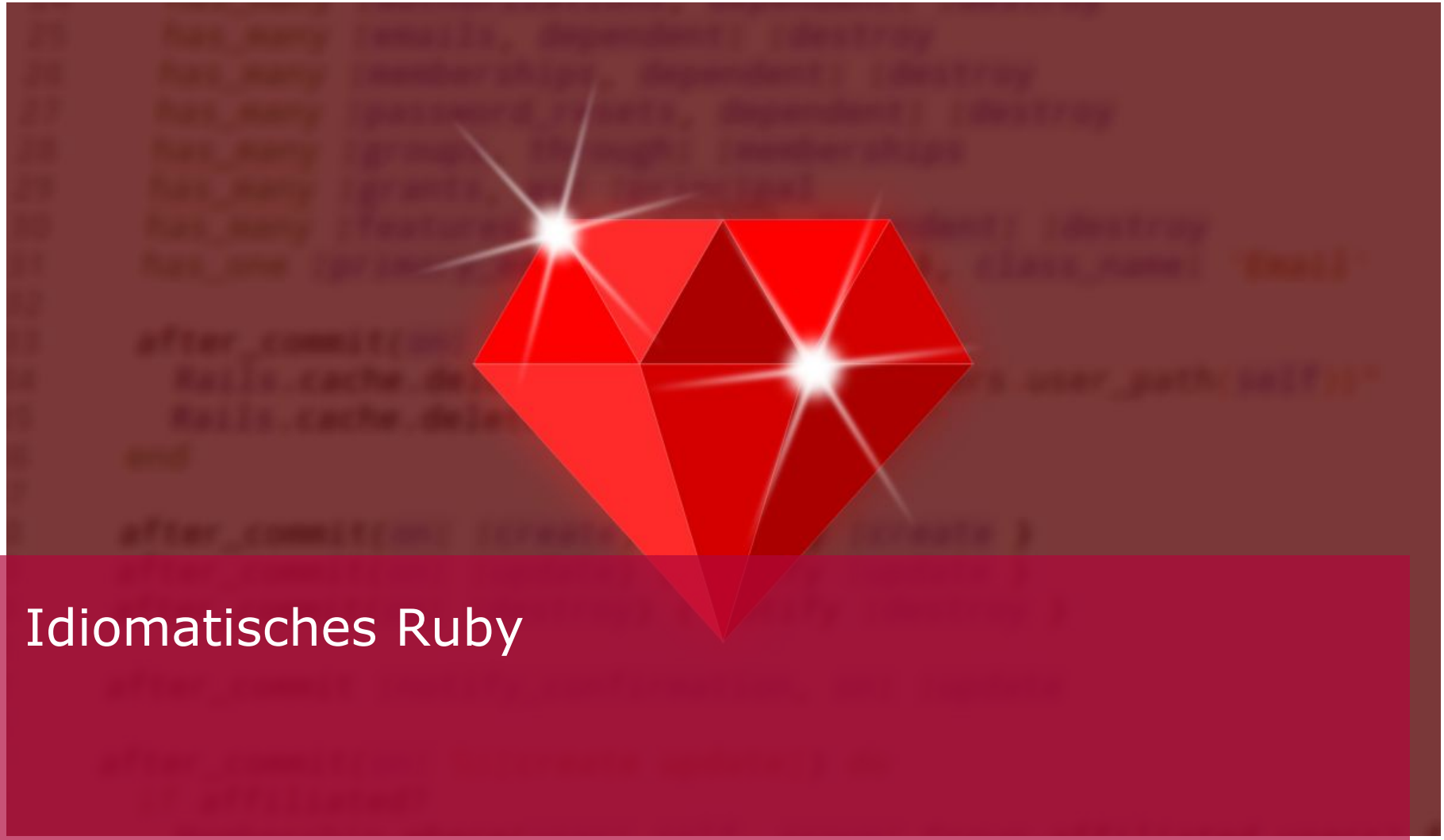
HPI Hasso Plattner Institut



Idiomatisches Ruby

Idiome

- Code-Schnipsel, die in Ruby-Quellcode oft auftauchen
- Verständnis hilft beim Lesen und Schreiben

