
# Agile Webentwicklung mit Ruby on Rails

Prof. Dr. Oliver Vornberger
Nils Haldenwang, B.Sc.

UNIVERSITÄT OSNABRÜCK

Institut für Informatik
AG Medieninformatik

# Ruby

## Einführung in die Syntax

# Regular Expressions

Ruby

Einführung in die Syntax

# Regular Expressions

- Pattern, das mit einem String abgeglichen werden kann
  - *String muss die Buchstabenfolge 'cat' enthalten*

- Mögliche Aktionen:
  - String auf Übereinstimmung mit Pattern prüfen
  - Teile aus String extrahieren
  - Teile des Strings ersetzen

# RegExp Literale

Literal

Matchoperator

abzugleichender String

```
/cat/ =~ "dog and cat"  # => 8   (match)
/cat/ =~ "catch"        # => 0   (match)
/cat/ =~ "Cat"          # => nil (no match)
/cat/i =~ "Cat"         # => 0   (match)

"catch" =~ /cat/        # => 0   (match)

"catch" !~ /cat/        # => false

c = "cat"
/#{c}/ =~ "catch"       # <=> /cat/ =~ "catch"

if /cat/ =~ "cat and dog"
  puts "There's a cat here somewhere."
end
```

Ergebnis:
Position
des Matches

kein Match:
nil

# Reservierte Zeichen

- . | ( ) [ ] { } + \ ^ $ * ?

```
/Perl|Python/ =~ "Perl is a scripting language."    # => 0
/Perl|Python/ =~ "Python is a scripting language." # => 0

/P(erl|ython)/ =~ "Perl is a scripting language."    # => 0
/P(erl|ython)/ =~ "Python is a scripting language." # => 0
/.(erl|ython)/ =~ "Python is a scripting language." # => 0

/\(cat\)/ =~ "(cat)" # => 0
```

# Wiederholungen

```
# + = one or more
/ab+c/ =~ "abc"   # => 0   (match)
/ab+c/ =~ "abbc"  # => 0   (match)
/ab+c/ =~ "ac"    # => nil (no match)

# * = zero or more
/ab*c/ =~ "abc"   # => 0   (match)
/ab*c/ =~ "abbc"  # => 0   (match)
/ab*c/ =~ "ac"    # => 0   (match)

# ? = zero or one
/ab?c/ =~ "abc"   # => 0   (match)
/ab?c/ =~ "abbc"  # => nil (no match)
/ab?c/ =~ "ac"    # => 0   (match)

/ab{1,2}c/ =~ "abc"    # => 0   (match)
/ab{1,2}c/ =~ "abbbc"  # => nil (no match)
/ab{1,2}c/ =~ "ac"     # => nil (no match)
```

# Ersetzungen

```
str = "Dog and Cat"

str.sub(/Cat/, "Dragon") # => Dog and Dragon

str.sub(/a/, "*")        # => Dog *nd Cat

str.gsub(/a/, "*")       # => Dog *nd C*t

str.gsub(/a/) { |match| match.upcase }
# => Dog And CAt
```

# RegExp Objekte

```ruby
name = "Wall-E"
/a/ =~ name                    # => 1
/a/.match(name)                # =>    # <MatchData "a">
Regexp.new("a").match(name) # =>    # <MatchData "a">

def show_regexp(string, pattern)
  match = pattern.match(string)
  if match
    "#{match.pre_match}<#{match[0]}>#{match.post_match}"
  else
    "no match"
  end
end

show_regexp("very interesting", /t/)
# => very in<t>eresting
```

# Anker

```
show_regexp("Mississippi", /iss/) # => M<iss>issippi

str = "this is\nthe time"

# start/end of line
show_regexp(str, /^the/)    # => this is\n<the> time
show_regexp(str, /is$/)     # => this <is>\nthe time

# start/end of string
show_regexp(str, /\Athis/) # => <this> is \nthe time
show_regexp(str, /\Athe/)  # => no match
show_regexp(str, /e\z/)    # => this is\nthe tim<e>

# word/nonword boundaries
show_regexp(str, /\bis/)   # => this <is>\nthe time
show_regexp(str, /\Bis/)   # => th<is> is\nthe time
```

