

USER FUNCTIONS

```
func greet(name: String) = {
  "Hello, " + name
}
```

- ◆ Declared with func
- ◆ Are not callable from outside
- ◆ Used only after their declaration
- ◆ Only in CONTENT_TYPE DAPP
- ◆ Without annotation
- ◆ Return type automatically inferred
- ◆ Last statement is the result
- ◆ Type come after argument's name

COMMENTS

```
# This is a comment line
# No multiline comments
"Hello world!" # You can do this
```

DIRECTIVES

```
{-# STDLIB_VERSION 3 #-}
{-# CONTENT_TYPE DAPP #-}
{-# SCRIPT_TYPE ACCOUNT #-}
```

- ◆ 2 content types: expression or dapp
- ◆ 2 script types: account or asset
- ◆ dApp content type is not allowed for an asset

VARIABLES

```
let a = "Bob"
let b = 1
```

- ◆ Declared with let only
- ◆ All variables are immutable
- ◆ Type inferred from right value side
- ◆ Global or function scope
- ◆ Unused variables not calculated

IF

```
if (20 > 42) then "do thing"
else "do something else"
```

- ◆ Can be used as an expression
- ◆ ELSE branch always required

BASIC TYPES

```
Boolean # true
String # "Hey"
Int # 1610
ByteVector # base58'...' etc.
```

STRING

- ◆ Operators needs same type values
- ◆ Only double quotes

SPECIAL TYPES

```
List # [16, 10, "hello"]
Nothing #
Unit # unit
```

UNIT

- ◆ No null type in RIDE
- ◆ Built-in functions return Unit type instead of null

LIST

- ◆ Known size only
- ◆ cons(1997, initList) to prepend element
- ◆ No way to concatenate two lists
- ◆ No way to prepend multiple values

UNION TYPE

```
Union(String | Unit)
```

- ◆ Can use pattern matching
- ◆ Return UNIT if key-value pairs don't exist
- ◆ Script will terminate in case of Unit
- ◆ Use getString + extract or getStringValue

EXCEPTIONS

```
throw("Here is exception text")
```

- ◆ Terminate execution immediately
- ◆ No ways to catch thrown exceptions
- ◆ Send feedback to the user
- ◆ Used for debug (no debugger in RIDE)

PATTERN MATCHING

```
let a = match getInteger(this, "someKey") {
  case a:Int => a
  case _ => 0
}
```

- ◆ Only for predefined types
- ◆ "_" represent every other cases

ANNOTATION @VERIFIER

```
@Verifier(tx)
func verify() = {
  match tx {
    case t: TransferTransaction => sigVerify(tx.bodyBytes,
      tx.proofs[0], tx.senderPublicKey)
    case _ => false
  }
}
```

- ◆ Always declared after USER function
- ◆ Only 1 Verifier per script
- ◆ Can't be called from the outside
- ◆ @Verifier always returns Boolean
- ◆ "tx" is an object with all fields of the current outgoing transaction
- ◆ No @Verifier = account/dApp key verification by default
- ◆ @Verifier to an account/dApp open transactions to anyone by default
- ◆ Set SigVerify on cases you needs then case _ => false
- ◆ tx.senderPublicKey is always key of the current account/dApp

ANNOTATIONS @CALLABLE

```
@Callable(i)
func callMaybe() = {
  let randomValue = getRandomValue()
  WriteSet([DataEntry("key", randomValue)])
}
```

- ◆ Always declared after USER function
- ◆ @Callable called from outside with InvokeScriptTransaction
- ◆ "i" is bonded to the function with invocation infos (callers' public key, address, payment attached to the transaction, fee, transactionId etc)

READING STATE

```
getInteger(this, "key")
getInteger(otherAccount, "key")
```

- ◆ func getInteger : Address, String => Int | Unit
- ◆ func getString : Address, String => String | Unit
- ◆ func getBoolean : Address, String => Boolean | Unit
- ◆ func getByteVector : Address, String => ByteVector | Unit
- ◆ Use extract() or value() to get the value, examples:

```
value(getString(this, "value"));      getString(this, "value").value()
extract(getString(this, "value"));    this.getInteger("key")
```

EXECUTION RESULTS

```
@Callable(i)
func giveAway(age: Int) = {
  ScriptResult(
    WriteSet([DataEntry("age", age)]),
    TransferSet([ScriptTransfer(i.caller, age, unit)])
  )
}
```

- ◆ Write own state
- ◆ Transfer own tokens
- ◆ @Callable return either ScriptResult, WriteSet or TransferSet
- ◆ WriteSet can contain up to 100 DataEntry
- ◆ TransferSet can contain up to 10 ScriptTransfer

FOLD

```
func sum(a:Int, b:Int) = a + b
let arr = [1,2,3,4,5]
let sum = FOLD<5>(arr, 0, sum) # result: 15
```

- ◆ Everything happens in pre-compile time
- ◆ You have to know the max size of your collection
- ◆ If you get more elements than expected, you get Exception



Join our Telegram



Use SigVerify to secure account if using a @Verifier

```
@Verifier(tx)
func verify() = {
  match tx {
    case t: TransferTransaction => sigVerify(tx.bodyBytes, tx.proofs[0], tx.senderPublicKey)
    case s: SetScriptTransaction => sigVerify(tx.bodyBytes, tx.proofs[0], tx.senderPublicKey)
    case _ => false
  }
}
```