

Groovy 4 the Win?

FALK SIPPACH // EMBARC

JavaLand, Phantasialand Brühl

Mittwoch, 16.03.2022



Groovy 4 the Win?

Seit vielen Jahren ist Groovy eine der innovativsten Sprachen auf der Java-Plattform. Sourcecode sowohl dynamisch als auch statisch typisieren zu können, ermöglicht sowohl eine ungeahnte Flexibilität, aber auch hohe Compiler-Sicherheit, wo es angeraten ist. Die Laufzeit- als auch Compile-Time-Metaprogrammierung beschert so manchem Entwickler magische Momente. Groovy Code lässt sich zudem sehr einfach mit Java integrieren und ist trotz der Ähnlichkeit in der Syntax deutlich prägnanter, verständlicher sowie besser lesbar und somit auch wartbarer.

Aber Java hat aufgeholt und sich bei vielen Features von Groovy, Scala und Co. inspirieren lassen. Zudem gab es im Groovy-Umfeld politisch motivierte Rückschläge, dann den Wechsel zur Apache Software Foundation und mittlerweile dank Gradle und Co. schwindelerregend in die Höhe geschnellte Downloadzahlen. Nun steht mit Groovy 4 wieder ein Major-Update mit vielen interessanten Neuerungen in den Startlöchern.

In diesem Vortrag schauen wir, wo sich Java und Groovy angenähert haben, wo Groovy gegebenenfalls auch von neuen Strukturen und APIs auf der Java-Plattform profitiert und wo die allseits beliebte Skriptsprache aber weiterhin den gewissen Unterschied ausmacht. Falls Ihr Groovy bereits verwendet, bekommt ihr einen kompakten Überblick über die neue Features. Und falls ihr es noch nicht nutzt, dann solltet ihr erst recht in diesem Vortrag dabei sein!





1995

"[Dieses Foto](#)" von Unbekannter Autor ist lizenziert gemäß [CC BY-SA](#)



Java hat Overhead

Groovy ist leichtgewichtig

sehr ähnlich, kompatible Syntax

Ist Java eigentlich noch State of the Art?

**Vorreiter, Inspiration
für Java**

**Java hat mittlerweile
Groovy überholt**



Agenda



- 1 Was war nochmal Groovy?
- 2 Geschichte
- 3 Neue Features
- 4 Fazit



Agenda



1 Was war nochmal Groovy?

2 Geschichte

3 Neue Features

4 Fazit

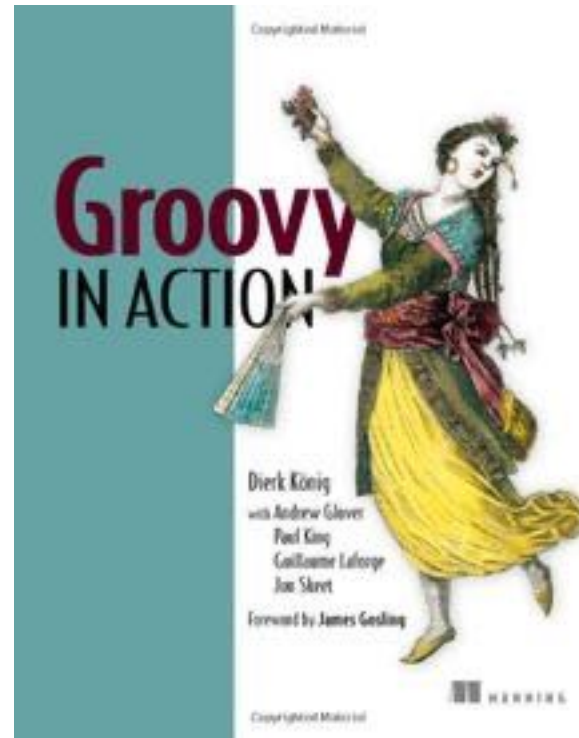
1



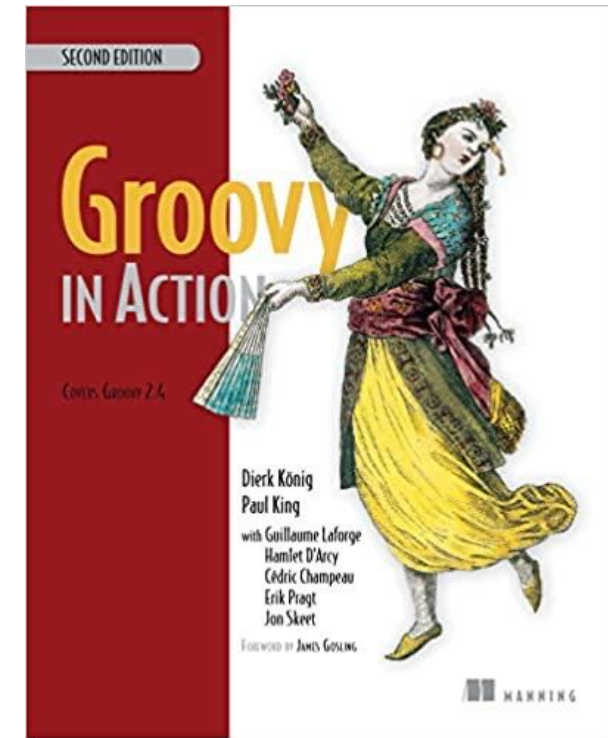
Groovy

- **vielschichtige General Purpose Programmiersprache**
- **Statische und dynamische Typisierung**
- **Erweiterbare Laufzeit- und Compile-Zeit-Metaprogrammierung**
- **Java-ähnliche Syntax und gute Integration**
- **Unterstützung für Domain Specific Languages**
- **Skripting**