# Zeichenklassen

Verneinung

andere Semantik bei reservierten Zeichen

```
show_regexp("Price $12", /[aeiou]/) # => Pr<i>ce $12
show_regexp("Price $12", /[$]/)     # => Price <$>12

str = "see [The PickAxe]"
show_regexp(str, /[A-F]/)     # => see [The Pick<A>xe]
show_regexp(str, /[A-Fa-f]/) # => s<e>e [The PickAxe]

show_regexp("Price $12", /[^\w]/) # => Price< >$12
show_regexp("Price $12", /\W/)    # => Price< >$12
```

\w = [A-Za-z0-9_]



Ruby

# Gruppierungen

\d = Ziffer

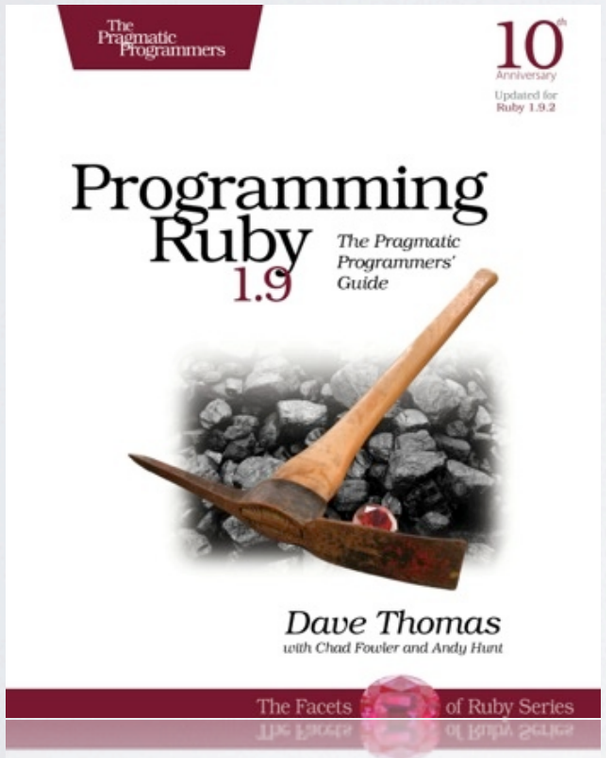
```
show_regexp("banana", /an+/)    # => b<an>ana

show_regexp("banana", /(an)+/) # => b<anan>a

regexp = /(\d\d):(\d\d)(\w{2})/
matches = regexp.match "12:50am"

"Hour is #{matches[1]}, minute is #{matches[2]}."
# => Hour is 12, minute is 50.

matches = /((\d\d):(\d\d))(\w{2})/.match "12:50am"

"Time is #{matches[1]}." #  => Time is 12:50.

"Hour is #{matches[2]}, minute is #{matches[3]}."
# => Hour is 12, minute is 50.

"AM/PM is #{matches[4]}" #  => AM/PM is am
```



Ruby

# Quellen / Literaturempfehlungen

http://www.ruby-doc.org/core-1.9.3/Regexp.html




Dave Thomas,
Chad Fowler,
Andy Hunt,
*Programming Ruby 1.9*,
Pragmatic Bookshelf, 2009
S. 99-118.



Ruby

# Demo

# Klassen und Objekte

Ruby

Einführung in die Syntax

# Klassendefinition

```ruby
class Person

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end

end

nils = Person.new("Nils", "Haldenwang")

puts nils.inspect
# #<Person:0x007ff59d9827f0
#       @first_name="Nils",
#       @last_name="Haldenwang">
```



Ruby

# Getter-Methoden

```ruby
class Person
  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end

  def first_name
    @first_name
  end

  def last_name
    @last_name
  end
end

nils = Person.new("Nils", "Haldenwang")

puts "#{nils.first_name} #{nils.last_name}"
#  Ausgabe: Nils Haldenwang
```



Ruby

# Class-Macro: attr_reader

```ruby
class Person
  attr_reader :first_name, :last_name

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end
end

nils = Person.new("Nils", "Haldenwang")

puts "#{nils.first_name} #{nils.last_name}"
#  Ausgabe: Nils Haldenwang
```

