BASTA!®

Share Connect 2010
Die Konferenz für SharePoint Development & Management

SQLCON 2010

Oliver Sturm | www.oliversturm.com

Oliver Sturm

Functional Magic

Microsoft MVP Most Valuable Professional

ineta

thinktecture

Oliver Sturm (@olivers)

Oliver Sturm

thinktecture

- **Consultant and Trainer**
- **Associate Consultant at thinktecture**

- **.NET Application Architecture**
  - User Interfaces
  - Data Handling / Data Access Architectures
  - Programming Languages
  - DevExpress Component/Framework Products

- **Microsoft MVP for C#**
- **INETA Europe Speaker**

- **Services: http://www.oliversturm.com**
- **Blog: http://www.sturmnet.org/blog**

- **oliver@oliversturm.com**

Agenda

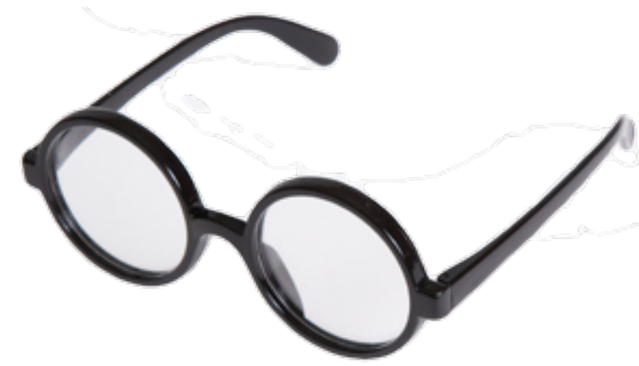
- Where are we with .NET and FP?
- The Magic of Functional Programming
- Erlang
- Haskell

Hybrid World .NET

- Object Oriented system, primarily
- Functional elements creeping in
- F# now a “standard” language
- Parallelization makes FP interesting