2007



2015



=

ausdrucksstarke Syntax

+

mächtige Bibliotheken

+

Meta-Programmierung



Agenda



1 Was war nochmal Groovy?

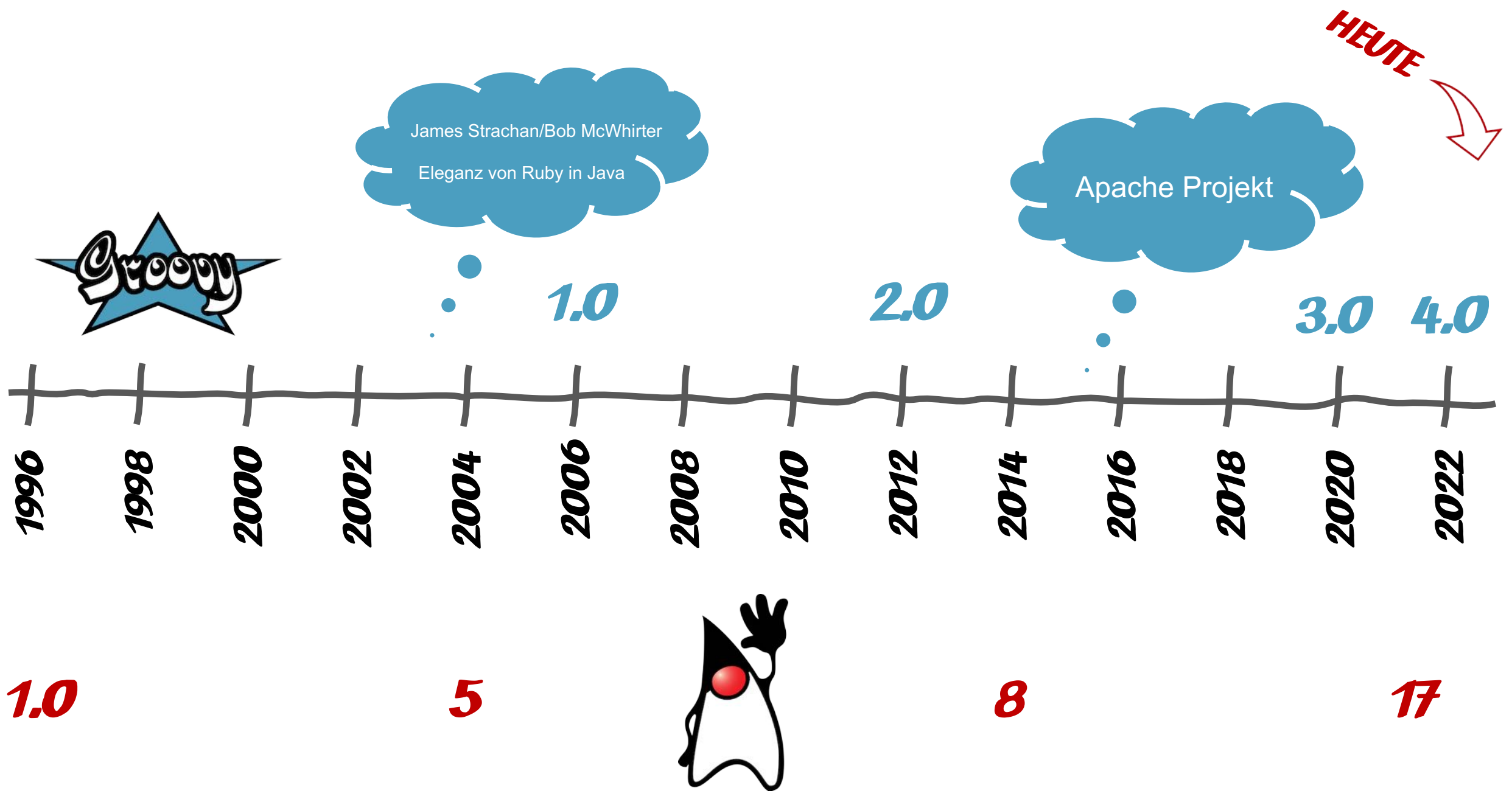
2 Geschichte

3 Neue Features

