Rust

Rust Rust “” C C++ Go JavaScript Rust

Rust Rust

Rust Scala Swift

```
let x = 8;
```

Rust

```
let x: i32 = 8;
```

x 8 x 8 “i32” 8 i32……

```
let mut x = 8;
```

Swift/Scala `let` `var`

Rust “”

```
let mut x: i32 = 1;
x = 7;
let x = x; //  x

let y = 4;
// 30 lines of code ...
let y = "I can also be bound to text!";
// 30 lines of code ...
println!("y is {}", y);      //  let y
```

Yin y   y x

`y` “” `let y = 4` `y` 30 `y` 30 `y` `y` let `y`

`let y = ...`

Rust C# Java

```
int x = 8;
```

```
let x = 8;  // x  i32
```

`x` i32“i32” C# C# varbool

```
var correct = ...;
var id = ...;
var slot = ...;
var user = ...;
var passwd = ...;
```

Visual Studio Visual Studio github code review C# Java

Rust C Java Rust

“”

Rust “”

```
let mut y = 5;
let x = (y = 6);  // x has the value `()`, not `6`
```

`x` tuple`()` OCaml OCaml OCaml `print_string`

```
print_string "hello world!\n";;
```

```
hello world!
- : unit = ()
```

`print_string` “”“statement” C `printf`“” OCaml “” `()` `()` C voidC void void

```
int main()
{
  void x;
}
```

C int void void void

Rust `y = 6` `()` `y = 6` “” 6 y `let x = (y = 6);` `y = 6` `()` tuple `()`

`let x = (y = 6);` `y = 6` Rust

JavaScript PHP undefined

return

Rust “” Rust “return”

```
fn add_one(x: i32) -> i32 {
    x + 1
}
```

return return

```
fn foo(x: i32) -> i32 {
    return x + 1;
}
```

“”

```
fn main() {
    println!("{}", add_one(7));
}

fn add_one(x: i32) -> i32 {
  if (x < 5) {
      if (x < 10) {
        // ...
        x * 2
      } else {
        // ...
        x + 1
      }
  } else {
    // ...
    x / 2
  }
}
```

if if if “return”“return”“return” if “”

“return”Rust “poor style”

“”

Rust Swift Swift Rust “mut”

```
fn main() {
    let m = [1, 2, 3];      //
    m[0] = 10;              //
    m = [4, 5, 6];          //
}
```

```
fn main() {
    let mut m = [1, 2, 3];  //
    m[0] = 10;              //
    m = [4, 5, 6];          //
}
```

Rust GCRC“”

 N GC RC GC RC

Rust move semantics, borrowing, lifetime blog “”

 Rust “fight with the borrow checker” lifetime parse

```
fn foo<'a, 'b>(x: &'a str, y: &'b str) -> &'a str {
}
```

 Rust lifetime `'a` `'b` lifetime lifetime

 Rust move semantics C Lifetime

Rust Linear Logic Linear Logic 1

“”workaroundGCRC

 lifetime C “” :P

…… Rust C Rust C JavaC# Swift Rust ……

530