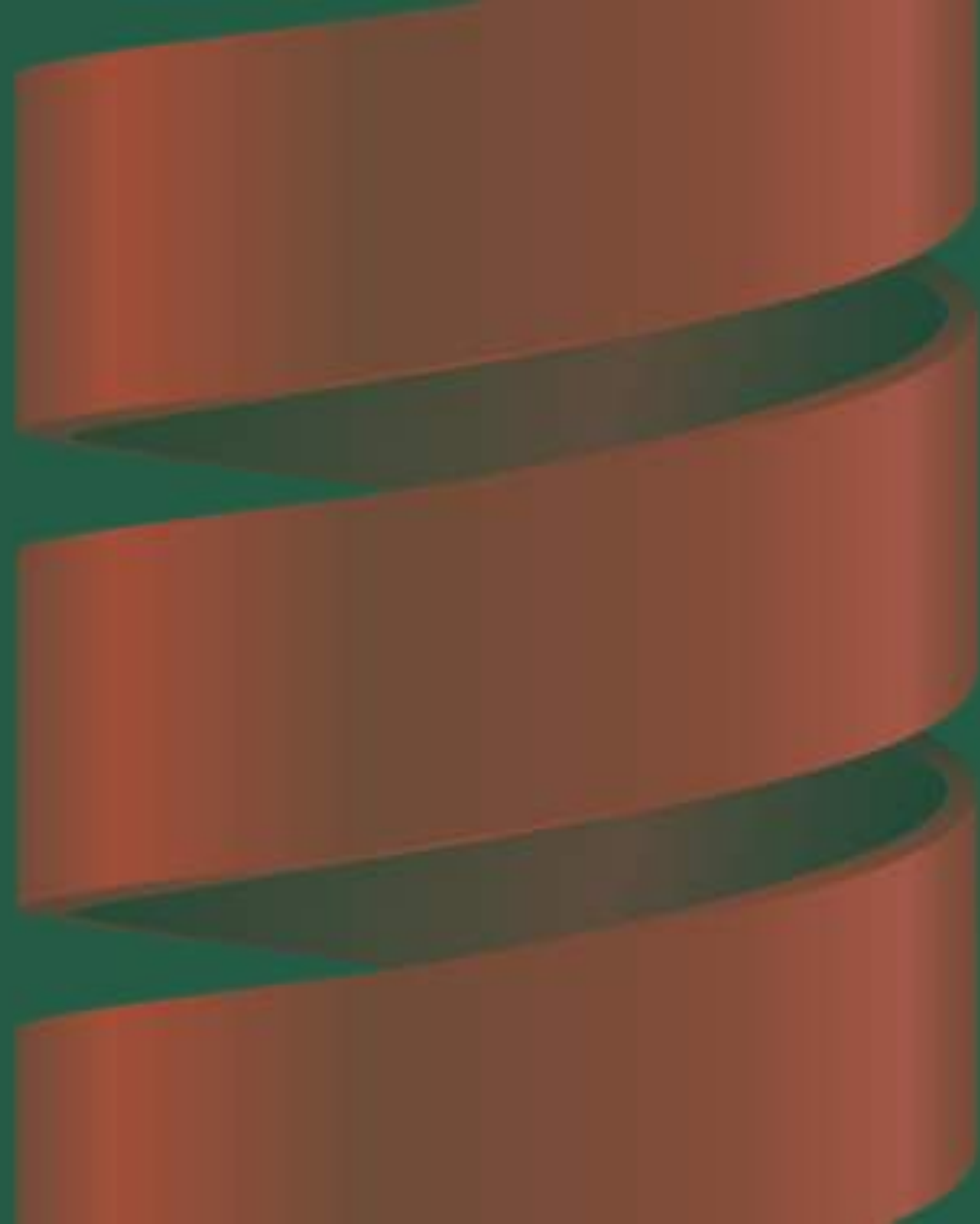


SOUNDCLOUD





- > 12 HOURS UPLOADED EVERY MINUTE
- > ~35K LISTENING YEARS EVERY MONTH
- > >135M TRACKS (INCLUDING CONTENT FROM MAJORS: SONY/UNIVERSAL/WARNER)
 - > ~180M MONTHLY ACTIVE USERS