4 Fazit

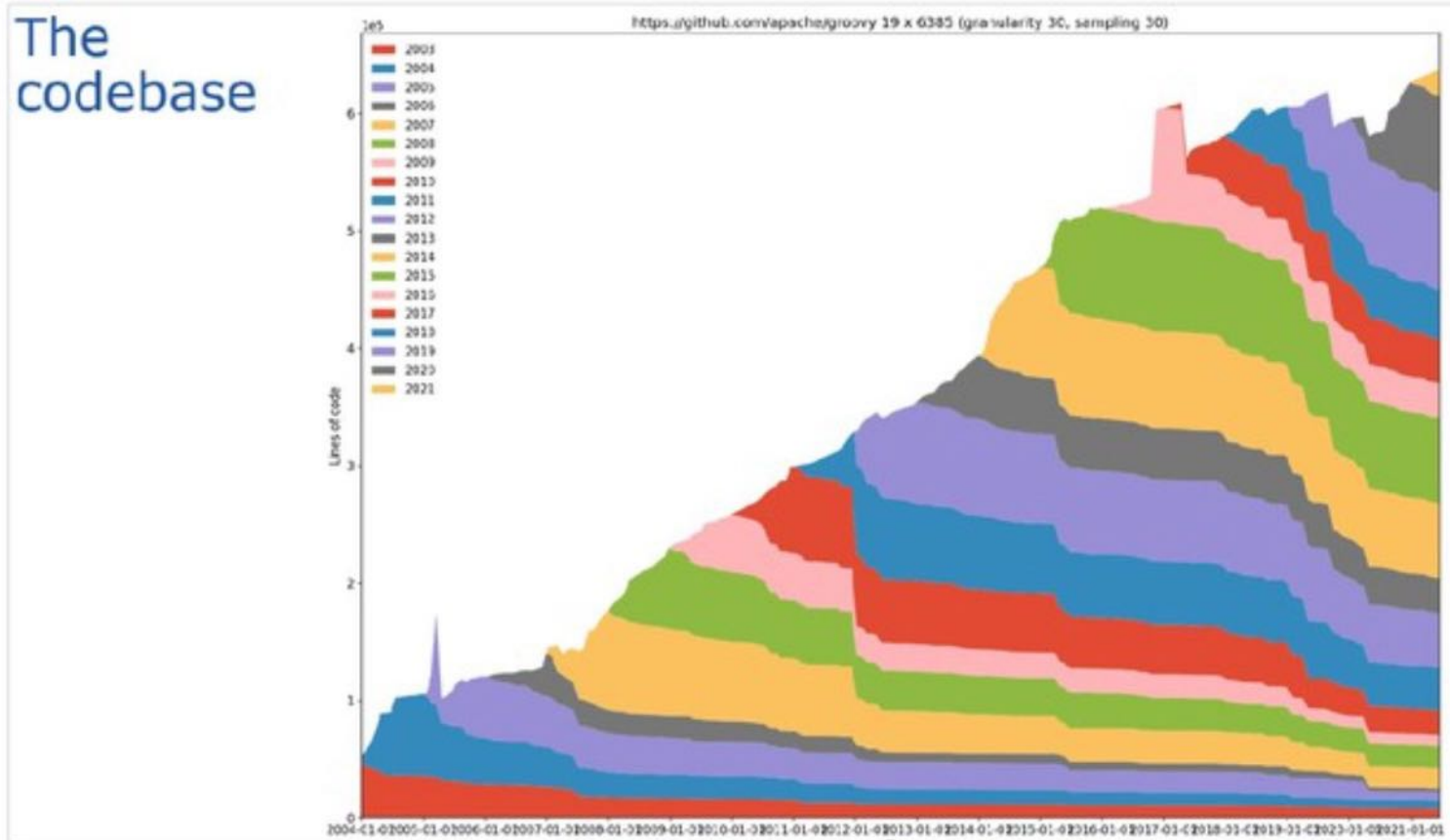
2





Groovy status in 2021: Healthy

- > 600K lines of source code
- > 19K commits
- > 8K issues & enhancements resolved
- > 500 contributors
- > 200 releases

