Automatic Eta Expansion

The conversion of *methods* into *functions* has been improved and happens automatically for methods with one or more parameters.

```
def m(x: Boolean, y: String)(z: Int): List[Int]
val f1 = m
val f2 = m(true, "abc")
```

This creates two function values:

```
f1: (Boolean, String) => Int => List[Int]
f2: Int => List[Int]
```

The syntax `m _` is no longer needed and will be deprecated in the future.

Automatic eta-expansion and nullary methods

Automatic eta expansion does not apply to "nullary" methods that take an empty parameter list.

```
def next(): T
```

Given a simple reference to `next` does not auto-convert to a function. One has to write explicitly `() ⇒ next()` to achieve that. Once again since the `_` is going to be deprecated it's better to write it this way rather than `next _`.

The reason for excluding nullary methods from automatic eta expansion is that Scala implicitly inserts the `()` argument, which would conflict with eta expansion. Automatic `()` insertion is limited in Scala 3, but the fundamental ambiguity remains.

More details