Literals and Identifiers

FunC literals and identifiers are easy to understand. Keep reading for a full primer. See if you can spot fun ones.

Numeric literals

FunC allows decimal and hexadecimal integer literals. Numeric literals specify fixed and floating-point numbers. Hex integer literals use the base-16 number.

Below are valid FunC number literals:

- `0`, `123`, `-17`, `00987`
- `0xef`, `0xEF`, `0xEF`, `0x0`
- `-0xfFAb`, `0x0001`, `-0`, `-0x0`

String literals

A string literal is a set of characters or words inside quotation marks. FunC string literal tips:

> *They go inside quotes like:* `"geeky string literal"`.

> `\n` *and multi-line string literals don't work in FunC.*

> *Only used in asm function definitions.*

Identifiers

Below are valid FunC identifiers.

Simple	Complex
`CHECK ' _+_`	`? & fatal!`
`_internal_value`	`message_found?`
`elem0 elem1 elem2`	`query query' query''`
`_internal_value`	`(int, int) -> int`
`get_pubkeys&signatures`	`dict::udict_set_builder`

Below are fun FunC identifiers:

- `{hehehe}`
- `-alsovalidname`
- `0xefefefhahaha`
- `pa{--}in"`aaa`"`

Use cases

Symbol	Description
`'`	Using `'` at the end of a variable name is allowed. For example, all modifying built-in primitives for hashmap manipulation (except ones with prefix `~`) return a new hashmap version with data. We recommend using the same name followed by `'`. Example: `name` becomes `name'`.
`?`	Sometimes `?` is used for boolean variables. FunC TVM bools are represented by integers: 0 = false and -1 = true. `?` is returned as a flag indicating a successful operation like `udict_get?`. Read about the FunC standard library.
`'`	Using `'` at the end of a variable name is allowed with FunC. For example, all modifying built-in primitives for hashmap manipulation (except ones with prefix `~`) return a new hashmap version with data. We recommend you name the values with the same name followed by `'`.

Specials

FunC has special types of identifiers to use such as back quotes `` ` ``. In the quotes any symbol is allowed except for `\n`.

Examples below:

- `` `I'm a variable too` `` is valid
- `` `any symbols ; ~ () are allowed here...` `` is valid

Fatals

Below are fatal identifiers that don't work in FunC.

- `take(first)Entry`
- `"not_a_string`
- `msg.sender`
- `send_message,then_terminate`
- `_`
- `pa;;in"`aaa`"` (`;` is prohibited)
- `{-aaa-}`
- `aa(bb`
- `123` (it's a number)