Haskell para hackers usando CodeWars

esLibre

Granada, viernes 21 de Junio
ETSIIT Universidad de Granada.

Reynaldo Cordero

- SSII de la Universidad de Alcalá https://www.uah.es/
- **HaskellMAD** https://www.meetup.com/Haskell-MAD/
- **Haskellnautas** https://haskellnautas.herokuapp.coms

Haskell para hackers usando CodeWars

Por qué estoy aquí

- Suficientemente viejo como para haber participado en **Hispalinux**
  - Coordinador en la lista **Software Libre en la Administración**
- Como @**norcoreano**, soy de los que se apuntan a un **bombardeo**
- Enamorado de **Haskell** (Programación **funcional**)

- Objetivo personal:
  - Enfrentar el
    - **problema más doloroso** entre los **desarrolladores** de software

Haskell para hackers usando CodeWars

Por qué estoy aquí

- Suficientemente viejo como para haber participado en **Hispalinux**
  - Coordinador en la lista **Software Libre en la Administración**
- Como @**norcoreano**, soy de los que se apuntan a un **bombardeo**
- Enamorado de **Haskell** (Programación **funcional**)

- Objetivo personal:
  - Enfrentar el
    - **problema más doloroso** entre los **desarrolladores** de software
      - **Hemos dejado a la mayoría de las mujeres fuera**

Haskell para hackers usando CodeWars

Mis dos minutos de reflexión: ¿Qué pasó y qué puedo hacer?

- ***Statu quo*** en los **lenguajes** (de **alto nivel** de **propósito general**)
  - Se aprende **mal** a programar (produce **exclusión**)
    - Las **herramientas** usuales podrían no ser las idóneas
      - Lenguajes: imperativo, OO, ... ¿Probamos con el **funcional**?

- ***Statu quo*** de la **documentación**
  - **RTFM**

Haskell para hackers usando CodeWars

Lenguajes de programación: la **complejidad accidental**

- Cada lenguaje tiene una **razón de ser** y un **objetivo**
  - **C**
  - **C++**
  - **Java**
  - **Brainfuck**

  - **Haskell**
    - Expertos, en gran medida relacionados con la **universidad**
    - El lenguaje crece a golpe de paper científico **matemático**
    - Compilador **GHC** con licencia **BSD**
    - **Emacs** es el editor preferido a día de hoy
    - Sus **objetivos** son:
      - **Mantenibilidad**
      - **Legibilidad (Comprensión del código)**
      - **Reducir o eliminar los errores** (detección **antes** de ejecutar)

Haskell para hackers usando CodeWars

Lenguajes de programación: la **complejidad accidental**

- Cada lenguaje tiene una **razón de ser** y un **objetivo**
  - **C**
  - **C++**
  - **Java**
  - **Brainfuck**

  - **Haskell**
    - Expertos, en gran medida relacionados con la **universidad**
    - El lenguaje crece a golpe de paper científico **matemático**
    - Compilador **GHC** con licencia **BSD**
    - **Emacs** es el editor preferido a día de hoy
    - Sus **objetivos** son:
      - **Mantenibilidad**
      - **Legibilidad (Comprensión del código)**
      - **Reducir o eliminar los errores** (detección **antes** de ejecutar)
    - Es el **Debian** de los lenguajes

Haskell para hackers usando CodeWars

Lenguajes de programación: la **complejidad accidental**

- La **clave** de la programación funcional es la **simplicidad**
  - Todo son **expresiones** (no hay sentencias)
  - Todo gira sobre la **composición** de funciones
    - **facilidad de comprensión**
    - **facilidad de cambio**
    - **depuración más fácil**
    - **mayor flexibilidad**
    - **modularidad**

  - **Construcciones complejas:** **Estado, Objeto, Métodos, Sintaxis, Herencia, Interruptor / coincidencia, Vars, Bucles imperativos, Actores, ORM, Condicionales.**
  - **Construcciones simples:** **valores, funciones, espacios de nombres, datos, polimorfismo, refs administrados, funciones Set, colas, manipulación de datos declarativos, reglas, consistencia.**