CASE STUDY:

TYPE-LEVEL PROGRAMMING IN THE REAL WORLD

IN THE BEGINNING THERE WAS

```
def json(o: Any): Result
```



```
def typedJson[A : Writes](o: A): Result
```



```
implicit val writes = Json.writes[Track]
```



```
@ import play.api.libs.json.{Json => PJson}
import play.api.libs.json.{Json => PJson}
@ case class Omg(_1: Int, _2: Int, _3: Int, _4: Int, _5: Int,
_6: Int, _7: Int, _8: Int, _9: Int, _10: Int, _11: Int, _12: Int,
_13: Int, _14: Int, _15: Int, _16: Int, _17: Int, _18: Int,
_19: Int, _20: Int, _21: Int, _22: Int, _23: Int)
defined class Omg
@ PJson.writes[Omg]
cmd9.sc:1: No unapply or unapplySeq function found for class Omg.
val res9 = PJson.writes[Omg]
                        ^
```

Compilation Failed


```
import Json.writes._ // Json.writes.deriveInstance, implementation details  
implicit val writes = Json.writes[Track]
```

```
cmd9.sc:1: No unapply or unapplySeq function found for class Omg.  
val res9 = PJson.writes[Omg]  
                  ^
```

Compilation Failed

```
@ import com.soundcloud.json.Json
```

```
import com.soundcloud.json.Json
```

```
@ import Json.writes._
```

```
import Json.writes._
```

```
@ Json.writes[Omg]
```

```
res11: play.api.libs.json.Writes[Omg] =  
  play.api.libs.json.Writes$$anon$5@60ec44ee
```




HOW DOES THIS WORK EXACTLY? ଠ_ଠ



>700LOC OF MACRO



TWO (MANDATORY) BUILDING BLOCKS

- HLists
- Generic (TINY LIE: WHAT WE ACTUALLY NEED IS A
LabelledGeneric)
- [OPTIONAL] Coproducts

HLIST

```
@ import shapeless._  
import shapeless._
```

```
@ val hlist = 1l :: "hello" :: HNil  
hlist: Long :: String :: HNil = 1 :: hello :: HNil
```



```
@ hlist(0)
res7: Long = 1
```

```
@ hlist(1)
res8: String = hello
```

```
@ hlist(2)
<console>:16: error:
Implicit not found: Scary[Type].Please#Ignore
You requested to access an element at the position
TypelevelEncodingFor[2.type]
but the HList Long :: String :: HNil is too short.
```

```
      hlist(2)
            ^
```

```
Compilation failed.
```

GENERIC

```
@ case class Track(id: Long, payload: String)  
defined class Track
```

```
@ val generic = Generic[Track]  
generic: shapeless.Generic[Track]{type Repr = Long :: String :: HNil} =  
  anon$macro$3$1@7f8f5e52
```



```
@ val representation = generic.to(Track(1, "hello"))
representation: res0.Repr = 1 :: hello :: HNil
```

```
@ representation(0)
res10: Long = 1
```

```
@ representation(1)
res11: String = hello
```

```
@ representation(2)
<console>:19: error:
Implicit not found: Scary[Type].Please#Ignore
You requested to access an element at the position
TypelevelEncodingFor[2.type]
but the HList Long :: String :: HNil is too short.
      representation(2)
                     ^
```

```
@ generic.from(hlist)  
res7: Track = Track(1L, "hello")
```



```
Track(id: Long, payload: String)
```

(Long, String)

Long :: String ::