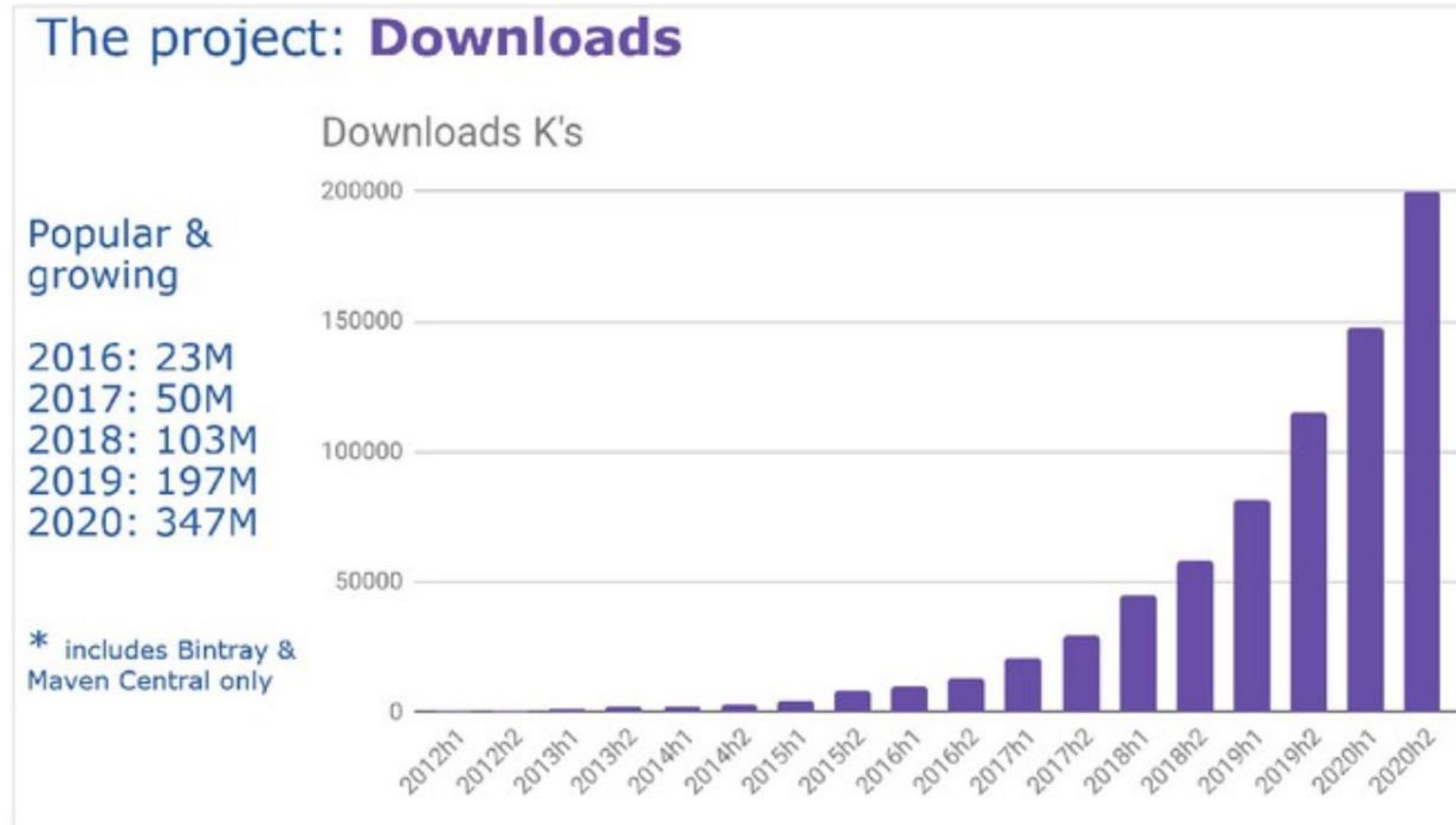


<https://speakerdeck.com/paulk/groovy-roadmap?slide=3>



Groovy – Downloads increasing

- > 1B downloads and growing



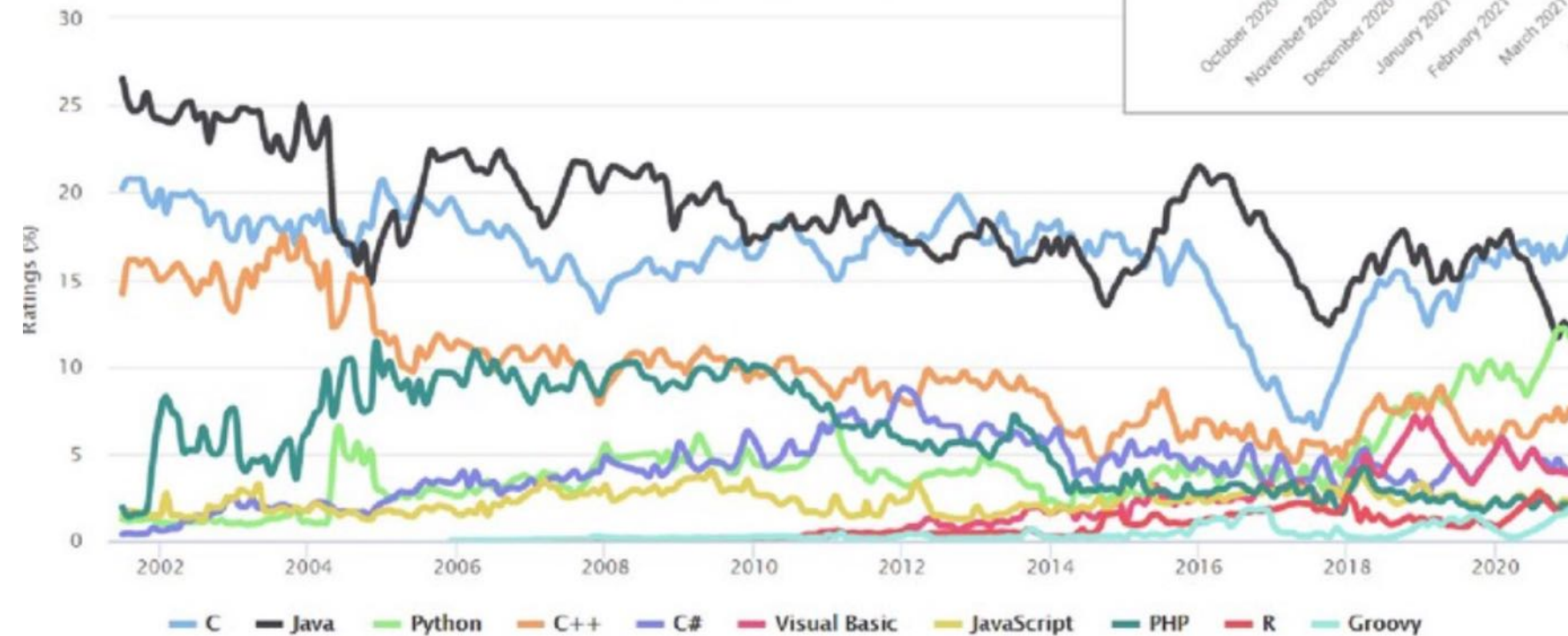
<https://speakerdeck.com/paulk/groovy-roadmap?slide=4>



