

# Scala (Programmiersprache)

## Ein Teaser und allgemeinere Gedanken

---

Sebastian Eidecker

16. März 2016

2016-03-11

Scala (Programmiersprache)

Scala (Programmiersprache)

Ein Teaser und allgemeinere Gedanken

---

Sebastian Eidecker  
16. März 2016

*Wer als Werkzeug nur einen Hammer hat,  
sieht in jedem Problem einen Nagel.*

— Paul Watzlawick

2016-03-11

Scala (Programmiersprache)

*Wer als Werkzeug nur einen Hammer hat,  
sieht in jedem Problem einen Nagel.*

— Paul Watzlawick

Ich möchte heute über Scala reden. Eigentlich will ich aber nicht nur über Scala reden. Ich möchte nicht Scala verkaufen, sondern etwas völlig anderes. Scala ist spannend - für Nerds, die des Geldes wegen Java benutzen. Wir müssen aber auch über wichtigere Probleme reden. Und vielleicht passt Scala ja doch irgendwie.

In der IT wollen wir Probleme lösen und Möglichkeiten schaffen. Ich bin der festen Überzeugung, dass wir zu selten über den Kern unserer Probleme nachdenken und auch, dass sich diese Probleme gerade ändern und immer schneller ändern werden.

Ich glaube daher, dass wir unser Problem **und** unseren Werkzeugkasten kennen müssen. Ich glaube, dass wir zumindest wissen müssen, was im Baumarkt ausliegt und wir bei Bedarf einkaufen können. Daher kann es aus meiner Sicht nie schaden, sich neue Werkzeuge und Arbeitsweisen anzuschauen, damit man sie bei Bedarf zumindest im Hinterkopf hat. Sonst kann es passieren, dass wir das Problem gar nicht verstehen, weil wir kein passendes Werkzeug dafür besitzen und verwenden können.

# Worüber reden wir?

## IT im Wandel

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?  
IT im Wandel

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# Worüber reden wir?

## IT im Wandel

### Herausforderungen

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?  
IT im Wandel  
Herausforderungen

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# Worüber reden wir?

## IT im Wandel

### Herausforderungen

### Manifeste

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?  
IT im Wandel  
Herausforderungen  
Manifeste

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# Worüber reden wir?

## IT im Wandel

### Herausforderungen

### Manifeste

## Scala

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?  
IT im Wandel  
Herausforderungen  
Manifeste  
Scala

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# Worüber reden wir?

## IT im Wandel

### Herausforderungen

### Manifeste

## Scala

### Management Summary

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?  
IT im Wandel  
Herausforderungen  
Manifeste  
Scala  
Management Summary

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# Worüber reden wir?

## IT im Wandel

### Herausforderungen

### Manifeste

## Scala

### Management Summary

### Ein wenig Code

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?

IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

Ein wenig Code

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.



# Worüber reden wir?

## IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

Ein wenig Code

Spannendes

2016-03-11

Scala (Programmiersprache)

Worüber reden wir?

IT im Wandel

Herausforderungen

Manifeste

Scala

Management Summary

Ein wenig Code

Spannendes

Deswegen, möchte ich über zwei Dinge reden: Neue Herausforderungen und Scala.

# IT im Wandel

---

## Herausforderungen

2016-03-11

Scala (Programmiersprache)  
└─ IT im Wandel  
    └─ Herausforderungen

IT im Wandel  

---

Herausforderungen

# Software Engineering

2016-03-11

Scala (Programmiersprache)  
└─ IT im Wandel  
    └─ Herausforderungen

Software Engineering

# Software Engineering

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel

└ Herausforderungen

Software Engineering

Software Engineering. Passt der Begriff?

# Forderungen an IT

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel

└ Herausforderungen

Forderungen an IT

# Forderungen an IT

- Stabilität und Resilienz

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel

└ Herausforderungen

Forderungen an IT

- Stabilität und Resilienz

Magnor: Sorgen sie durch ihre Arbeit dafür, dass Systeme Kriterien der Stabilität und Resilienz erfüllen. Seien sie

„dran“ an den Problemen, antizipieren sie sie, schaffen sie Fall-Back-Lösungen – auch an Wochenenden.

# Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel

└ Herausforderungen

Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag

IT entwickelt sich zunehmend zu einem eigenen Produktionsfaktor (4-Sektor-Hypothese). Seien sie ein

verlässlicher Partner für das Business, nehmen sie die Business-Seite auch an die Hand, helfen und unterstützen

sie, leisten sie einen wahrgenommenen Wertbeitrag[...]

# Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag
- Businessstreiber

2016-03-11

Scala (Programmiersprache)  
└─ IT im Wandel  
    └─ Herausforderungen

## Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag
- Businessstreiber

Die Entwicklung der IT treibt zunehmend die Business-Seite und verändert Geschäftsmodelle. Seien sie an der

Front und treiben sie das Business, entwickeln sie neue Ideen, forschen sie nach neuen Lösungen (Beispiele: Uber,

3D-Printing, CoLo21, ...



# Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag
- Businessstreiber

— Matthias Magnor – CEO Surface und Contract Logistics

2016-03-11

Scala (Programmiersprache)  
└─ IT im Wandel  
└─ Herausforderungen

## Forderungen an IT

- Stabilität und Resilienz
- Wertbeitrag
- Businessstreiber

— Matthias Magnor – CEO Surface und Contract Logistics

Das Bewusstsein, dass ein Wandel stattfindet, ist im Business angekommen. Diese Forderungen stammen von

Matthias Magnor!

# IT im Wandel

---

## Manifeste

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel

└ Manifeste

IT im Wandel

Manifeste

Ein Manifest (lateinisch manifestus ‚handgreiflich gemacht‘) ist eine öffentliche Erklärung von Zielen und

Absichten, oftmals politischer Natur. (Wikipedia)

# Manifeste

- Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel  
└ Manifeste

Manifeste

- Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)

# Manifeste

- Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel  
└ Manifeste

Manifeste

- Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)

# Manifeste

- Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel  
└ Manifeste

Manifeste

- Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

# Manifeste

- Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)
- Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)
- Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel  
└ Manifeste

## Manifeste

- Antwortbereit, Widerstandsfähig, Elastisch, Nachrichtenorientiert (2013)
- Gut gefertigt, Stets Mehrwert, Gemeinschaft aus Experten, Produktive Partnerschaften (2009)
- Individuen und Interaktionen, Funktionierende Software, Zusammenarbeit mit dem Kunden, Reagieren auf Veränderung (2001)

Es gibt Manifeste von Softwareentwicklern, die sehr Ähnliches aussagen. (Reaktives Manifest, agiles Manifest,

Manifest der Software Craftmanship-Bewegung als Beispiele) Diese sind bekannt und – ich habe zumindest das

Gefühl – auch anerkannt. Ich will auch nur sehr kurz darauf eingehen, es soll ja vor allem um Scala gehen.

# Wo stehen wir?

2016-03-11

Scala (Programmiersprache)

└ IT im Wandel  
└ Manifeste

Wo stehen wir?

# Wo stehen wir **im Wettbewerb**?

2016-03-11

Scala (Programmiersprache)

- └ IT im Wandel
  - └ Manifeste

Wo stehen wir **im Wettbewerb**?

TODO: Was bedeutet das



# Scala

---

## Management Summary

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Scala

Management Summary

# Scalable Language

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Scalable Language

# Scalable Language

*This means that Scala grows with you. You can play with it by typing **one-line expressions** and observing the results. But you can also rely on it for **large mission critical systems** [...]*

— [www.scala-lang.org](http://www.scala-lang.org)

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
└─ Management Summary

## Scalable Language

*This means that Scala grows with you. You can play with it by typing **one-line expressions** and observing the results. But you can also rely on it for **large mission critical systems** [...]*

— [www.scala-lang.org](http://www.scala-lang.org)

TODO: Was bedeutet das? Scala ist sehr flexibel. Es vereint sehr viele Konzepte und ist im API-Design sehr stark. So

sollen unterschiedlichste Szenarien und Anforderungen bedient werden. Schauen wir, ob das Versprechen

gehalten wird.

# Eigenschaften

2016-03-11

- Scala (Programmiersprache)
  - Scala
  - Management Summary

Eigenschaften

# Eigenschaften

- Objektorientiert

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Eigenschaften  
• Objektorientiert

Keine Primitive, wie in Java. Ein schönes Beispiel: Dort hat man per Autoboxing Komplexität eingeführt statt

Schönheit. NullPointerException bei Primitiven anyone?

