Subscripts

Julian Wollersberger

Allgemein

- Erweiterte Index-Syntax
  ```
  arrr[3.141]
  matrix[42.0]["Nero"]
  ```
- Getter, Setter, Privacy
- Overloading
- Vererbbar
- Classes, Structures, Enumerations

Syntax

```
class WonderfulSubscript {
    subscript(index: Int) -> Int {
        get {
            return 42
        }
        set(newValue) {
            print("[\(index)] = \(newValue)")
        }
    }
}
let myWonderfulSubscipt = WonderfulSubscript();
myWonderfulSubscipt[3] = 42
```

Nachteil

- Unnötige Syntax
- Niemand fragte danach
- Verwirrend, weil Schwarze Magie
- Alternative:

```
class NobodyNeedsSubscripts {
    func get(index: Complex) -> Int { ... }
    func set(index: Complex,newValue: Int){...}
}
```

Mehrdimensional

- Durch mehrere Index-Parameter

```
subscript(x: Double, y:Double, z:Double,
          time: DateTime) -> Inhalt {
    // ...
}

raumzeit[3.14][15][926]["2053-05-08"] = 42
```

Overloading

- Mehrere subscript Blöcke
- Durch Typ entschieden

```
class Raumzeit {
    subscript(x: Double, y: Double, z: Double,
              time: DateTime) -> Inhalt {
        // ...
    }
    subscript(point: RaumZeitPunkt) -> Inhalt {

    }
}
```

Verkürzungen

- Read-Only

```
subscript(index: Int) -> Int {
    return 42
}
```

- Implicit newValue parameter

```
subscript(agent: String) -> Agent? {
    get { }
    set {
        internalDict[agent] = newValue
    }
}
```

Beispiel

- Index mit String und read-only:
  Gibt die Antwort auf das Leben, das Universum und alles zurück.
- Eigenes Dictionary mit 2 Indizes:
  Wochentag (als Enum) und Schulstunde liefern den Lehrer.