Fewer Braces

Edit this page on GitHub

By and large, the possible indentation regions coincide with those regions where braces `{ ... }` are also legal, no matter whether the braces enclose an expression or a set of definitions. There is one exception, though: Arguments to function can be enclosed in braces but they cannot be simply indented instead. Making indentation always significant for function arguments would be too restrictive and fragile.

To allow such arguments to be written without braces, a variant of the indentation scheme is implemented under language import

```
import language.experimental.fewerBraces
```

Alternatively, it can be enabled with command line option `-language:experimental.fewerBraces`.

This variant is more contentious and less stable than the rest of the significant indentation scheme. It allows to replace a function argument in braces by a `:` at the end of a line and indented code, similar to the convention for class bodies. It also allows to leave out braces around arguments that are multi-line function values.

Using `:` At End Of Line

Similar to what is done for classes and objects, a `:` that follows a function reference at the end of a line means braces can be omitted for function arguments. Example:

```
times(10):
  println("ah")
  println("ha")
```

The colon can also follow an infix operator:

```
credentials ++ :
```

```
credentials ++ :
  val file = Path.userHome / ".credentials"
  if file.exists

  then Seq(Credentials(file))
  else Seq()
```

Function calls that take multiple argument lists can also be handled this way:

```
val firstLine = files.get(fileName).fold:
    val fileNames = files.values
    s"""no file named $fileName found among
      |${values.mkString(\n)}""".stripMargin
  :
    f =>
      val lines = f.iterator.map(_.readLine)
      lines.mkString("\n)
```

Lambda Arguments Without Braces

Braces can also be omitted around multiple line function value arguments:

```
val xs = elems.map x =>
  val y = x - 1
  y * y
xs.foldLeft (x, y) =>
  x + y
```

Braces can be omitted if the lambda starts with a parameter list and `=>` or `=>?` at the end of one line and it has an indented body on the following lines.

Syntax Changes

```
SimpleExpr       ::=  ...
                    |  SimpleExpr `:` IndentedArgument
                    |  SimpleExpr FunParams (‘=>’ | ‘?=>’) IndentedArgument
InfixExpr        ::=  ...
                    |  InfixExpr id `:` IndentedArgument
IndentedArgument ::=  indent (CaseClauses | Block) outdent
```

Note that a lambda argument must have the `=>` at the end of a line for braces to be optional. For instance, the following would also be incorrect:

```
xs.map x => x + 1   // error: braces or parentheses are required
```

The lambda has to be enclosed in braces or parentheses:

The lambda has to be enclosed in braces or parentheses.

```
xs.map(x => x + 1)  // ok
```

‹ Experi... CanTh... ›