# Eigenschaften

- Objektorientiert
- Funktional

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional

# Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference

Aber Type Inference, wenn gewünscht

# Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default



# Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)

# Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)
- Ausdrucksstark (APIs/DSLs schreiben)

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)
- Ausdrucksstark (APIs/DSLs schreiben)

Bei Java bricht man sich schon die Hände, will man ein Builder-Pattern rekursiv umsetzen oder eine fluent API

# Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)
- Ausdrucksstark (APIs/DSLs schreiben)
- Jung (2004, Hype 2011)

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Eigenschaften

- Objektorientiert
- Funktional
- Statisch typisiert mit Type Inference
- Immutable by default
- Gewohnte Syntax („Java ohne Semikolon“)
- Ausdrucksstark (APIs/DSLs schreiben)
- Jung (2004, Hype 2011)

# Versprechen

2016-03-11

- Scala (Programmiersprache)
  - Scala
  - Management Summary

Versprechen

# Versprechen

- Produktivitätssteigerung

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Versprechen

- Produktivitätssteigerung

# Versprechen

- Produktivitätssteigerung
- Höhere Codequalität

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Versprechen

- Produktivitätssteigerung
- Höhere Codequalität

# Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß

# Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code



# Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code
- Höheres Abstraktionsniveau

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code
- Höheres Abstraktionsniveau

# Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code
- Höheres Abstraktionsniveau
- Skalierbarkeit

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

## Versprechen

- Produktivitätssteigerung
- Höhere Codequalität
- Mehr Spaß
- durch
- Weniger Code
- Höheres Abstraktionsniveau
- Skalierbarkeit

Passt in unseren Entwicklungsprozess. Leichte Änderungen an Deployment etc.

# Scala und die Java-Plattform

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Scala und die Java-Plattform

# Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Scala und die Java-Plattform  
• Java-Bytecode, läuft auf JVM

# Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
└─ Management Summary

Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar

# Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar
- Bekannte IDEs

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
└─ Management Summary

Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar
- Bekannte IDEs

# Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar
- Bekannte IDEs
- Ähnlicher Paketierungs- und Buildprozess (sbt)

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Management Summary

Scala und die Java-Plattform

- Java-Bytecode, läuft auf JVM
- Java-Bibliotheken nutzbar
- Bekannte IDEs
- Ähnlicher Paketierungs- und Buildprozess (sbt)

# Scala

---

## Ein wenig Code

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Scala

---

Ein wenig Code

Alles geklaut von Wikipedia oder Heiko Seebergers Buch



# Eine Java-Klasse

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
   └─ Ein wenig Code

Eine Java-Klasse

# Eine Java-Klasse

```
1 public class Person {  
2     private final String firstName;  
3     private final String lastName;  
4     public Person(String firstName, String lastName) {  
5         this.firstName = firstName;  
6         this.lastName = lastName;  
7     }  
8     public String getFirstName() {  
9         return firstName;  
10    }  
11    public String getLastName() {  
12        return lastName;  
13    }
```

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Eine Java-Klasse

```
1 public class Person {  
2     private final String firstName;  
3     private final String lastName;  
4     public Person(String firstName, String lastName) {  
5         this.firstName = firstName;  
6         this.lastName = lastName;  
7     }  
8     public String getFirstName() {  
9         return firstName;  
10    }  
11    public String getLastName() {  
12        return lastName;  
13    }
```

2016-03-11

# Eine Java-Klasse

```
1  @Override
2  public boolean equals(Object o) {
3      if (this == o) return true;
4      if (o == null || getClass() != o.getClass()) return false;
5      Person person = (Person) o;
6      if (firstName != null ?
7          !firstName.equals(person.firstName) :
8          person.firstName != null) return false;
9      if (lastName != null ?
10         !lastName.equals(person.lastName) :
11         person.lastName != null) return false;
12     return true;
13 }
```

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Eine Java-Klasse

```
1  @Override
2  public boolean equals(Object o) {
3      if (this == o) return true;
4      if (o == null || getClass() != o.getClass()) return false;
5      Person person = (Person) o;
6      if (firstName != null ?
7          !firstName.equals(person.firstName) :
8          person.firstName != null) return false;
9      if (lastName != null ?
10         !lastName.equals(person.lastName) :
11         person.lastName != null) return false;
12     return true;
13 }
```

