```
let name = expr
```

```
let name = expr1 in expr2
```

The `name` binding is only active in `expr2`.

Instances of shadowing, creating a hole in the previous binding's scope.

```
fun param1 ... paramN -> expr
```

By using the `begin` and `end` keywords.

As in Haskell, give the function name followed by the arguments, space delimited.

```
funcname arg1 ... argN
```

As in Haskell:

```
param1Type -> ... -> paramNType -> resultType
```

They associate right, so the following two are equivalent:

```
X -> Y -> Z
X -> (Y -> Z)
```

Simly leave off arguments, at the end of the parameter list. The result will be a new function.

```
let name param1 ... paramN = expr
```