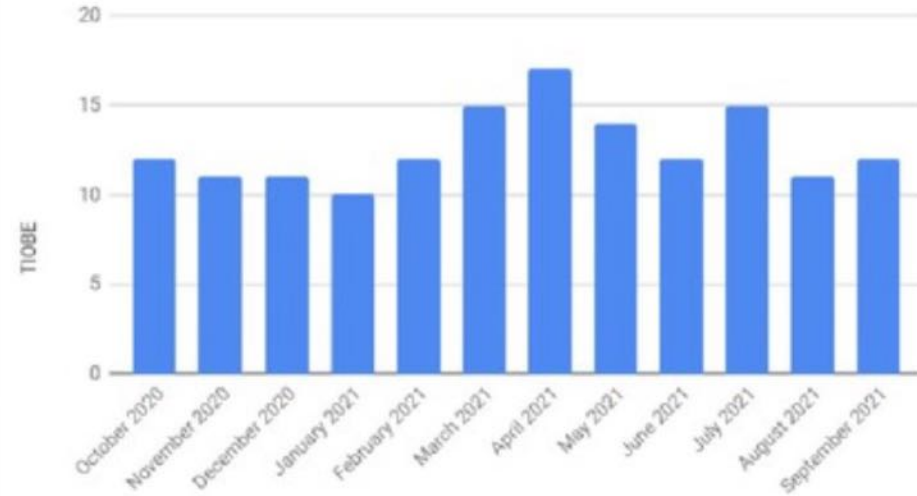
Groovy – Ranking steady

TIOBE Programming Community Index

Source: www.tiobe.com



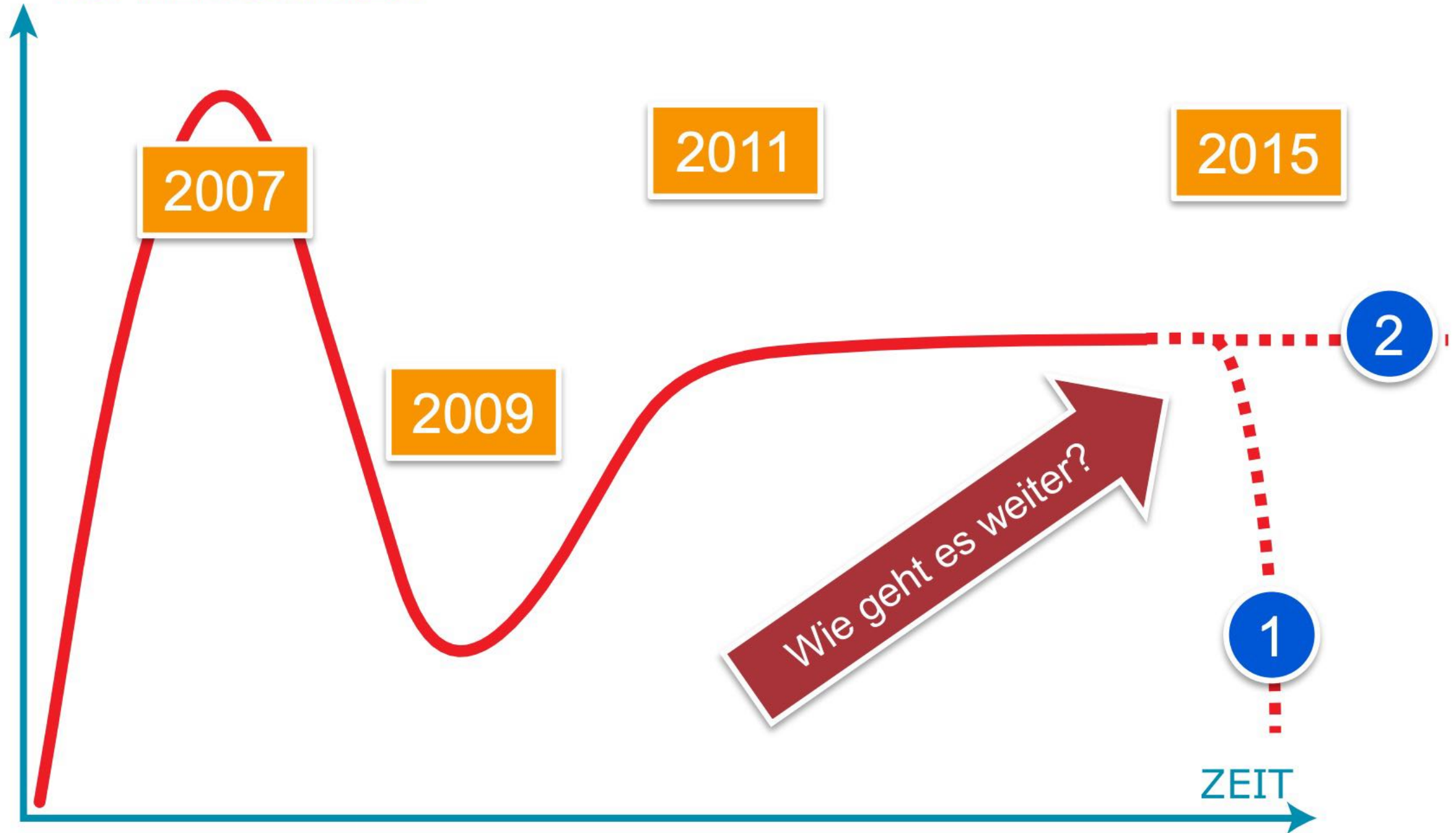
TIOBE index over the past 12 months



<https://speakerdeck.com/paulk/groovy-roadmap?slide=5>



AUFMERKSAMKEIT



Incubation

WARNING: Some features of Groovy 4 are designated as "incubating". Where appropriate, related classes or APIs of these features may be annotated with the `@Incubating` annotation.