2016-03-11

# Eine Java-Klasse

```
1  @Override
2  public int hashCode() {
3      int result = firstName != null ? firstName.hashCode() : 0;
4      result =
5          31 * result + (lastName != null ? lastName.hashCode() :
6              0);
7      return result;
8  }
```

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Eine Java-Klasse

```
1  @Override
2  public int hashCode() {
3      int result = firstName != null ? firstName.hashCode() : 0;
4      result =
5          31 * result + (lastName != null ? lastName.hashCode() :
6              0);
7      return result;
8  }
```

2016-03-11

# Businesslogik?

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Businesslogik?

Wo ist die Businesslogik versteckt? Ich gebe zu, das wird nur ein Beispiel, aber es zeigt symptomatisch eine Schwäche von Java: Java ist leicht verständlich, Änderungen werden aber immer non-breaking eingeführt und fühlen sich oft deplatziert an. Scala hat den Vorteil, neuer zu sein und kennt die Schmerzen der Java-Entwickler. Daher sind viele Dinge eingebaut, die man sich schon lange wünscht oder Konzepte, die man eigentlich schon immer umsetzen wollte, wenn es denn nicht so lästig wäre. (Immutable Objects, Lambdas, Generics, checked Exceptions als Beispiel)

Ein besonders lästiges Nicht-Feature-Feature sind in meinen Augen JavaBeans, wie wir sie gerade gesehen haben.

Leicht verständlich, aber strikte Richtlinien durch Konvention. Und wehe, dein Framework mag die Namen Deiner

Getter und Setter nicht.

# Dasselbe in Scala

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Dasselbe in Scala

Ja, nur ein Beispiel, ein besonders krasses. Aber dennoch ein alltägliches.

# Dasselbe in Scala

```
1 case class Person(firstName:String, lastName:String)
```

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Dasselbe in Scala

```
case class Person(firstName:String, lastName:String)
```

Ja, nur ein Beispiel, ein besonders krasses. Aber dennoch ein alltägliches.

# Es wird funktional – Quicksort

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Es wird funktional – Quicksort

Man sieht: Ausdrucksstark, aber auch nicht mehr so straight forward wie Java (vor 8)



## Es wird funktional – Quicksort

```
1 def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = xs match {  
2   case Nil      => xs  
3   case y :: ys => ys partition (_ <= y) match {  
4     case (l1, l2) => quickSort(l1) ++ (y :: quickSort(l2))  
5   }  
6 }
```

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Es wird funktional – Quicksort

```
> def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = xs match {  
>   case Nil      => xs  
>   case y :: ys => ys partition (_ <= y) match {  
>     case (l1, l2) => quickSort(l1) ++ (y :: quickSort(l2))  
>   }  
> }
```

Man sieht: Ausdrucksstark, aber auch nicht mehr so straight forward wie Java (vor 8)

# Schnelldurchlauf Scala – **Endlich!**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Schnelldurchlauf Scala – **Endlich!**

Jetzt endlich ein Schnelldurchlauf durch einige grundlegende Features von Scala. Leider nur kurz, ich habe meine Zeit ja mit Management Summary und meiner politischen Agenda vertrödelt. Alle Beispiele sind natürlich geklaut.

Also hier nur ein paar Dinge zum Grundverständnis und ein paar nette Sachen, die Java-Entwickler interessieren dürften.

# val und var – Immutables

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

val und var – Immutables

i ist val per Default, also immutable. Letzter Ausdruck ist return

# val und var – Immutables

```
1 def addOne(i: Int): Int = { i += 1; i }
```

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

val und var – Immutables

```
· def addOne(i: Int): Int = { i += 1; i }
```

i ist val per Default, also immutable. Letzter Ausdruck ist return

# Funktionen – Benannte Parameter

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Funktionen – Benannte Parameter

# Funktionen – Benannte Parameter

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Funktionen – Benannte Parameter