  - Excelente explicación en la conferencia:
    - **Simple Made Easy** - **Rich Hickey**

Haskell para hackers usando CodeWars

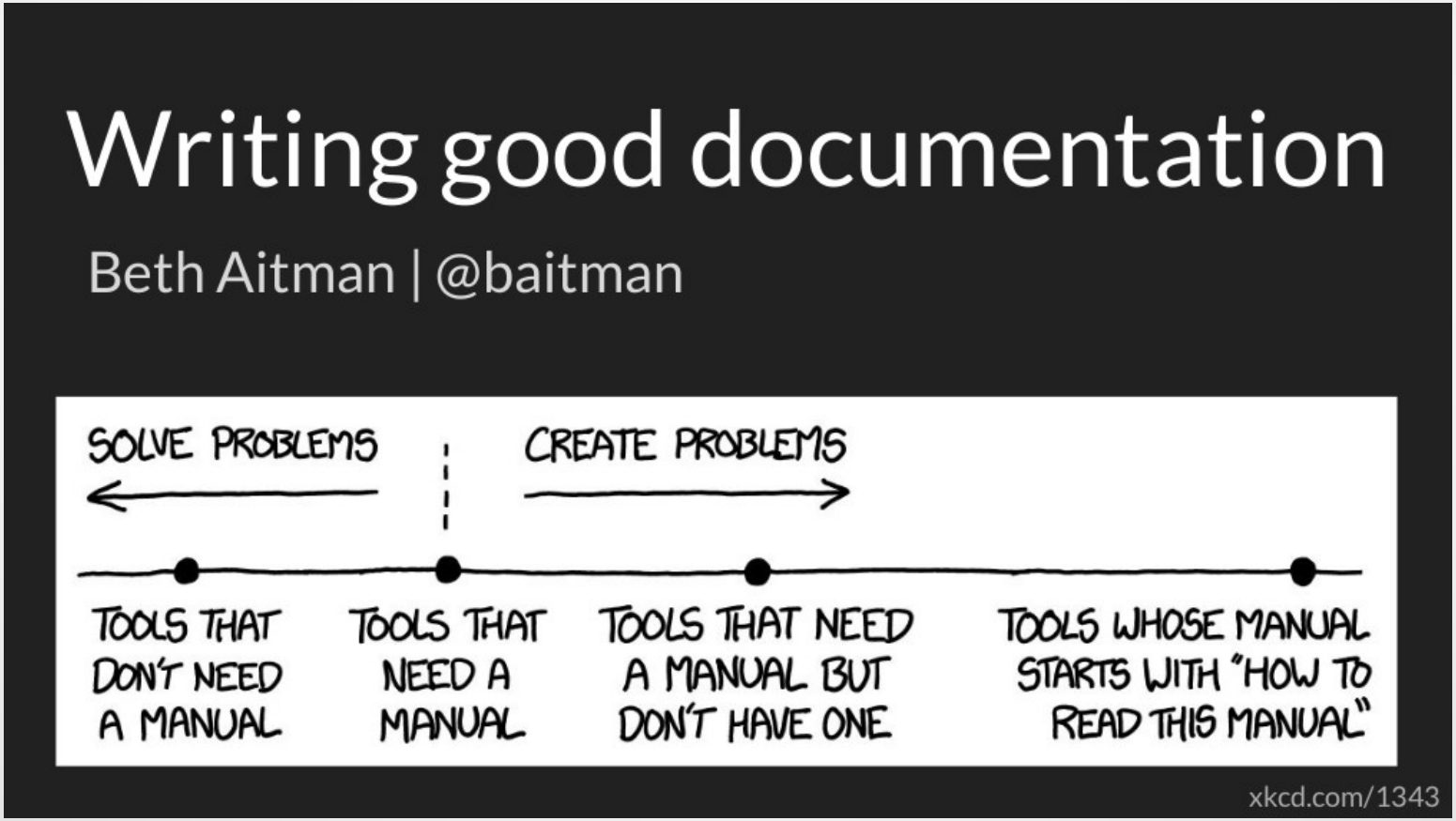
Lenguajes de programación: la **complejidad accidental**

