Yaml

YAML（YAML Ain't Markup Language）是一种简洁的非标记语言。以数据为中心，使用空白，缩进，分行组织数据，从而使得表示更加简洁易读。

与Json对照：http://nodeca.github.io/js-yaml/

基本规则

YAML有以下基本规则： 1、大小写敏感 2、使用缩进表示层级关系，禁止使用tab缩进，只能使用空格键 3、缩进长度没有限制，只要元素对齐就表示这些元素属于一个层级。 4、使用#表示注释 5、字符串可以不用引号标注

数据结构

1、纯量

纯量是最基本的、不可再分的值。

- 数值直接以字面量的形式表示。
- 布尔值用 `true` 和 `false` 表示。
- `null` 用 `~` 表示。
- 时间采用 ISO8601 格式。如：2001-12-14t21:59:43.10-05:00
- 允许使用两个感叹号，强制转换数据类型。如：!!str 123、!!str true

2、map集合（对象）使用冒号（:）表示键值对，同一缩进的所有键值对属于一个map，示例：

```
# YAML表示
age : 12
name : itcast
```

3、list集合（数组）

使用连字符（-）表示，示例：

```
# YAML表示
- a
- b
- 12
```

数据结构嵌套

map和list的元素可以是另一个map或者list嵌套，示例：

1、map嵌套map

```yaml
# YAML表示
websites:
 YAML : yaml.org
 ITCAST : itcast.cn
```

2、map嵌套list

```yaml
# YAML表示
languages:
 - Java
 - Go
 - Python
 - c++
```

3、list嵌套list

```yaml
# YAML表示
-
  - Java
  - Go
  - Python
-
  - c
  - c++
  - PHP
# 简化方式
- - Java
  - Go
  - Python
- - c
  - c++
  - PHP
```

4、list嵌套map

```yaml
# YAML表示
-
  id: 1
  name: itcast
-
  id: 2
  name: heima
```

字符串

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' }
```

- 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"
```

- 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' }
```

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。

```
- &s Steve
- Clark
- Brian
- Oren
- *s
```

转为 JavaScript 代码如下。

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

函数和正则表达式的转换

这是 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);
}
```