Agenda



- 1 Was war nochmal Groovy?
- 2 Geschichte
- 3 Neue Features**
- 4 Fazit

3



Die Änderungen im Überblick

GINQ, a.k.a. Groovy-Integrated Query

Switch Expressions

Maven Koordinaten

Built-in type checkers

Records

Entfernung Legacy Package

Groovy Contracts

Built-in macro methods

TOML Support

Sealed Types

Moduländerungen

Enhanced Ranges

JavaShell

POJO Annotation



Maven Coordinates

org.codehaus.groovy => org.apache.groovy



Legacy package removal

~~groovy.util.XmlSlurper~~ => groovy.xml.XmlSlurper

groovy.xml.XmlParser

groovy.ant.AntBuilder

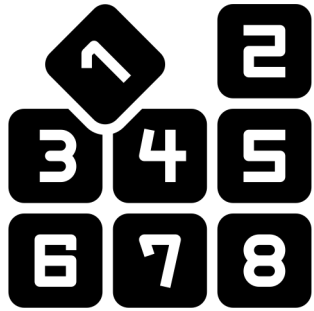
groovy.test.GroovyTestCase



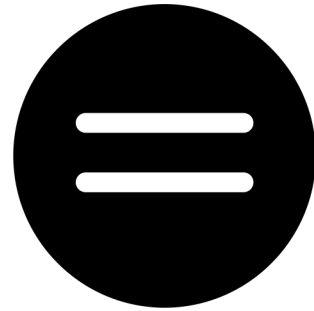
Java: Switch Statement

```
public String describeInt(int i) {  
    String str = "not set";  
    switch(i) {  
        case 1:  
        case 2:  
            str = "one or two";  
            break;  
        case 3:  
            str = "three";  
            break;  
    }  
    return str;  
}
```





Nur wenige Datentypen



Nur Gleichheit

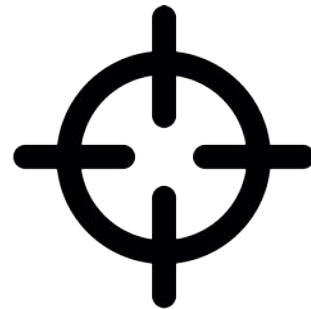


Stolperfalle Fallthrough



Fehlende Compiler-Sicherheit

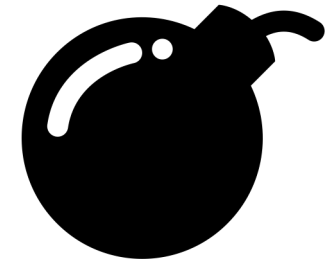
Probleme mit Switch Statements



Fehlender Scope



Unnötiger Boilerplate
Code



Nicht Null-Safe

Groovy: Switch Statement on Steroids

```
def x = new Date()
```

```
switch(x) {  
    case "Groovy" : println "in x steht 'Groovy'"  
        break  
    case Date : println "x ist ein Datum"  
        break  
    case 1..10 : println "x ist Zahl zw. 1 und 10"  
        break  
    default : println "x ist etwas anderes"  
}
```

flexibler:
case über Class,
Container, Reguläre
Ausdrücke, Closures
(Fallback ist equals)

Groovy: isCase() überschreiben

case in switch ruft isCase() auf und wertet Parameter als Boolean aus

```
// Closure prüft, ob Monat ein 'r' enthält  
def c = { Date date ->  
    new SimpleDateFormat('MMMM')  
        .format(date)  
        .contains('r')  
}  
  
c.isCase(new Date())
```

Muscheln essen in Monaten
die ein 'r' enthalten

Switch Expressions

```
print "Which language do you favor (1 for Groovy, 2 for Java): "  
def lang = switch(System.in.newReader().readLine() as Integer) {  
    case 1 -> "Groovy"  
    case 2 -> "Java"  
}  
println(lang)
```



Switch Expression: Unterschiede zu Java

If no default branch is present, an **implicit one returning null** is added.

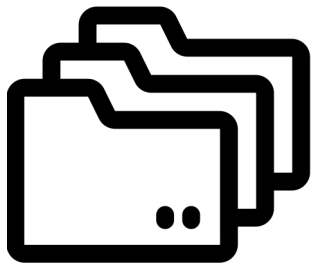
```
// default branch avoids GroovyCastException
int i = switch(s) {
    case 'one' -> 1
    case 'two' -> 2
    default -> 0
}
```

```
// default branch avoids NullPointerException
Optional.of(switch(i) {
    case 1 -> 'one'
    case 2 -> 'two'
    default -> 'buckle my shoe'
}))
```

Records in Java

```
record Point(int x, int y) {}  
record Rectangle(Point p1, Point p2) {  
    double area() { .. }  
}
```

transparente Tuple von Daten
immutable
minimalistisch/kompakt/prägnant



**Value Objects, DTOs,
JPA Projections**



Im Java Bytecode ...

```
package de.sippsack.records;
```

```
import java.math.BigDecimal;
```

```
import java.util.Currency;
```

```
public record MonetaryAmount(BigDecimal value, Currency currency) {}
```

→ records javap MonetaryAmount.class

Compiled from "MonetaryAmount.java"

```
public final class de.sippsack.records.MonetaryAmount extends java.lang.Record {  
    public de.sippsack.records.MonetaryAmount(java.math.BigDecimal, java.util.Currency);  
    public final java.lang.String toString();  
    public final int hashCode();  
    public final boolean equals(java.lang.Object);  
    public java.math.BigDecimal value();  
    public java.util.Currency currency();  
}
```



Records

Records in Groovy



```
record Point(int x, int y, Color color) {  
}
```

```
@groovy.transform.RecordType  
class MonetaryAmount {  
    BigDecimal value  
    Currency currency  
}
```

```
def fiveEuro = new MonetaryAmount(5.0, EUR)
```

Records: @RecordType kombiniert ...