# Setter-Methoden

```ruby
class Person
  attr_reader :first_name, :last_name

  # initializer left out

  def first_name=(first_name)
    @first_name = first_name
  end

  def last_name=(last_name)
    @last_name = last_name
  end
end

nils = Person.new("Nils", "Haldenwang")
nils.first_name = "Han"
nils.last_name = "Solo"

puts "#{nils.first_name} #{nils.last_name}"
#  Ausgabe: Han Solo
```

# Class-Macro: attr_writer

```ruby
class Person
  attr_reader :first_name, :last_name
  attr_writer :first_name, :last_name

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end
end

nils = Person.new("Nils", "Haldenwang")
nils.first_name = "Han"
nils.last_name = "Solo"

puts "#{nils.first_name} #{nils.last_name}"
#  Ausgabe: Han Solo
```

# Class-Macro: attr_accessor

```ruby
class Person
  attr_accessor :first_name, :last_name

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end
end

nils = Person.new("Nils", "Haldenwang")
nils.first_name = "Han"
nils.last_name = "Solo"

puts "#{nils.first_name} #{nils.last_name}"
#  Ausgabe: Han Solo
```



Ruby

# Virtuelle Attribute

```ruby
class Person
  attr_accessor :first_name, :last_name

  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name  = last_name
  end

  def full_name
    "#{first_name} #{last_name}"
  end
end

nils = Person.new("Nils", "Haldenwang")

puts nils.full_name
#  Ausgabe: Nils Haldenwang
```

# Vererbung

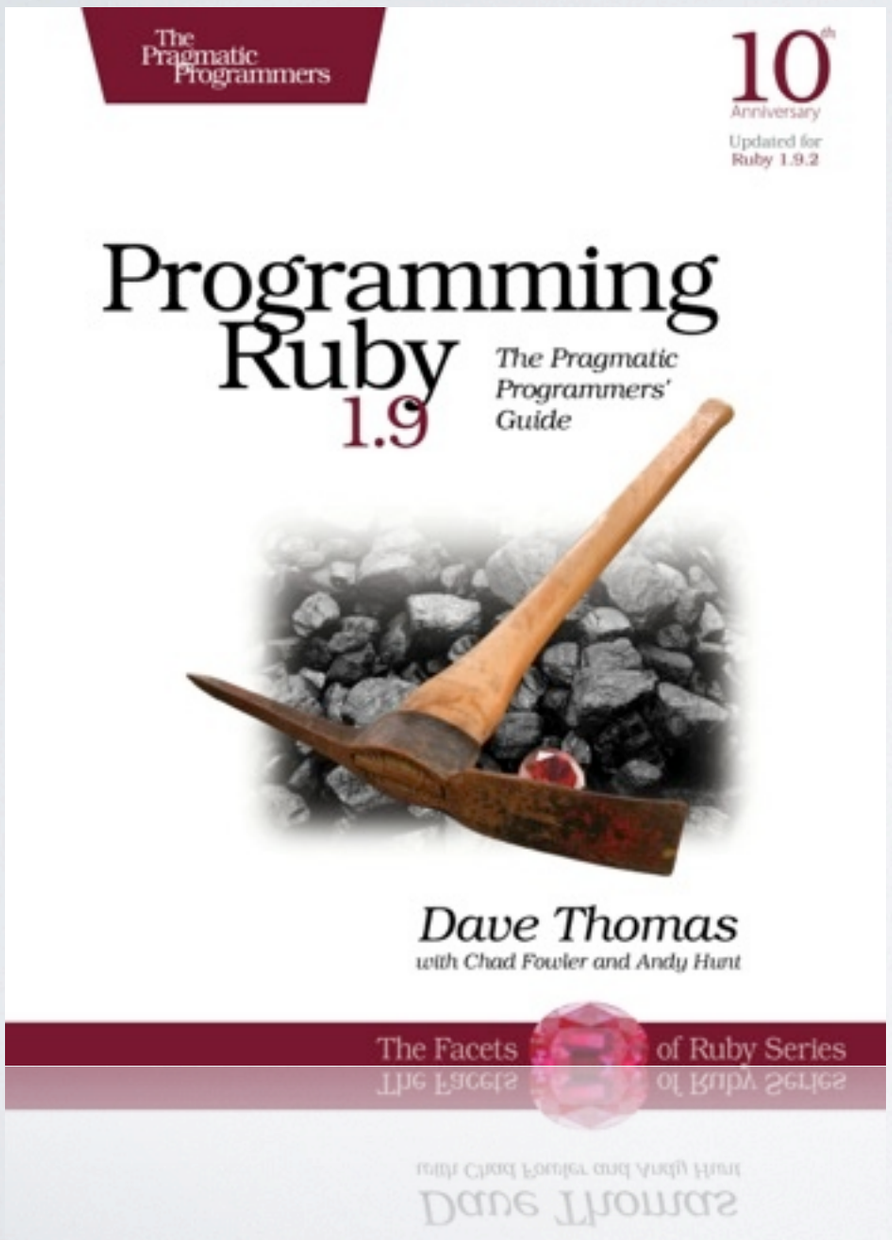
```ruby
class Student < Person
  attr_accessor :matriculation_number

  def initialize(first_name, last_name, matriculation_number)
    super first_name, last_name # call super initializer
    @matriculation_number = matriculation_number
  end
end

luke = Student.new("Luke", "Skywalker", 427323)

luke.full_name              # => Luke Skywalker
luke.matriculation_number # => 427323

luke.kind_of? Student     # => true
luke.kind_of? Person      # => true
luke.instance_of? Person # => false
```

