

Defining variables

Constant

```
let constantVar = "Hello, World!"
```

Implicit

```
var implicitVar = "Hello, World!"
```

Explicit

```
var explicitVar: String = "Hello, World!"
```

Loop types

Range

```
for x in 1...5
```

Iterate through array

```
for item in [1, 2, 4, 5]
```

Iterate through dictionary

```
for (key, value) in ["one":1, "two":2, ...]
```

Switch methods

```
switch ["1", "2", "3", 4, 5, 6, (1, 100)] {
    case "1":
        print("1")
    case "2", "3":
        print("2 or 3")
        print("No fallthrough")
    case 4...5:
        print("Interval matching")
    case (1...100, 1...100):
        print("Tuples work the same.")
    default:
        print("6")
        print("Required in switchs")
        print("Must be last")
}
TODO: _, value binding, fallthrough, where
```

Function signature anatomy

Argument Label (optional) Used when calling the function

Parameter Name Used in the implementation of the function

Return Type (optional) Can be a tuple for multiple return values

throws (optional) Causes errors to propagate up.

```
func FunctionName(ArgumentLabel ParameterName: String) throws -> <ReturnType> {
    // Code
}
```

Event handling

No language constructs available.

Error handling

Representing errors

```
enum ErrorEnum: Error {
    case ErrorTypeA
    case ErrorTypeB: String
}
```

Propagating Errors Using Throwing Functions

```
func canThrowErrors() throws -> String
```

Handling Errors Using Do-Catch

```
do {
    try results = ThrowTypeB()
} catch ErrorEnum.ErrorTypeB(let errorString) {
    // Handle error
}
```

Converting Errors to Optional Values

```
let x = try?
someThrowingFunction()
```

Disabling Error Propagation

```
let x = try!
someThrowingFunction()
```