- Elementos que determinan la **complejidad accidental**
  - **Boilerplate**
  - **Sintaxis**
  - **Acoplamiento**

Haskell para hackers usando CodeWars

Los **problemas** con la **documentación**

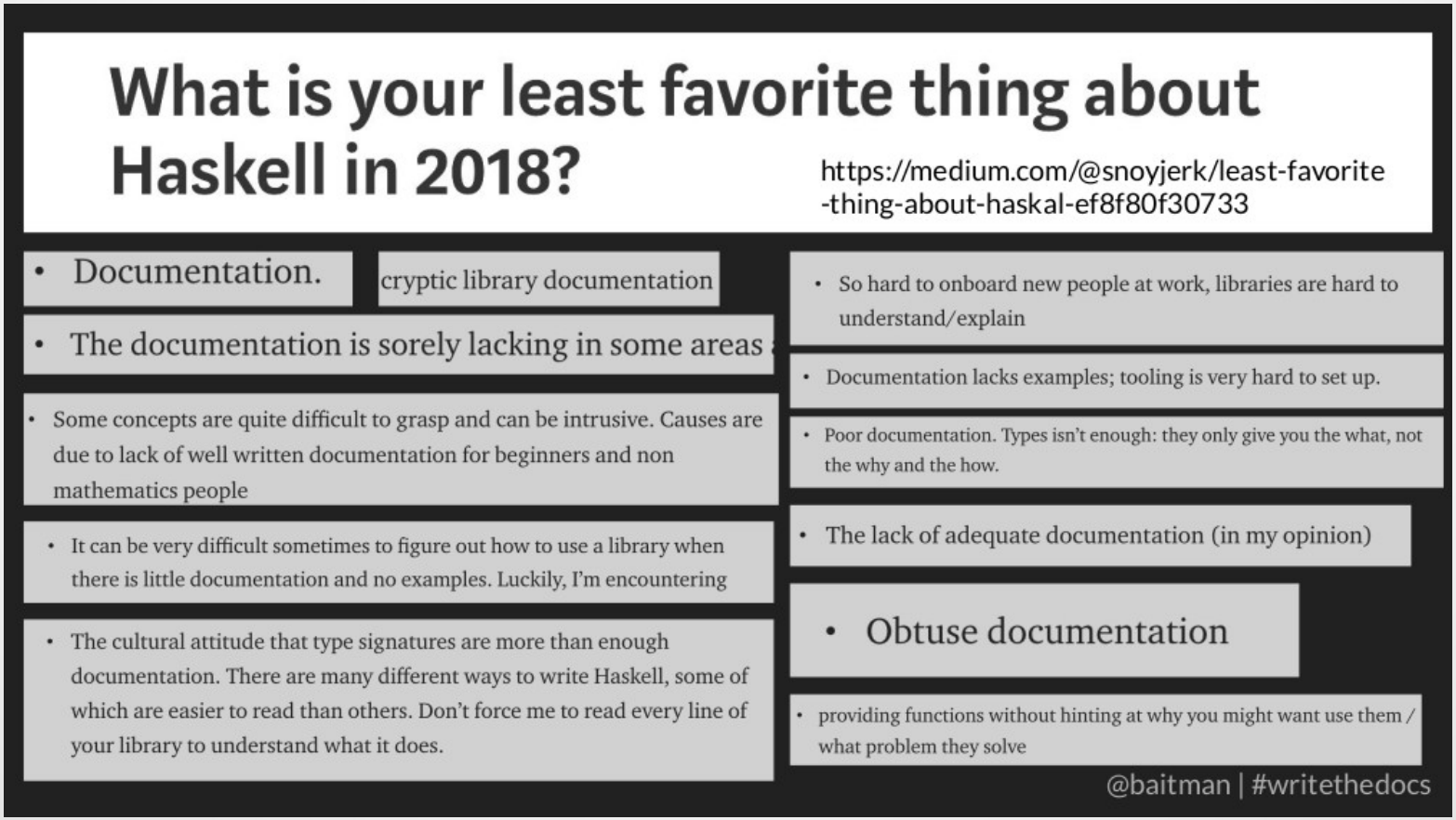
Recién presentado en el **ZuriHac 2019**



