

# Map! 関数

## 目次

1. 概要	1
2. 関数の定義	1
3. 関数の使用	1
4. valueの型	2
5. keyとvalueの型	3
6. keyの型	5
7. 関数の挙動	6

## 1. 概要

mapはkeyとvalueのペアを指定してmap関数を呼び出すことで、map関数は指定したkeyとvalueのペアをmap!関数で指定したkeyとvalueのペアに変換して、map!関数の結果を返す。map!関数は、map!関数の引数として、hash!、object!、map!、hash!、object!のいずれかを指定する。

## 2. 関数の定義

```
#(<key> <value>...)
```

```
<key> : 任意の型, 任意の型: scalar!, all-word!, any-string!
```

```
<value> : any-type!
```

## 3. 関数の使用

```
make map! <spec>
```

```
<spec> : keyとvalueのペアの整数
```

specは整数のペアの整数を指定する。map!関数は、map!関数の引数として、map!関数の結果を返す。

例:

- map!関数は、specの整数のペアの整数を指定する。
- map!関数は、reduce関数で指定したlogic!関数の結果を返す。

例:

```
#(a: 1 b: 2)
== #(
  a: 1
  b: 2
)

make map! [a 1 'b 2 "c" 3]
== #(
  a: 1
  b: 2
  "c" 3
)
```

key any-word! map key set-word!  
 map key value word key set-  
 word key word map keys-of  
 set-word word set-  
 word word

NOTE:

- hash! block! map! 空のmapを返す
- none key key
- map
- series! value map

map map copy

## 4. value

```
<map>/<key>
get '<map>/<key>

<map> : map! word
<key> : word key
```

select

```
select <map> <key>

<map> : map
<key> : key
```

map 関数の使用法は、`/case` オプションで指定する。

```
get/case '<map>/<key>
select/case <map> <key>
```

key が存在しない場合は `none` を返す。

例

```
m: #(Ab: 2 aB: 5 ab: 10)
m/ab
== 2
select m 'aB
== 2
get/case 'm/aB
== 5
select/case m 'ab
== 10
```

## 5. key-value

map の使用法

```
<map>/<key>: <value>
set '<map>/<key> <value>

<map>      : map!word
<key>      : mapvaluewordkey
<value>    : value
```

map の使用法

```
put <map> <key> <value>

<map> : map
<key> : mapvaluekey
```

map の使用法

```
extend <map> <spec>

<map> : map
<spec> : 1
```

map 的更新操作分为三种：set/case、put/case 和 extend/case。其中，set/case 和 put/case 都需要指定 key，而 extend/case 只需要指定 map。

```
set/case '<map>/<key> <value>'  
put/case <map> <key> <value>  
extend/case <map> <spec>
```

extend 操作需要指定 key，key 的值可以是任意合法的表达式。

NOTE:

- map 的 key 必须是合法的表达式，key 的值可以是任意合法的表达式。
- map 的 key 必须是合法的表达式，key 的值可以是任意合法的表达式，value 的值可以是任意合法的表达式。

□□

```

m: #(Ab: 2 aB: 5 ab: 10)
m/ab: 3
m
== #(
  Ab: 3
  aB: 5
  ab: 10
)

put m 'aB "hello"
m
== #(
  Ab: "hello"
  aB: 5
  ab: 10
)

set/case 'm/aB 0
m
== #(
  Ab: "hello"
  aB: 0
  ab: 10
)
set/case 'm/ab 192.168.0.1
== #(
  Ab: "hello"
  aB: 0
  ab: 192.168.0.1
)

m: #(%cities.red 10)
extend m [%cities.red 99 %countries.red 7 %states.red 27]
m
== #(
  %cities.red 99
  %countries.red 7
  %states.red 27
)

```

## 6. key

map key/value key none

```

m: #(a: 1 b 2 "c" 3 d: 99)
m
== #(
  a: 1
  b: 2
  "c" 3
  d: 99
)
m/b: none
put m "c" none
extend m [d #[none]]
m
== #(
  a: 1
)

```

## NOTE

mapは辞書型を返す

none!

␣

word!

mapは辞書型を返す。spec blockは辞書型を返す。

clearは辞書型を返す。keyは辞書型を返す。

```

clear #(a 1 b 2 c 3)
== #()

```

## 7. 辞書型

- findはkeyをmapから検索する。trueを返す。noneを返す。

```

find #(a 123 b 456) 'b
== true

```

- length?はmapのkey/valueの数を返す。

```

length? #(a 123 b 456)
== 2

```

- keys-ofはmapのkeyを辞書型から辞書型に変換する。

```

keys-of #(a: 123 b: 456)
== [a b]

```

- values-ofはmapのvalueを辞書型から辞書型に変換する。

```
values-of #(a: 123 b: 456)  
== [123 456]
```

- **body-of** `map` `key/value` `block`

```
body-of #(a: 123 b: 456)  
== [a: 123 b: 456]
```