Beispiele

- Verketten von Block-Methoden:

```ruby
["John", "Jane", "Jonathan"]
  .select { |name| name.length < 5 }
  .map    { |name| "Hello #{name}!" }
  .each   { |message| puts message }
```

- Interpolation von Zeichenketten:

```ruby
name = "Jo"

puts "Hallo #{name * 2}"
# Hallo JoJo
```

Idiome

HPI Hasso Plattner Institut

Beispiele

- “Gefährliche” Methoden (mit Seiteneffekten):

```ruby
a = [1, 2, 3]
a.map! { |num| num + 3 }
a # => [4, 5, 6]
```

- Fragemethoden:

```ruby
"Ruby lernen".empty? # => false

[1, 2, 3].any? { |num| num.even? } # => true
```

- Truthy und falsy / `if`-Einzeiler:

```ruby
puts "Wahr" if Object.new # Wahr
puts "Falsch" unless nil # Falsch
```

Mehrfachzuweisung

- Mehrere Variablen in einem Ausdruck mit Werten belegen
  - z.B. “Tausch” von Variablen-Werten
  - Rückgabe ist Liste aller Werte

```
string = "antwort"
number = 42

string, number = number, string
# => [42, "antwort"]
```

Safe Navigation Operator

- Aufruf von Methoden auf unbestimmten Objekten
  - Gibt `nil` zurück, wenn Empfänger `nil` ist

```
["a", "aa", "aaa"].first    # => "a"
[].first                    # => nil

["a", "aa", "aaa"].first.length
# => 1
[].first.length
# NoMethodError (undefined method `length` for nil:NilClass)

["a", "aa", "aaa"].first&.length
# => 1
[].first&.length
# => nil
```

Object#tap

- Übergibt Empfänger an Block, führt Block aus und gibt Empfänger zurück
  - Vermeidung von temporären Variablen
  - Fokus auf Objekt statt auf Initialisierung des Objektes

```ruby
def self.from_file(filename)
  instance = new
  instance.load(filename)
  instance
end
```

```ruby
def self.from_file(filename)
  new.tap do |instance|
    instance.load(filename)
  end
end
```

- Überprüfen von Zwischenergebnissen:

```ruby
["John", "Jane", "Jonathan"]
  .select { |name| name.length < 5 }
  .tap    { |filtered| puts filtered } # ["John", "Jane"]
  .map    { |name| "Hello #{name}!" }
  .each   { |message| puts message }
```

Memoization

- Potentiell teure Operationen so spät wie möglich ausführen
  - Speichern des Ergebnisses für weitere Aufrufe

```
class TaskList
  def initialize(filename)
    @filename = filename
  end

  def tasks
    @tasks ||= parse File.read(@filename)
  end

  def parse(contents)
    # ...
  end
end
```

Der Operator `||=` führt den darauffolgenden Ausdruck nur aus, wenn die zugewiesene Variable `nil` oder `false` ist. `a ||= b` ist eine Kurzform für `a = a || b`.

Memoization

- Auch komplexere Ausdrücke zwischen `begin...end` möglich

```
def options
  @options ||= begin
                 opts = { value: 3 }

                 if Time.now.friday?
                   opts[:happy] = true
                 end

                 # ...

                 opts
               end
end
```

Domain-Specific Languages (DSLs)

- Ruby-Syntax und Metaprogrammierung erlauben Konstrukte, die eigener Programmiersprache nahekommen
- Beispiel: Deklarative “Spec”-Syntax in `minitest`

```ruby
describe Hash do
  before do
    @hash = Hash.new
  end

  it "is empty by default" do
    @hash.must_be_empty
  end
end
```