@RecordBase

@RecordOptions

@TupleConstructor(namedVariant = true,
force = true, defaultsMode = DefaultsMode.AUTO)

@PropertyOptions

@KnownImmutable

@POJO

@CompileStatic

Java: Sealed Classes

```
sealed interface Figure permits Circle, Square {}  
record Circle(int radius) implements Figure {}  
record Square(int side) implements Figure {}
```

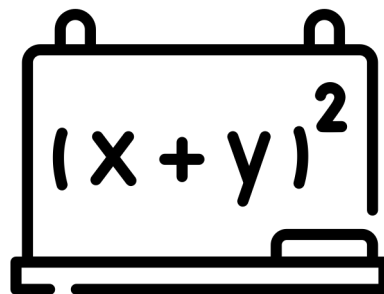


feingranulare Steuerung der Vererbungshierarchie

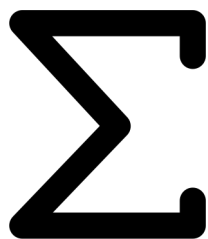
Algebraische Datentypen

Prüfung der “Exhaustiveness” im switch

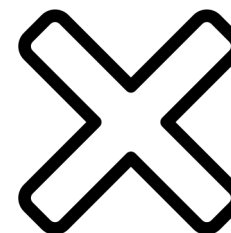




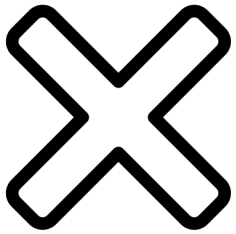
Algebraische Datentypen



Summentypen



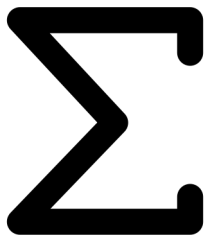
Produkttypen



Produkttypen

Java Beans, POJOs, Records:

Klassen mit Instanzvariablen



Summentypen

Enums, Sealed Classes

```
enum Figure { CIRCLE, SQUARE }
```

```
sealed interface Figure  
    permits Circle, Square {}
```

Sealed Types in Groovy

```
import groovy.transform.*
```

```
@Sealed interface Tree<T> {}
```

```
@Singleton final class Empty implements Tree {  
    String toString() { 'Empty' }  
}
```

```
@Canonical final class Node<T> implements Tree<T> {  
    T value  
    Tree<T> left, right  
}
```

```
Tree<Integer> tree = new Node<>(42, new Node<>(0, Empty.instance, Empty.instance),  
    Empty.instance)  
assert tree.toString() == 'Node(42, Node(0, Empty, Empty), Empty)'
```



Sealed Types in Groovy: 2. Beispiel

```
sealed abstract class Weather { }
```

```
class Rainy extends Weather { Integer rainfall }
```

```
class Sunny extends Weather { Integer temp }
```

```
class Cloudy extends Weather { Integer uvIndex }
```

```
def threeDayForecast = [  
    new Rainy(rainfall: 12),  
    new Sunny(temp: 35),  
    new Cloudy(uvIndex: 6)  
]
```

Sealed Types: Unterschiede zu Java

- **non-sealed (oder @NonSealed) ist nicht notwendig (ist der Default)**
- **@Sealed supports Sealed Classes für JDK8+ (emulated sealed classes)**
 - Groovy erkennt sie, der Java-Compiler nicht
 - für JDK 17+ schreibt Groovy die Informationen in den Bytecode
- **Klassen einer Sealed Hierarchie müssen nicht wie in Java im selben Package/Modul liegen**



JavaShell



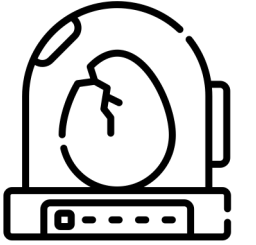
```
import org.apache.groovy.util.JavaShell

def opts = ['--enable-preview', '--release', '14']
def src = 'record Coord(int x, int y) {}'

Class coordClass = new JavaShell().compile('Coord', opts, src)

assert coordClass.newInstance(5, 10).toString() == 'Coord[x=5, y=10]'
```


POJO Annotation



```
@CompileStatic
@POJO
@Canonical(includeNames = true)
class Point {
    Integer x, y
}

@CompileStatic
@POJO
class PointList {
    @Delegate List<Point>
    points
}
```

Groovy Contracts: Design by Contract



```
import groovy.contracts.*

@Invariant({ speed() >= 0 })
class Rocket {
    int speed = 0
    boolean started = true

    @Requires({ isStarted() })
    @Ensures({ old.speed < speed })
    def accelerate(inc) { speed += inc }

    def isStarted() { started }
    def speed() { speed }
}

def r = new Rocket()
r.accelerate(5)
```



GINQ (Groovy-Integrated Query)



```
from p in persons
leftjoin c in cities on p.city.name == c.name
where c.name == 'Shanghai'
select p.name, c.name as cityName
```

```
from p in persons
groupby p.gender
having p.gender == 'Male'
select p.gender, max(p.age)
```

TOML Support



> Die **Syntax** von TOML gleicht der von .INI-Dateien und besteht primär aus **schluessel = "wert"-Paaren**, **[Abschnittsnamen]** und **# Kommentaren**.

Nach der Spezifikation werden folgende Datentypen unterstützt: **String, Integer, Float, Boolean, Datetime, Array und Table**.



WIKIPEDIA
Die freie Enzyklopädie

TOML lesen



```
def ts = new TomlSlurper()
def toml = ts.parseText '''language = "groovy"
sudo = "required"
dist = "trusty"
before_script = [ "unset _JAVA_OPTIONS\n\n \n" ]

[[matrix.include]]
jdk = "openjdk10"

[[matrix.include]]
jdk = "oraclejdk8" '''

assert 'groovy' == toml.language
assert 'required' == toml.sudo
assert 'trusty' == toml.dist
assert ['openjdk10', 'oraclejdk8'] == toml.matrix.include.jdk
assert ['unset _JAVA_OPTIONS'] == toml.before_script*.trim()
```

