

# ML | A Programming Language



[eqbal.me](http://eqbal.me)

## فهرست مطالب

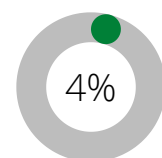


1. نمونه کد برنامه

2. معرفی

3. کاربرد

4. جزییات برنامه نویسی





اجزا



کاربرد



معرفی



نمونه کد

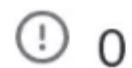
# o1/n2o

N2O: Standard ML Application Server



2

Contributors



0

Issues



19

Stars



4

Forks

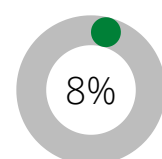


## o1/n2o

N2O: Standard ML Application Server. Contribute to o1/n2o development by creating an account on GitHub.



GitHub



8%

ML  
Meta Language

1400/3

1. بهینه بودن
2. چند نمونه ای ( Multi paradigm )
3. سادگی
4. تابع های رده بالا
5. بررسی نوع ایستا
6. چند ریختی
7. امکان پیاده سازی روابط جبری
8. ماژولار



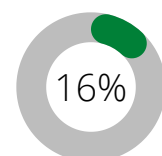
## ML ( Meta Language )

یک زبان برنامه نویسی تابعی همه منظوره

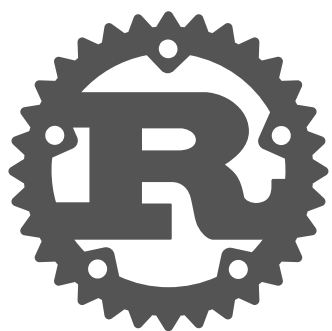
ساخته شده توسط رابین میلر و همکاران در اواخر دهه ۱۹۷۰

کلمه ML از فوق زبان (Meta language) گرفته شده است.

ML برای بهبود بخشیدن به رویه اثبات در قضیه LFC طراحی شده است.



- نوشتن کامپایلر
- اثبات قضیه های ریاضی



بیشتر زبان های حال حاضر از جمله:

Rust, Scala, C++, F#, Kotlin, Haskell

کامپایلرشان بر اساس زبان ML پیاده سازی شده است.

1. داده

2. عملیات اصلی

3. کنترل ترتیب

4. کنترل داده

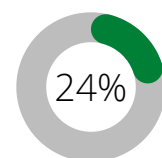
5. مدیریت حافظه

6. محیط عملیاتی



## داده اولیه

- `integer (int)`
- `reals (real)`
- `Bool (bool)`
- `Strings (string)`







اجزا



کاربرد



معرفی



نمونه کد

داده اولیه

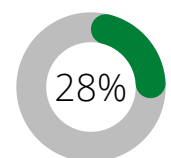
- عدد صحیح ( integers )

Attribute: int

Value: 1, 2, ~3, ~4

Operattors: +, -, \*, div, mod, <, <=, >, >=, <>

> val x = 7 : int



ML  
Meta Language

1400/3



داده اولیه

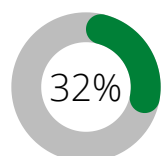
- عدد حقیقی ( Reals )

Attribute: real

Value: 3.0, 3.1415, ~3.2E2

Operation: +, -, \*, /, <, <=, >, >=, <>, log, exp, sin, arctan

> val length = 7.5 : real





داده اولیه

- بول ( bool )

Attribute: bool

Value: true, false,

Operation: not, or else(logical or), and also(logical and)

```
> val status = true : bool
```



داده اولیه

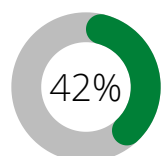
- رشته ( String )

Attribute: String

Value: "abc", "hello world\n", x^".sml"

Operation: ^(concatenation),length,substring

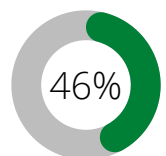
```
> val university = "uok" : string
```





## داده های ساختار یافته

- **Tuples:** ordered collection of values
- **Records:** collection of named values
- **Lists:** list of values of homogeneous type



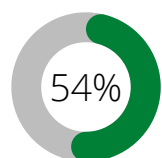


## داده های ساخت یافته

- تاپل ( Tuples )

```
>> Syntax: ( exp-list )  
- ( 1, 2, 3 );  
val it = (1,2,3) : int * int * int
```

```
>> Accessing by pattern matching or by label  
- val (a,b) = (2.3, "zippy");  
val a = 2.3: real  
val b = "zippy": string  
- #2 (a,b);  
val it = "zippy" : string
```





## داده های ساخت یافته

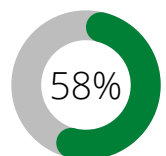
- رکورد ( Records )

>> A collection of labeled data items:

```
- val ex = { name = "eqbal", userid = 7 };  
{name:string, userid:int}
```

>> Accessing by pattern matching or by label

```
- #name ex;  
val it = "eqbal": string
```





## داده های ساخت یافته

- لیست ( Lists )

