

---

## Smalltalk: 1. Beispiel

hello

```
Transcript show: 'Hi World'
```

hello5

```
1 to: 5 do: [:i | (Transcript show: 'Hi World') cr]
```

hello: times

```
1 to: times do: [:i | (Transcript show: 'Hi World') cr]
```

---

# Smalltalk: Variablen und Parameter

```
hello: times say: text
    "Prints the text (several times) in Transcript window"
    (times > 100)
        ifTrue: [ Transcript show 'You will get bored!']
        ifFalse: [1 to: times do:
            [:i | (Transcript show: text) cr]]
```

```
hellox2: times say: text
    | timestwo |
    timestwo := 2 * times.
    self hello: timestwo say: text
```

---

# Smalltalk: Syntax Basics 1

**Namen:** großer Anfangsbuchstabe nur für globale Variablen (inkl. Klassen) und Klassenvariablen, sonst klein

**Kommentare:** in doppeltem Hochkomma ("Kommentar")

**Strings:** in einfachem Hochkomma ('x', 'x"', '')

**Character:** nach \$ (\$x, \$3, \$<, \$\$)

**Symbole:** nach # (#x, #'x', #do:, #ifTrue:ifFalse:)

**konst. Arrays:** #(1 2 + 3), #(1 2 (3 #(4)) 5)

**Zahlen:** 12, 3.14e-10, 2r101, 8r177, 16rFF, 2r1.1e2

---

## Smalltalk: Syntax Basics 2

**Assignment:** `var := expr` oder `var ← expr` (`← = _`)

**Pseudovariablen:** `nil`, `true`, `false`, `self`, `super`, `thisContext`

**unäre Nachricht:** `1.5 tan rounded` (= `(1.5 tan) rounded`)

**binäre Nachricht:** `3 + 4 * 5` (= 35, links nach rechts)

`3 + 4 factorial` (= 27, unäre Nachricht bindet stärker)

`(4/3) * 3 = 4` (= true, Brüche genau dargestellt)

`(3/4) == (3/4)` (= false, verschiedene Objekte)

**Keyword-Nachricht:** `1 to: 3 do: b` (sendet `#to:do:` an 1)

`(1 to: 3+4) do: b` (`#to:` mit 7 an 1, `#do:` an Ergebnis)

---

## Smalltalk: Syntax Basics 3

**Ausdrucksfolge:** box ← 20@30 corner: 60@90. "Punkt!"  
box containsPoint: 40@50. "hier optional"

**Ausdruckskaskade:** receiver unary; +23; at: 23 put: value

**Block:** [1. 2. 3], [: p1 p2 | p1+p2], [: p || v | v←p\*2. v+p]

**Blockausführung:** aBlock value (Block ohne Argumente)  
b value: a. b value: a1 value: a2 (bis zu 4 Argumenten)  
b valueWithArguments: anArray (Argumente in Array)

**Antwort-Ausdruck:** ↑ 2+3 (↑ = ^, terminiert Methode)

---

# Smalltalk: Syntax Basics 4

**bedingte Ausführung:** Nachrichten an aBoolean:

`#ifTrue:`, `#ifFalse:`, `#ifTrue:ifFalse`, `#ifFalse:ifTrue:`

Nachrichten an beliebige Objekte, Parameter Blöcke:

`#ifNil:`, `#ifNotNil:`, `#ifNil:ifNotNil:`, `#ifNotNil:ifNil:`

**Iterationen:** Empfänger, Parameter sind Blöcke:

`#whileTrue`, `#whileTrue:`, `#whileFalse`, `#whileFalse:`

**aufzählende Iterationen:** Nachrichten an anInteger:

`#timesRepeat:`, `#to:do:`, `#to:by:do:`

Nachricht an aCollection: `#do:`

Argumente nach `do:` sind Blöcke mit 1 Parameter  
andere Argumente sind Integer

---

## Smalltalk: Syntax Basics 5

**Auswahl:** aSymbol caseof: {[#a]->[1]. ['b' asSymbol]->[2]}  
aSymbol caseof: {[#a]->[1]. [#b]->[2]} otherwise: [3]

**Brace Arrays:** Arrays mit dynamisch berechneten Werten:  
{1. 2. 3}, {\$a. #brace. array}, {1 + 2}

### Klassendefinition:

```
SuperClass subclass: #NameOfClass
  instanceVariableNames: 'instVarName1 instVarName2'
  classVariableNames: 'ClassVarName1 ClassVarName2'
  poolDictionaries: ''
  category: 'Major-Minor'
```

---

# Smalltalk: Methodendefinition

lineCount

```
"Answer the number of lines represented by the receiver,  
where every cr adds one line."
```

```
| cr count |  
cr := Character cr.  
count := 1 min: self size.  
self do:  
    [:c | c == cr ifTrue: [count := count + 1]].  
^ count
```