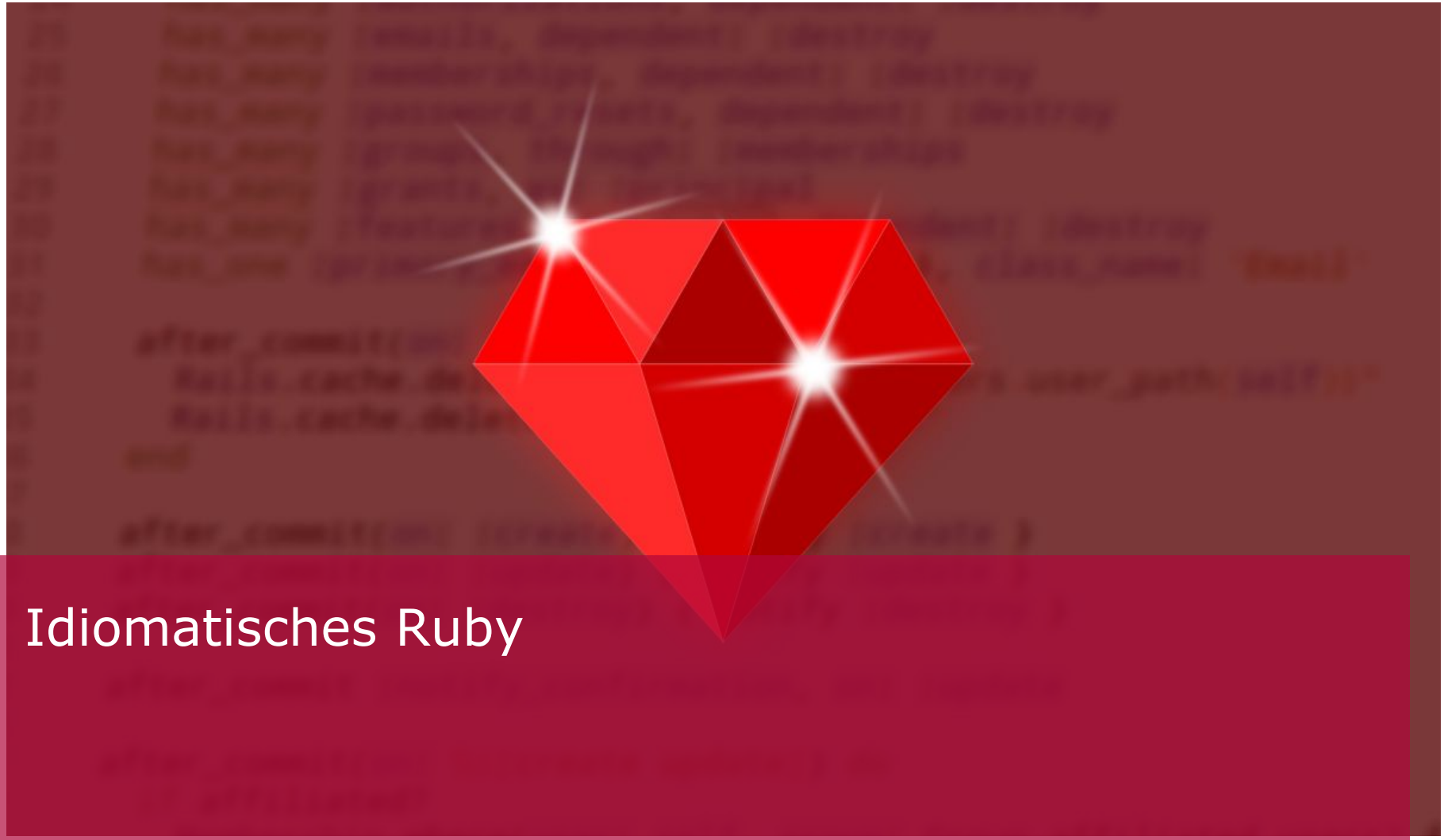
Ruby

Ruby is simple in appearance, but is very complex inside, just like our human body.

Ruby sieht von außen einfach aus, ist von innen aber sehr komplex, genau wie unser menschlicher Körper.

Yukihiro "Matz" Matsumoto
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2773

HPI Hasso Plattner Institut



Idiomatisches Ruby