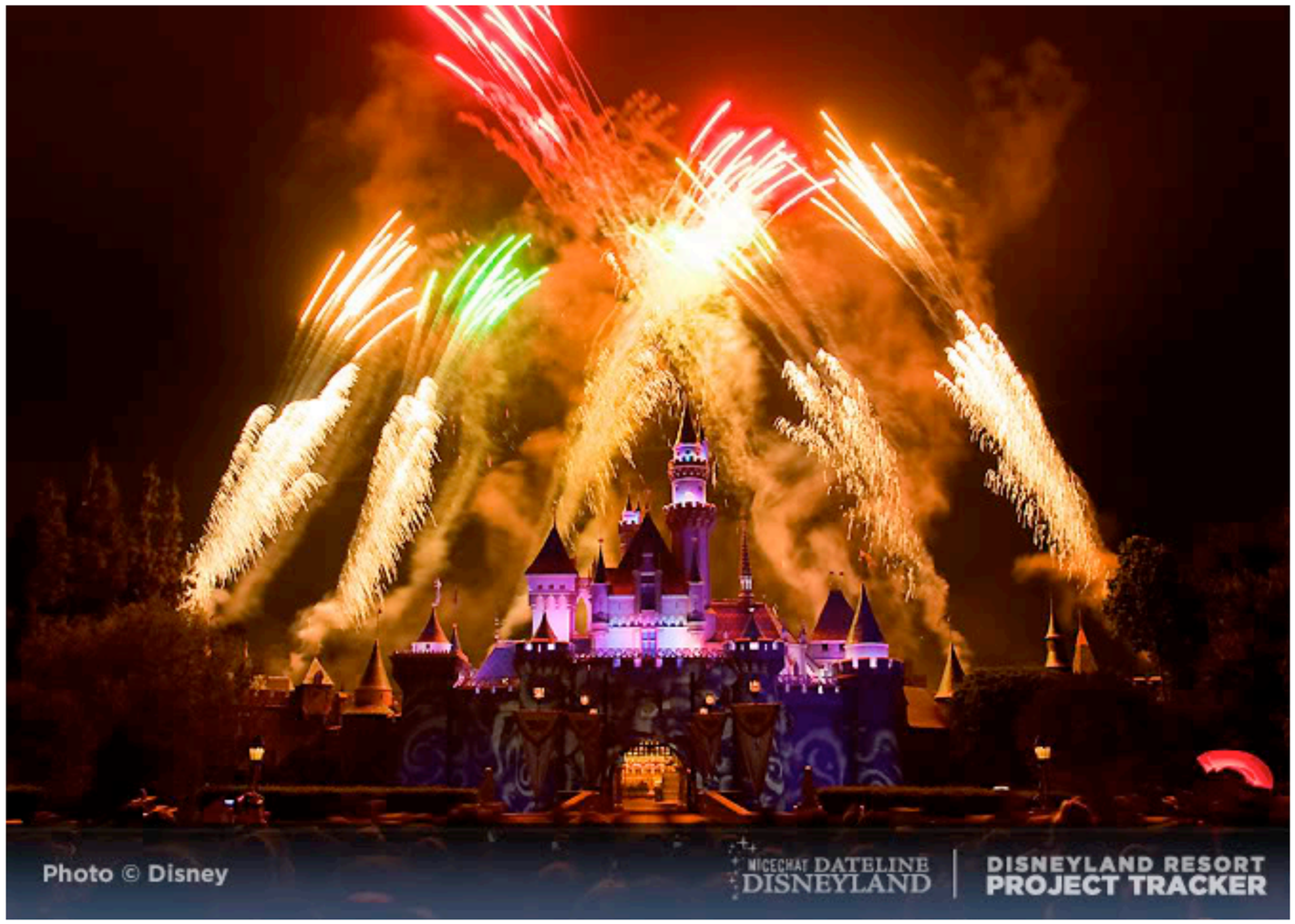
Functional Programming is Magic

- I never understood it, I might be wearing the wrong glasses!



- I never have to rebuild my app to fix bugs!
- Everything runs in parallel automatically!
- The system will run forever!
- My types will be so cool!
- Algorithms will write themselves!

Hermione Granger can do it



- … but we can’t, really
- Meanwhile, back in the real world…

Two functional languages

- Erlang
  - Functional Language
  - Dynamic Typing
  - Hot Swapping
  - Concurrency / Message Passing
  - Facebook uses it in their Chat app, also delicious, CouchDB, Amazon SimpleDB, ...

Two functional languages

- Haskell
  - Purely functional language
  - Non-strict evaluation
  - Interesting (strong!) type system
  - Monads...
  - Haskell in the industry: http://bit.ly/haskellindustry

Lots of stuff we can learn

- Simon Peyton Jones works for Microsoft :-)
- (Partly?) Functional APIs in .NET: LINQ, Reactive Extensions, Parallel Extensions
- Functional language elements in C#: lambda expressions, closures, iterators, expression trees, …
- F#

But some things are hard to imagine...

- Erlang hot-swapping code
- Erlang's actor model for concurrency - message passing
- Being really lazy with Haskell
- Type Classes
- Software Transactional Memory
- Concise data and algorithms

Erlang: Hot-swapping code

- Fix a problem in code
- Rebuild, but don't restart
- OTP patterns exist
- ASP.NET recycling AppDomains is a mechanism that looks similar (but isn't really very close)

Demo

Erlang: Hot-swapping code

Erlang Concurrency

- Actor Model
- "Shared-nothing asynchronous message passing"
- Very cheap "processes"
- No shared state between processes!
  No locking!
- A bit like Async Pattern in .NET, a very decoupled SOA idea, …

Demo

Erlang: Concurrency

Haskell: a non-strict language

- Often summarized as “lazy”
- Call-by-need is the “evaluation strategy” used by Haskell
- Expressions only get evaluated when the result is actually needed, and then memoized
- In many strict languages you find non-strict elements for short-circuiting

Demo

Being lazy with Haskell

Type Classes (Haskell)

- Type Classes define classes of types
- No, really!
- A bit like interfaces, but can contain implementation code
- A bit like abstract base classes, but are external to the type
- Haskell's type inference is aware of Type Classes

Demo

Type Classes

Haskell Concurrency and STM

- Some support for “automatic” parallelization - declarative parallelism
- Manual threading easy enough
- Data exchange through MVars leaves the locking problem
- Software Transactional Memory is elegant and safe through Monads

Demo

Haskell Concurrency and STM

Data and Algorithms in Haskell

- Algebraic datatypes / Discriminated Unions / Tagged Unions
- F# supports Discriminated Unions
- Algorithms - pattern matching, tail recursion, lazy eval...

Demo

Data and Algorithms in Haskell

RedBlackSet implementation from Chris Okasaki: Purely Functional Data Structures

Thank you

Please feel free to contact me about the content anytime.

oliver@oliversturm.com