Haskell para hackers usando CodeWars

Los **problemas** con la **documentación**

Recién presentado en el **ZuriHac 2019**



Haskell para hackers usando CodeWars

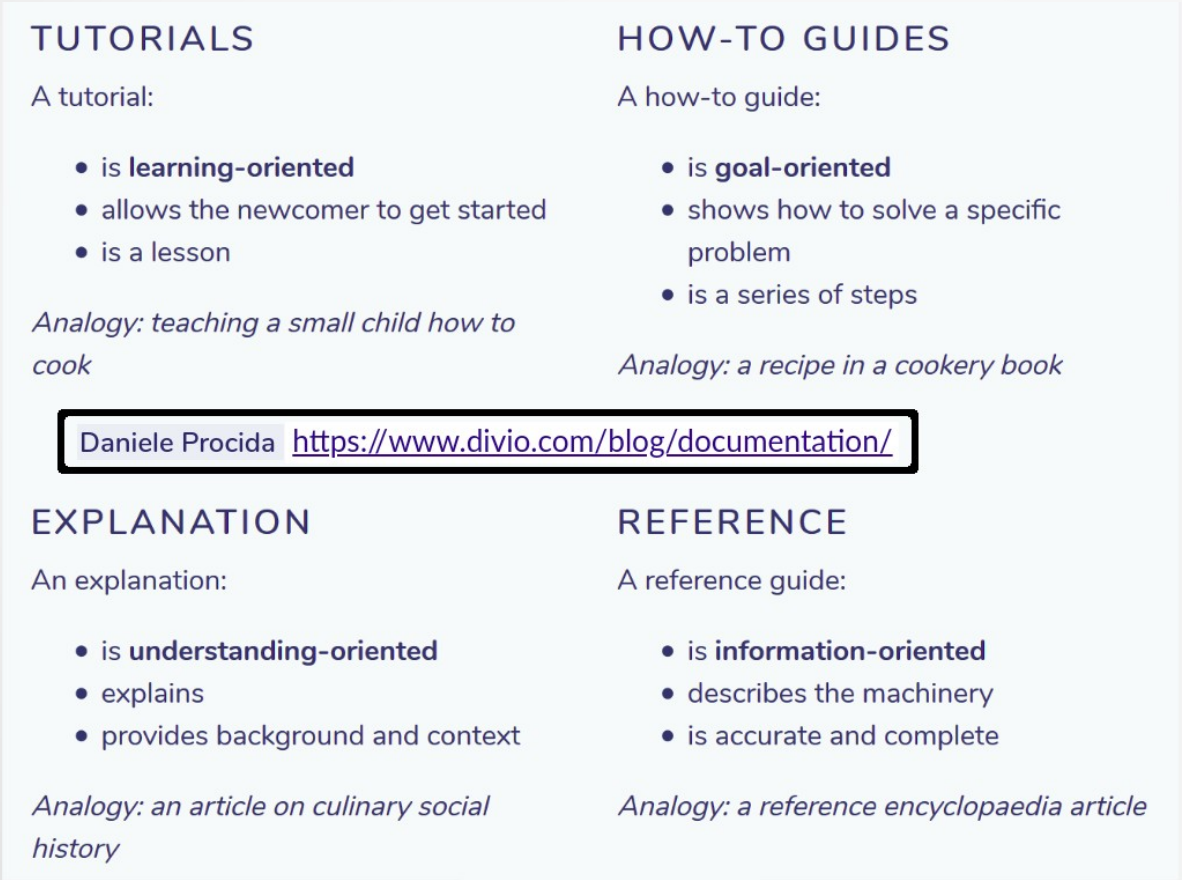
La Kata como método para superar el *statu quo*

