Inferring Closure Types

If a closure contains a single expression, Swift will consider its body in addition to its signature and the surrounding context when performing type inference. For example, in the following code the type of `doubler` is inferred to be `(Int) -> Int` using only its body:

```
let doubler = {
  $0 * 2
}
```

If a closure body is not a single expression, it will not be considered when inferring the closure type. This is consistent with how type inference works in other parts of the language, where it proceeds one statement at a time. For example, in the following code an error will be reported because the type of `evenDoubler` cannot be inferred from its surrounding context and no signature was provided:

```
// error: cannot infer return type for closure with multiple statements; add
explicit type to disambiguate
let evenDoubler = { x in
  if x.isMultiple(of: 2) {
    return x * 2
  } else {
    return x
  }
}
```

This can be fixed by providing additional contextual information:

```
let evenDoubler: (Int) -> Int = { x in
 // ...
}
```

Or by giving the closure an explicit signature:

```
let evenDoubler = { (x: Int) -> Int in
 // ...
}
```