# yaml 格式

* 缩进  
  使用空格缩进，缩进距离为两个空格，不能用tab缩进
* .yaml 文件开头  
  以”---“ 开头，表明一个文件的开始
* 列表表示  
  使用一个短横杠加一个空格，多个项使用同样的缩进级别作为同一列表的一部分  
  - value
* 字典表示  
  以“key: value"的形式，冒号后面要加一个空格
* pool:
* name: "rally\_pool"
* protocol: "TCP"
* lb\_method: "LEAST\_CONNECTIONS"

subnet\_id: "f30db620-862c-4d3a-9e1c-d29f326389

一、简介

YAML 语言（发音 /ˈjæməl/ ）的设计目标，就是方便人类读写。它实质上是一种通用的数据串行化格式。

它的基本语法规则如下。

* 大小写敏感
* 使用缩进表示层级关系
* 缩进时不允许使用Tab键，只允许使用空格。
* 缩进的空格数目不重要，只要相同层级的元素左侧对齐即可

# 表示注释，从这个字符一直到行尾，都会被解析器忽略。

YAML 支持的数据结构有三种。

* 对象：键值对的集合，又称为映射（mapping）/ 哈希（hashes） / 字典（dictionary）
* 数组：一组按次序排列的值，又称为序列（sequence） / 列表（list）
* 纯量（scalars）：单个的、不可再分的值

以下分别介绍这三种数据结构

## 二、对象

对象的一组键值对，使用冒号结构表示。

animal: pets

转为 JavaScript 如下。

{ animal: 'pets' }

Yaml 也允许另一种写法，将所有键值对写成一个行内对象。

hash: { name: Steve, foo: bar }

转为 JavaScript 如下。

{ hash: { name: 'Steve', foo: 'bar' } }

## 三、数组

一组连词线开头的行，构成一个数组。

- Cat

- Dog

- Goldfish

转为 JavaScript 如下。

[ 'Cat', 'Dog', 'Goldfish' ]

数据结构的子成员是一个数组，则可以在该项下面缩进一个空格。

-

- Cat

- Dog

- Goldfish

转为 JavaScript 如下。

[ [ 'Cat', 'Dog', 'Goldfish' ] ]

数组也可以采用行内表示法。

animal: [Cat, Dog]

转为 JavaScript 如下。

{ animal: [ 'Cat', 'Dog' ] }

## 四、复合结构

对象和数组可以结合使用，形成复合结构。

languages:

- Ruby

- Perl

- Python

websites:

YAML: yaml.org

Ruby: ruby-lang.org

Python: python.org

Perl: use.perl.org

转为 JavaScript 如下。

{ languages: [ 'Ruby', 'Perl', 'Python' ],

websites:

{ YAML: 'yaml.org',

Ruby: 'ruby-lang.org',

Python: 'python.org',

Perl: 'use.perl.org' } }

五、纯量

纯量是最基本的、不可再分的值。以下数据类型都属于 JavaScript 的纯量。

* 字符串
* 布尔值
* 整数
* 浮点数
* Null
* 时间
* 日期

数值直接以字面量的形式表示。

number: 12.30

转为 JavaScript 如下。

{ number: 12.30 }

布尔值用true和false表示。

isSet: true

转为 JavaScript 如下。

{ isSet: true }

null用~表示。

parent: ~

转为 JavaScript 如下。

{ parent: null }

时间采用 ISO8601 格式。

iso8601: 2001-12-14t21:59:43.10-05:00

转为 JavaScript 如下。

{ iso8601: new Date('2001-12-14t21:59:43.10-05:00') }

日期采用复合 iso8601 格式的年、月、日表示。

date: 1976-07-31

转为 JavaScript 如下。

{ date: new Date('1976-07-31') }

YAML 允许使用两个感叹号，强制转换数据类型。

e: !!str 123

f: !!str true

转为 JavaScript 如下。

{ e: '123', f: 'true' }

## 六、字符串

字符串是最常见，也是最复杂的一种数据类型。

字符串默认不使用引号表示。

str: 这是一行字符串

转为 JavaScript 如下。

{ str: '这是一行字符串' }

如果字符串之中包含空格或特殊字符，需要放在引号之中。

str: '内容： 字符串'

转为 JavaScript 如下。

{ str: '内容: 字符串' }

单引号和双引号都可以使用，双引号不会对特殊字符转义。

s1: '内容\n字符串'

s2: "内容\n字符串"

转为 JavaScript 如下。

{ s1: '内容\\n字符串', s2: '内容\n字符串' }

单引号之中如果还有单引号，必须连续使用两个单引号转义。

str: 'labor''s day'

转为 JavaScript 如下。

{ str: 'labor\'s day' }

字符串可以写成多行，从第二行开始，必须有一个单空格缩进。换行符会被转为空格。

str: 这是一段

多行

字符串

转为 JavaScript 如下。

{ str: '这是一段 多行 字符串' }

多行字符串可以使用|保留换行符，也可以使用>折叠换行。

this: |

Foo

Bar

that: >

Foo

Bar

转为 JavaScript 代码如下。

{ this: 'Foo\nBar\n', that: 'Foo Bar\n' }

+表示保留文字块末尾的换行，-表示删除字符串末尾的换行。

s1: |

Foo

s2: |+

Foo

s3: |-

Foo

转为 JavaScript 代码如下。

{ s1: 'Foo\n', s2: 'Foo\n\n\n', s3: 'Foo' }

字符串之中可以插入 HTML 标记。

message: |

<p style="color: red">

段落

</p>

转为 JavaScript 如下。

{ message: '\n<p style="color: red">\n 段落\n</p>\n' }

## 七、引用

锚点&和别名\*，可以用来引用。

defaults: &defaults

adapter: postgres

host: localhost

development:

database: myapp\_development

<<: \*defaults

test:

database: myapp\_test

<<: \*defaults

等同于下面的代码。

defaults:

adapter: postgres

host: localhost

development:

database: myapp\_development

adapter: postgres

host: localhost

test:

database: myapp\_test

adapter: postgres

host: localhost

&用来建立锚点（defaults），<<表示合并到当前数据，\*用来引用锚点。

下面是另一个例子。

- &showell Steve

- Clark

- Brian

- Oren

- \*showell

转为 JavaScript 代码如下。

[ 'Steve', 'Clark', 'Brian', 'Oren', 'Steve' ]

## 八、函数和正则表达式的转换

这是 [JS-YAML](https://github.com/nodeca/js-yaml) 库特有的功能，可以把函数和正则表达式转为字符串。

# example.yml

fn: function () { return 1 }

reg: /test/

解析上面的 yml 文件的代码如下。

var yaml = require('js-yaml');

var fs = require('fs');

try {

var doc = yaml.load(

fs.readFileSync('./example.yml', 'utf8')

);

console.log(doc);

} catch (e) {

console.log(e);

}

从 JavaScript 对象还原到 yaml 文件的代码如下。

var yaml = require('js-yaml');

var fs = require('fs');

var obj = {

fn: function () { return 1 },

reg: /test/

};

try {

fs.writeFileSync(

'./example.yml',

yaml.dump(obj),

'utf8'

);

} catch (e) {

console.log(e);

}