>> all elements must be of same type

>> Syntax:

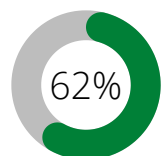
- [ 2, 6, 4, 9];

val it = [2,6,4,9]: int list

- [ "a", "b", "c"];

val it = ["a", "b", "c"]: string list

>> Accessing and Selection:







## عملیات اصلی

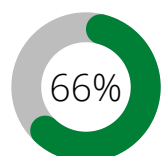
-> : => variable type assign

-> | => pip

-> ' => alpha

-> . -> member access

(\* comment \*)





## کنترل ترتیب

## Expressions

Arithmetic expressions

~

Unary minus

\*, /, *div*, *mod*

Multiplicative operators

+, -

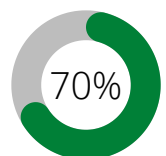
Additive operators

Each expression is terminated by a semicolon (;)

 $(expression_1; expression_2; \dots; expression_n)$ 

arithmetic relationships

=, &lt;&gt;, &gt;=, &lt;=, &lt;, and &gt;.



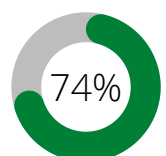


## کنترل ترتیب

If expression

**if** *expression* **then** *true\_part* **else** *false\_part*

```
13 fun max(ls : int list)=  
14   if null (tl ls)  
15   then hd ls  
16   else  
17     let val tmp = max(tl ls)  
18     in if (hd ls) < tmp then tmp else (hd ls)  
19   end;
```



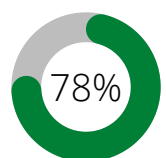


## کنترل ترتیب

### Function Definition

**fun** *function\_name* ( *parameters* ) = *expression*;

```
13  fun max(ls : int list)=  
14      if null (tl ls)  
15      then hd ls  
16      else  
17          let val tmp = max(tl ls)  
18          in if (hd ls) < tmp then tmp else (hd ls)  
19      end;
```





اجزا



کاربرد



معرفی

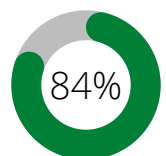


نمونه کد

## کنترل ترتیب

loop expression

**while** *expression<sub>1</sub>* **do** *expression<sub>2</sub>*



ML  
Meta Language

1400/3



## کنترل ترتیب

Exceptions:

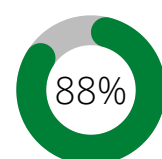
**exception** *exception\_name*

**raise** *exception\_name*

handle by:

*expression* **handle** *handler*

```
1 fun hd xs =  
2   case xs of  
3     [] => raise List.Empty  
4   | x::_ => x;  
5  
6  
7 (* *)  
8 exception MyException;  
9
```



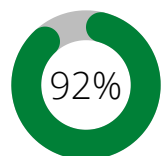


## کنترل ترتیب

### Structure:

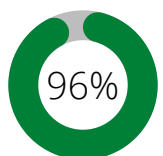
```
structure identifier = struct
  element1
  element2
  ...
  elementn end
```

```
- structure ListItem = struct
= val NewList = nil;
= fun add([ ],y)= y :: nil |
=      add(x,y) = x @ [y];
= fun size(x) = if x=[ ] then 0 else 1+size(tl(x));
= fun delete(x) = tl(x)
= end;
```



## کنترل داده

```
22 fun n_times(f, n, x)=
23     if n = 0
24     then x
25     else f(n_times(f, n - 1, x));
26
27 fun double x = x + x;
28 fun increment x = 1 + x;
29
30 (n_times(double, 4, 2) = 32);
31 (n_times(double, 4, 7) = 112);
32 (n_times(increment, 4, 2) = 6);
33 (n_times(tl, 4, [1, 2, 3, 4, 5]) = [5]);
```



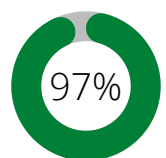


## مدیریت حافظه

**ML نیز مانند بیشتر زبان های تابعی دارای مدیریت حافظه خودکار است.**

مدیریت حافظه خودکار تکنیکی عمومی برای بازیافت حافظه اختصاص یافته به هر شی به صورت خودکار است. هرچند این از لحاظی غیر ممکن است که همیشه بتوان تعیین کرد که هر شی در حافظه مورد نیاز است یا خیر و به همین خاطر از روش های garbage collector برای رفع این مشکل استفاده شده است.

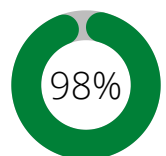
**در نسخه های کامپایلر جدید این زبان قابلیت پیشرفته garbage collectors را نیز به خوبی پشتیبانی میکند**





## محیط عملیاتی

محیط عملیاتی زبان از دو نوع `interactive` و `command line` پشتیبانی میکند و میتوان با استفاده از خط فرمان و نصف کامپایلر به اجرای دستورات زبان پرداخت.





اجزا



کاربرد



معرفی



نمونه کد



ML  
Meta Language

راه های ارتباطی:

