```
THE MANY FORMS OF SWIFT FUNCTIONS — A CHEATSHEET
================================================

No parameters, no return value

      func foo()

      called with

      foo()

No parameters, return type

      func foo() -> Int

      called with

      var a = foo()

Single parameter, no return value

      func foo(bar: Int)

      called with

      foo(10)

Single parameter, return value

      func foo(bar: Int) -> Int

      called with

      var a = foo(10)

Multiple parameters, return value

      func foo(bar: int, bash: String) -> Int

      called with

      var a = foo(10, "hello")

One or more parameters, multiple return values

      func foo(bar: Int) -> (bash: Int, baz: String)

      called with

      var a = foo(10, "Hello")

      a.bash = ...
      a.baz = "..."
```

One or more parameters, optional multiple return values

```
func foo(bar: Int) -> (bash: Int, baz: String)?
```

called with

```
if let a = foo(10) {
    println ("Bash is \(bash) and baz is \(baz)")
}
```

Default parameters

```
func foo(bar: Int, bash: String = "bash!") -> Int
```

called with

```
var a = foo(10)
var b = foo(10, "bling!")
```

External parameter names

```
func foo(externalBar bar: Int) -> Int
```

called with

```
var a = foo(externalBar: 10)
```

Shorthand external parameter names

```
func foo(#bar: Int, #bash: String) -> Int
```

called with

```
var a = foo(bar: 10, bash: "Bling!")
```

External parameter names with default parameters

```
func foo(#bar: Int, bash: String = "bash!") -> Int
```

called with

```
var a = foo(bar: 10)
var b = foo(bar: 10, bash: "Bling!")
```

Variadic parameters

```
func foo(bar: Int, bash: String...) -> Int
```

called with

```
var a = foo(10, "Bling!", "Clang!")
```

Variable parameters

```
      func foo(var bar: Int, var bash: String) -> Int

      called with

      var a = foo(10, "Bling!")

In-out parameters

      func foo(inout bar: Int, inout bash: String)

      called with

      var x = 10
      var y = "Bling!"
      foo(&x, &y)
```