# Offene Klassen

```ruby
class String
  def hellotize
    "Hello, #{self}!"
  end
end

"Joe".hellotize
#  => Hello, Joe!
```

"Ruby treats you as a grown-up programmer."

# Vertiefung




Ranges, S. 95-97.

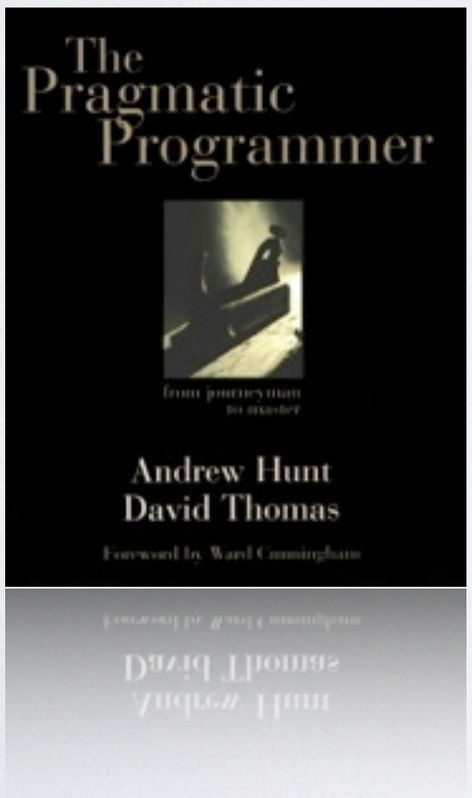
Case-When-Expression, S. 141 f.

# Ruby

## Editoren und IDEs

# Power Editing

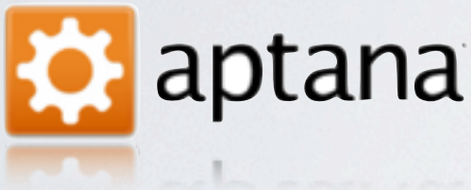
"You need to be able to manipulate text as effortlessly as possible, because text is the basic raw material of programming."




Andrew Hunt, David Thomas,
*The Pragmatic Programmer,*
25. Auflage, Addison-Wesley 2010

# IDEs


http://netbeans.org/features/ruby/




http://www.aptana.com/products/radrails


http://www.jetbrains.com/ruby/

# Text Editoren

http://macromates.com/

http://www.gnu.org/software/emacs/

http://www.vim.org/

# Anwendungsgebiete für Ruby

- übersichtliche Implementation komplexer Algorithmen
  - Beispiel: http://www.nils-haldenwang.de/computer-science/machine-learning/how-to-apply-naive-bayes-classifiers-to-document-classification-problems
- Automatisierung alltäglicher Aufgaben
  - Beispiel: Code-Highlighting für Vorlesungsfolien