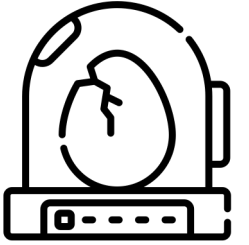
TOML schreiben



```
def builder = new TomlBuilder()
builder.records {
  car {
    name 'HSV Maloo'
    year 2006
    homepage new URL('http://example.org')
    record {
      type 'speed'
      description 'production pickup truck with speed of 271kph'
    }
  }
}

assert builder.toString() == '''\
records.car.name = 'HSV Maloo'
records.car.year = 2006
records.car.homepage = 'http://example.org'
records.car.record.type = 'speed'
records.car.record.description = 'production pickup truck with speed of 271kph'
'''
```

Enhanced Ranges



`1..10`

`1..<11`

`0<..10`

`0<..`



Agenda

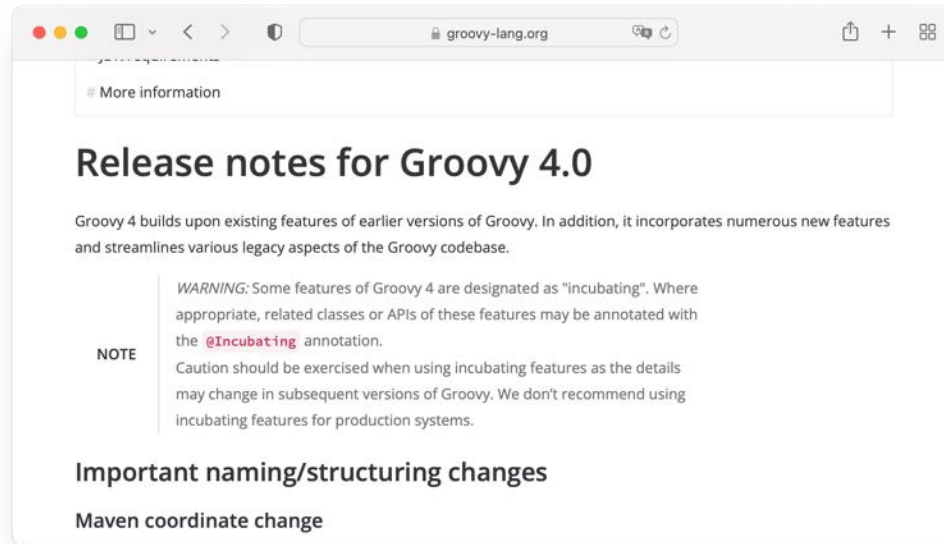


- 1 Was war nochmal Groovy?
- 2 Geschichte
- 3 Neue Features
- 4 **Fazit**

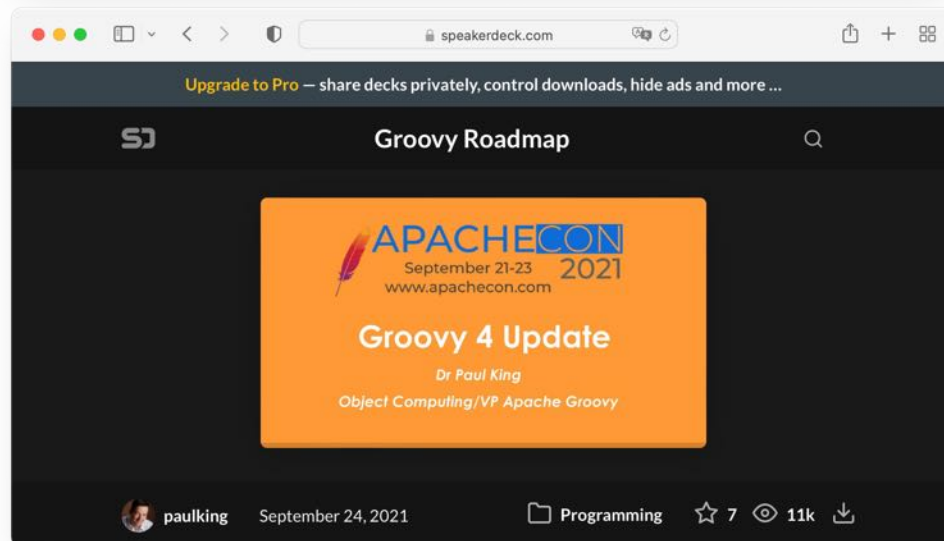
4



Weitere Infos



<https://groovy-lang.org/releases/groovy-4.0.html>



<https://speakerdeck.com/paulk/groovy-roadmap>



Groovy "Rockstars"

 **Jens Rossa** @ReussRunner  

@mittie wir sollten es sagen "einfach geil" ;) Hat Spass gemacht die Groovy-Session! nxt year dann evtl nicht im Keller, sondern im Ballsaal

 **Dierk König** @mittie · 23h
@reussrunner einfach @jaxcon sagen :-)

 **Jens Rossa** @ReussRunner · 22h
hej @jaxcon , bitte für den groovy talk von @mittie 2016 den Ballsaal frei machen. groovy rocks! (== im mainstream angekommen ;))

 **Martin** @PStibbons · 22h
@ReussRunner @jaxcon Genau das! @mittie 4 Ballsaal!

 **JAXCON** @jaxcon · 22h
@pstibbons wir geben bei der nächsten Planung unser bestes ;)



still rocks!



Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

Schwerpunkte:

- Architekturberatung und -bewertung
- Cloud- und Java-Technologien



fs@embarc.de



@sipp sack



→ [xing.to/fsi](https://www.xing.to/fsi)



Folien von heute als PDF zum Download



→ embarc.de/download/



Groo



Vielen Dank.

Ich freue mich auf Eure Fragen!



Falk Sippach



fs@embarc.de



@sipp sack



→ [xing.to/fsi](https://www.xing.to/fsi)