Match Expressions

Edit this page on GitHub

The syntactical precedence of match expressions has been changed. `match` is still a keyword, but it is used like an alphabetical operator. This has several consequences:

1. `match` expressions can be chained:

```
xs match {
  case Nil => "empty"
  case _   => "nonempty"
} match {
  case "empty"    => 0
  case "nonempty" => 1
}
```

(or, dropping the optional braces)

```
xs match
  case Nil => "empty"
  case _   => "nonempty"
match
  case "empty" => 0
  case "nonempty" => 1
```

2. `match` may follow a period:

```
if xs.match
  case Nil => false
  case _   => true
then "nonempty"
else "empty"
```

3. The scrutinee of a match expression must be an `InfixExpr`. Previously the scrutinee could be followed by a type ascription `: T`, but this is no longer supported. So `x : T match { ... }` now has to be written `(x: T) match {`

supported. So `x : T match { ... }` now has to be written `(x: T) match { ... }`.

Syntax

The new syntax of match expressions is as follows.

```
InfixExpr    ::=  ...
               |  InfixExpr MatchClause
SimpleExpr   ::=  ...
               |  SimpleExpr ‘.’ MatchClause
MatchClause  ::=  ‘match’ ‘{’ CaseClauses ‘}’
```

‹ Chang... Vararg ... ›