PUTTING THIS TOGETHER

```
object writes extends LabelledProductTypeClassCompanion[Writes] with DefaultWrites {
  object typeClass extends LabelledProductTypeClass[Writes] {
    override def emptyProduct: Writes[HNil] =
      Writes(_ => PlayJson.obj())

    override def product[H, T <: HList](name: String, headEv: Writes[H], tailEv: Writes[T]) =
      Writes[H :: T] {
        case head :: tail =>
          val h = headEv.writes(head)
          val t = tailEv.writes(tail)

          (h, t) match {
            case (JsNull, t: JsObject) => t
            case (h: JsValue, t: JsObject) => PlayJson.obj(name -> h) ++ t
            case _ => PlayJson.obj()
          }
        }
      }

    override def project[F, G](instance: => Writes[G], to: F => G, from: G => F) =
      Writes[F](f => instance.writes(to(f)))
  }
}
```



```
override def emptyProduct: Writes[HNil] =  
  Writes(_ => PlayJson.obj())
```

```
override def product[H, T <: HList](name: String,  
  headEv: Writes[H], tailEv: Writes[T]) =  
  Writes[H :: T] {  
    case head :: tail =>  
      val h = headEv.writes(head)  
      val t = tailEv.writes(tail)  
  
      (h, t) match {  
        // For Nones  
        case (JsNull, t: JsObject) => t  
        case (h, t: JsObject) =>  
          PlayJson.obj(name -> h) ++ t  
        case _ => PlayJson.obj()  
      }  
  }  
}
```

```
override def project[F, G](instance: => Writes[G],  
  to: F => G, from: G => F) =  
  Writes[F](f => instance.writes(to(f)))
```




THE WHOLE CODE



THREE DETAILS


```
import play.api.libs.json.DefaultWrites
```

```
... with DefaultWrites
```



```
@annotation.implicitAmbiguous(  
    "You have a Unit hiding somewhere in your types")  
implicit def noUnits: Writes[Unit] = null  
implicit def noUnitsBitte: Writes[Unit] = null
```

```
@ Json.writes[Unit]
```

```
cmd2.sc:1: You have a Unit hiding somewhere in your types
```

```
val res2 = Json.writes[Unit]
```

```
^
```




COPRODUC::+::S


```
@ {  
  sealed trait Playable  
  case class Track(id: Long, payload: String) extends Playable  
  case class Album(tracks: List[Track]) extends Playable  
}  
defined trait Playable  
defined class Track  
defined class Album  
  
@ Generic[Playable]  
res2: Generic[Playable]{type Repr = Album :+: Track :+: CNil} =  
  $sess.cmd2$anon$macro$1$1@2f4344c6
```

```
@ import com.soundcloud.json.Json
import com.soundcloud.json.Json
@ import play.api.libs.json.{Json => PJson}
import play.api.libs.json.{Json => PJson}
@ PJson.writes[Playable]
cmd5.sc:1: not found: type Writes
val res5 = PJson.writes[Playable]
                  ^
```

Compilation Failed

```
@ import Json.writes._
import Json.writes._
@ Json.writes[Playable]
res6: play.api.libs.json.Writes[Playable] =
  play.api.libs.json.Writes$$anon$5@c5ff6e1
```

```
@ PJson.toJson(Album(List(Track(1, "payload1"),  
                                Track(2, "payload2"))))  
res7: play.api.libs.json.JsonValue =  
  {"tracks":[{"id":1,"payload":"payload1"},  
             {"id":2,"payload":"payload2"}]}
```




The Type Astronaut's
Guide to Shapeless

Dave Gurnell
foreword by Miles Sabin

BY @DAVEGURNELL

BY @CVOGT

[HTTPS://GITHUB.COM/XDOTAI/PLAY-JSON-EXTENSIONS](https://github.com/XDOTAI/PLAY-JSON-EXTENSIONS)

A large white question mark is centered over a photograph of a marathon finish line at night. The background shows a dark running track, a blue and white inflatable archway, and spectators on the left. A banner for 'AIRPORT NIGHT RUN' is visible on the right, along with a sign that says 'ZIEL' (Goal) with an arrow pointing left. The scene is dimly lit, with some light reflecting off the track and the archway.

?

***QUESTIONS**