

目 录

目 录

1. 前言.....	1
2. 1.1 背景.....	1
3. 背景.....	1
4. 背景.....	2
5. 背景.....	3
6. 背景.....	5
7. 背景Red/System.....	5
8. 背景.....	6
9. 背景.....	8
10. 背景.....	9
11. 背景.....	10
12. 背景.....	10

1. 前言

Red是一个基于DSL的编程语言，它的设计目标是提供一个简单、易用、且高效的编程环境。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。

Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。

Red/System是Red的一个子集，它专注于系统级编程。Red/System是Red的一个子集，它专注于系统级编程。Red/System是Red的一个子集，它专注于系统级编程。

Red/System是Red的一个子集，它专注于系统级编程。Red/System是Red的一个子集，它专注于系统级编程。Red/System是Red的一个子集，它专注于系统级编程。

2. 1.1 背景

1.1 背景。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。

3. 背景

Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。Red的DSL部分是其核心，它允许用户通过简单的语法来定义复杂的逻辑。

red/red是Red的一个子集，它专注于系统级编程。red/red是Red的一个子集，它专注于系统级编程。red/red是Red的一个子集，它专注于系统级编程。

□ □

□□□□

```
func [
  arg1
  arg2
  ...
][
  print arg1
  ...
]
```

```
func [
  arg1          ;-- 0000000000
  arg2
  ...
][
  print arg1    ;-- 000000000000
  ...
]
```

4. □□□□□□□□

□□□□□□□□ □□□□ □□□□ □□□□

[illegible][illegible]

```
[
  hello
][
  world
];-- 0000000000
```

[illegible]

```
array: [[ ] [ ] [ ] [ ]]
list:  [ [ ] [ ] [ ] [ ] ]

either a = 1 [ ["hello"]] [ ["world"]]
either a = 1 [ [ "hello" ] ] [ [ "world" ] ]
```

[illegible]

b: either $a = 1$ $[a + 1][3]$

[illegible]

1111

```
b: either a = 1 [  
    a + 1  
][  
    123 + length? mold a  
]
```

33

```
b: either a = 1
    [a + 1][123 + length? mold a]
```

Red either

[illegible]

```
print either a = 1 ["hello"][
    append mold a "this is a very long expression"
]

while [not tail? series][
    print series/1
    series: next series
]
```

5.

111

□ □ □ □ □ □

11

Red word

8. ☐ ☐ ☐ ☐ ☐

Red/System

□□□□

```
do-nothing: func [][]
increment: func [n [integer!]][n + 1]

increment: func [n [integer!]][
    n + 1
]

increment: func [
    n [integer!]
][
    n + 1
]
```

111

```
do-nothing: func [
][
]

do-nothing: func [

][

]

increment: func [
    n [integer!]
][n + 1]
```

000
00/local000000000000
000000word000000000000000000000000

```
make-world: func [
  earth    [word!]
  wind     [bitset!]
  fire     [binary!]
  water    [string!]
  /with
          thunder [url!]
  /only
  /into
          space   [block! none!]
  /local
  plants animals men women computers robots
][
  ...
]
```

```
make-world: func [
  [throw] earth [word!]           ;-- 00000000000000000000
    wind    [bitset!]
    fire [binary!]                 ;-- 000000000000
    water   [string!]
  /with
    thunder [url!]
  /only
  /into space [block! none!] ;-- /with 0000000000000000
  /local
    plants animals ;-- 00000000
    men women computers robots
][
  ...
]
```

7

```
increment: func ["Add 1 to the argument value" n][n + 1]

make-world: func [
    "Build a new World"
    earth      [word!]      "1st element"
    wind       [bitset!]    "2nd element"
    fire       [binary!]    "3rd element"
    water      [string!]
    /with      "Additional element"
    thunder    [url!]
    /only      "Not implemented yet"
    /into      "Provides a container"
    space      [unset!]     "The container"
    /local
    plants animals men women computers robots
][
    ...
]
```

```
make-world: func ["Build a new World" ;-- 世界の構築
  earth [word!]      "1st element"
  wind  [bitset!]    "2nd element" ;-- 風
  fire  [binary!]
  "3rd element"      ;-- `fire` 火
  water [string!]
  /with              "Additional element"
    thunder [url!]
  /only "Not implemented yet" ;-- 未実装のドキュメント
  /into
    "Provides a container" ;-- 世界はコンテナ
    space [unset!] "The container"
  /local
    plants animals men women computers robots
][
  ...
]
```

[illegible]


```
foo arg1 arg2 arg3 arg4 arg5

process-many
  argument1
  argument2
  argument3
  argument4
  argument5
```

```
foo arg1 arg2 arg3
    arg4 arg5

foo
    arg1 arg2 arg3
    arg4 arg5

process-many
    argument1
        argument2
            argument3
                argument4
                    argument5
```

```
head insert (copy/part [1 2 3 4] 2) (length? mold (2 + index? find "Hello" #"o"))

head insert
  copy/part [1 2 3 4] 2
  length? mold (2 + index? find "Hello" #"o")
```

[illegible]

11. □ □ □ □ □ □ □ □ □ □

`1000000 " " {}`

- `load()` method
- `save()` method

00000000001000000000 " 000000000000 { } 00000000000000000000 ^" 00000000000000000000

12. □□□□□□

11