- Nuevo enfoque: la **Kata** con el objetivo de
  - Superar la idea de alguien enseñando y alguien aprendiendo
  - Facilita descubrir y valorar cosas **por uno mismo**
  - La metodología es muy **simple** y hasta **obvia**:
    - Aprende a programar **de la misma forma** que se aprende a
      - **Tocar un instrumento** musical
      - Dominar un **arte marcial**
    - Los elementos que no deben faltar son
      - El **ciclo** de **intentar**, **evaluar** el resultado y **mejorarlo** “un poquito”
      - Poner este resultado en **común**, y obtener nuevas **ideas**

Haskell para hackers usando CodeWars

Los **problemas** con la **documentación**

No hay a día de hoy un encaje de las **Katas**



TUTORIALS

A tutorial:

- is **learning-oriented**
- allows the newcomer to get started
- is a lesson

Analogy: teaching a small child how to cook

HOW-TO GUIDES

A how-to guide:

- is **goal-oriented**
- shows how to solve a specific problem
- is a series of steps

Analogy: a recipe in a cookery book

Daniele Procida https://www.divio.com/blog/documentation/

EXPLANATION

An explanation:

- is **understanding-oriented**
- explains
- provides background and context

Analogy: an article on culinary social history

REFERENCE

A reference guide:

- is **information-oriented**
- describes the machinery
- is accurate and complete

Analogy: a reference encyclopaedia article

Haskell para hackers usando CodeWars

La Kata y sus fases (prueba de concepto)

En **CodeWars**

1) **Plantear** y **resumir** el problema → Copiar en un **block de notas**
2) **Desarrollar** los **ejemplos** a mano (**Añadir** alguno)
3) Escribir los **test** (**añadir** alguno)
4) Verificar que **compila** (En metodología TDD: primer **rojo**)

Replicar en **local**

1) **Crear** nueva kata mediante **script** (**newh-test**)
   1) Como entrada, el **nombre** de la kata (**stack** y **cabal** lo procesan)
   2) Se debe obtener
      1) El **fichero** de los **tests**
      2) El **fichero** para el código de la **solución**
      3) Se ejecuta el primer **commit** de git con el boilerplate inicial
      4) Se cargan en el **editor** los dos ficheros
      5) Abrir un intérprete interactivo o REPL (**ghci**)
2) **Copypaste** de los **test** desde CodeWars (y prueba de primer **rojo**)
3) **Implementar** una primera solución
4) **Validar** la solución (pasar los test)

Haskell para hackers usando CodeWars

La Kata y sus fases (prueba de concepto)

(Seguimos en **local**)

5) Hacer **commit** en la **rama** “primera solución” (OK00)
6) **Reformatear** el código buscando la **legibilidad** y **expresividad**
    1) **Expandir** hacia abajo **partiendo** las líneas
    2) **Alinear** elementos comunes entre las lineas
    3) Buscar mejores **nombres** para las **variables** y **funciones**
        1) En Haskell el uso de **let** y de **where** ayuda
        2) Usar de **lambdas** cuando convenga por contexto
7) **Validar** la solución (pasar los test)
8) Hacer **commit**, actualizando OK00

En **CodeWars**

1) **Copypaste** de la **solución** desde **local** hacia **CodeWars**
2) **Validar** la solución (ya la tenemos probada de antemano…)
3) Ahora te permite acceder a las **soluciones** de **otros usuarios**
4) Trabajar las **soluciones** más interesantes en **local**
    1) **Aprender** de los compañeros y descubrir **nueva información**
    2) Hacer **commit** en sucesivas ramas OK01, OK02, ...

Haskell para hackers usando CodeWars

Referencias

- **Beth Aitman** - ZuriHac 2019 docs talk
  - https://zfoh.ch/zurihac2019/#docs-track
- **Rich Hickey** - Simple Made Easy
  - https://github.com/matthiasn/talk-transcripts/blob/master/Hickey_Rich/SimpleMadeEasy.md
- **Daniele Procida** - What nobody tells you about documentation
  - https://www.divio.com/blog/documentation/