1 **class**

# Klassen und Objekte

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Klassen und Objekte

# Klassen und Objekte

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Klassen und Objekte

1 **class**



# Listen

2016-03-11

- Scala (Programmiersprache)
  - Scala
    - Ein wenig Code

Listen

# Listen

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Listen

1 **class**

# Pattern Matching

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Pattern Matching

# Pattern Matching

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Pattern Matching

1 **class**

# Type Inference

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Type Inference

# Type Inference

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Type Inference

· **class**

# Alles hat einen Typ – aber kein return

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Alles hat einen Typ – aber kein return

# Alles hat einen Typ – aber kein return

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Alles hat einen Typ – aber kein return

**class**



# ??? – Mein heimlicher Star

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

??? – Mein heimlicher Star

## ??? – Mein heimlicher Star

```
1 def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = ???
```

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

??? – Mein heimlicher Star

```
1 def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = ???
```

## ??? – Mein heimlicher Star

```
1 def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = ???
```

Kompilierbar, aber nicht gefährlich.

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

??? – Mein heimlicher Star

```
1 def quickSort[A <% Ordered[A]](xs: List[A]): List[A] = ???
```

Kompilierbar, aber nicht gefährlich.

# Vererbung und Traits

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Vererbung und Traits

# Vererbung und Traits

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Vererbung und Traits

1 **class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

# class

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

```
1  class
```

# Funktionen funktional – Lambdas schön

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Funktionen funktional – Lambdas schön



# Funktionen funktional – Lambdas schön

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Funktionen funktional – Lambdas schön

1 **class**

# Flatmap that shit!

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Flatmap that shit!

# Flatmap that shit!

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Flatmap that shit!

**class**

# Arbeit mit Strings

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Arbeit mit Strings

# Arbeit mit Strings

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Arbeit mit Strings

1 **class**

# Tupel

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Tupel

# Tupel

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Tupel

· **class**

# Implicits

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Implicits



# Implicits

1

**class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

Implicits

└─ **class**

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Ein wenig Code

class

2016-03-11

- Scala (Programmiersprache)
  - Scala
    - Ein wenig Code

1 class

# Scala

---

## Spannendes

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Scala

Spannendes

# Akka

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Akka

# Akka

- Scalable real-time transaction processing

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Akka

- Scalable real-time transaction processing

# Akka

- Scalable real-time transaction processing
- Will die aktuellen Probleme lösen

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Akka

- Scalable real-time transaction processing
- Will die aktuellen Probleme lösen

# Akka

- Scalable real-time transaction processing
- Will die aktuellen Probleme lösen
- 

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Akka

- Scalable real-time transaction processing
- Will die aktuellen Probleme lösen
-



# ScalaTest

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

ScalaTest

# ScalaTest

- Fachlich verständliche Tests

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

ScalaTest

- Fachlich verständliche Tests

# ScalaTest

- Fachlich verständliche Tests
- Testdatengenerierung

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

ScalaTest

- Fachlich verständliche Tests
- Testdatengenerierung

# Scalatest – Beispiel

```
1 "Creating a Time" should {
2   "throw an IllegalArgumentException for hours less than 0 or
   greater equal 24" in {
3     forAll("hours") { (hours: Int) =>
4       whenever(hours < 0 || hours >= 24) {
5         an[IllegalArgumentException] should be thrownBy Time(
6           hours)
7       }
8     }
9   }
```

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Scalatest – Beispiel

```
1 "Creating a Time" should {
2   "throw an IllegalArgumentException for hours less than 0 or
   greater equal 24" in {
3     forAll("hours") { (hours: Int) =>
4       whenever(hours < 0 || hours >= 24) {
5         an[IllegalArgumentException] should be thrownBy Time(
6           hours)
7       }
8     }
9   }
```

2016-03-11

# Domain Specific Languages

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
    └─ Spannendes

Domain Specific Languages

# Domain Specific Languages

- 

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
   └─ Spannendes

Domain Specific Languages  
•

# Domain Specific Languages

- 
- 

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Domain Specific Languages

- 
-

# Domain Specific Languages

- 
- 
- 

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Domain Specific Languages

- 
- 
-



# Meine wenig qualifizierte Meinung

2016-03-11

Scala (Programmiersprache)  
└─ Scala  
   └─ Spannendes  
      └─ Meine wenig qualifizierte Meinung

Meine wenig qualifizierte Meinung

## Vorteile

2016-03-11

- Scala (Programmiersprache)
  - Scala
    - Spannendes
      - Meine wenig qualifizierte Meinung

Meine wenig qualifizierte Meinung

Vorteile

## Vorteile

- Für moderne Architekturen geeignet

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

## Nachteile

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

#### Nachteile



# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

## Nachteile

- Komplex

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

#### Nachteile

- Komplex

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

## Nachteile

- Komplex
- Zukunftssicher?

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

#### Nachteile

- Komplex
- Zukunftssicher?

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

## Nachteile

- Komplex
- Zukunftssicher?
- Zahl an Entwicklungssklaven

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

#### Nachteile

- Komplex
- Zukunftssicher?
- Zahl an Entwicklungssklaven

# Meine wenig qualifizierte Meinung

## Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

## Nachteile

- Komplex
- Zukunftssicher?
- Zahl an Entwicklungssklaven
- Binärkompatibilität nicht in alle Ewigkeit

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

└─ Meine wenig qualifizierte Meinung

### Meine wenig qualifizierte Meinung

#### Vorteile

- Für moderne Architekturen geeignet
- Verständlich funktional
- Java-Ökosystem
- Macht Spaß
- Statisch typisiert

#### Nachteile

- Komplex
- Zukunftssicher?
- Zahl an Entwicklungssklaven
- Binärkompatibilität nicht in alle Ewigkeit

*We've found that Scala has enabled us to  
deliver things faster with less code. It's  
reinvigorated the team.*

— Graham Tackley, Guardian

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

*We've found that Scala has enabled us to  
deliver things faster with less code. It's  
reinvigorated the team.*

— Graham Tackley, Guardian

Der Guardian ist der Inbegriff für den Technologiewandel in einer klassischen, behäbigen Branche. Lange, bevor

Springer wach geworden ist.

# Mehr für Nerds

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Mehr für Nerds

Wie gesagt habe ich wenig Ahnung von Scala, da ich es noch nicht produktiv eingesetzt habe. Ich lasse mich aber gerne zwingen, für Interessierte ein Hands on durchzuführen, bei dem wir ein paar Stunden mit Scala in einer echten IDE spielen, nicht auf Folien. Ich würde natürlich auch dafür aus Büchern klauen.

# Mehr für Nerds

- [Sprecht mich an](#)

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes

Mehr für Nerds

- [Sprecht mich an](#)

Wie gesagt habe ich wenig Ahnung von Scala, da ich es noch nicht produktiv eingesetzt habe. Ich lasse mich aber gerne zwingen, für Interessierte ein Hands on durchzuführen, bei dem wir ein paar Stunden mit Scala in einer echten IDE spielen, nicht auf Folien. Ich würde natürlich auch dafür aus Büchern klauen.

# Mehr für Nerds

- Sprecht mich an
- Hands on-Termin bei Interesse

2016-03-11

Scala (Programmiersprache)

└─ Scala  
└─ Spannendes

Mehr für Nerds

- Sprecht mich an
- Hands on-Termin bei Interesse

Wie gesagt habe ich wenig Ahnung von Scala, da ich es noch nicht produktiv eingesetzt habe. Ich lasse mich aber gerne zwingen, für Interessierte ein Hands on durchzuführen, bei dem wir ein paar Stunden mit Scala in einer echten IDE spielen, nicht auf Folien. Ich würde natürlich auch dafür aus Büchern klauen.



# Mehr für Nerds

- Sprecht mich an
- Hands on-Termin bei Interesse
- Heiko Seeberger: „Durchstarten mit Scala. Tutorial für Einsteiger (2. Aufl.)“

2016-03-11

Scala (Programmiersprache)

└─ Scala  
└─ Spannendes

Mehr für Nerds

- Sprecht mich an
- Hands on-Termin bei Interesse
- Heiko Seeberger: „Durchstarten mit Scala. Tutorial für Einsteiger (2. Aufl.)“

Wie gesagt habe ich wenig Ahnung von Scala, da ich es noch nicht produktiv eingesetzt habe. Ich lasse mich aber gerne zwingen, für Interessierte ein Hands on durchzuführen, bei dem wir ein paar Stunden mit Scala in einer echten IDE spielen, nicht auf Folien. Ich würde natürlich auch dafür aus Büchern klauen.

2016-03-11

Scala (Programmiersprache)

└─ Scala

└─ Spannendes