

# Языки разметки JSON и YAML



Алексей  
Метляков



**Алексей Метляков**

DevOps Engineer, OpenWay



# План занятия

1. [Что такое языки разметки?](#)
2. [Для чего они нужны в DevOps?](#)
3. [Какие инструменты используются?](#)
4. [Синтаксис JSON](#)
5. [Синтаксис YAML](#)
6. [Возможности преобразования](#)
7. [Итоги](#)
8. [Домашнее задание](#)



# Что такое языки разметки?

---

# Суть языков разметки


Язык разметки - способ получения отформатированного текста на основе просто текста:

- Яркий пример - **HTML**
- **XML**, как стандарт
- Появление NoSQL key-value хранилищ данных и использование **JSON** для выгрузки

---

## Для чего они нужны в DevOps?

- Описание конфигураций серверов и шагов автоматизации, например **Ansible, Puppet**
- **Работа с API** Bitbucket, GitLab, GitHub, Nexus, Crowd, Jira, Confluence, etc.



**Какие инструменты  
используются?**

---

## Какие инструменты используются?

- **Ansible** - для описания всех структур данных
- **Puppet** (Hiera) - для описания конфигураций
- **Azure** - для построения Pipeline
- **Docker** (Compose) - для описания настройки сервисов, состоящих из нескольких контейнеров





# Синтаксис JSON

# Основа JSON

JSON - формат обмена данными, который легко читается “вживую”.

Он **состоит из двух видов объектов**:

- Коллекции пар ключ-значение (key-value)
- Упорядоченный список значений (array)

# Структура коллекции JSON

**Общая структура коллекции (объекта)** выглядит следующим образом:

{ ключ: значение }, где:

- Структура обязательно должна начинаться с { и заканчиваться на }
- Ключ - строковый тип, заключенный в двойные кавычки
- Значение может быть строкой, числом, массивом, объектом или иметь значение null, true или false
- Каждый элемент внутри структуры должен быть обособлен пробелом (горизонтальным табом, символом возврата каретки)

# Структура массива JSON

**Общая структура упорядоченного списка значений (массива)** выглядит следующим образом:

[ значение, значение ], где:

- Структура обязательно должна начинаться с [ и заканчиваться на ]
- Значение может быть строкой, числом, массивом, объектом или иметь значение null, true или false
- Каждый элемент внутри структуры должен быть обособлен пробелом (горизонтальным табом, символом возврата каретки)
- Массив может быть пустым, тогда его структура выглядит так: [ ]

# Строки JSON

В оформлении строк необходимо пользоваться правилами:

- Строка должна быть заключена в кавычки
- Спецсимволы должны экранироваться символом \
- Список спецсимволов
  - “
  - \
  - \
  - \b - backspace
  - \n - переход на новую строку
  - \f - переход на новую страницу
  - \r - возврат каретки
  - \t - горизонтальный таб
  - \u[четыре шестнадцатеричных цифры] - юникод

# Числа JSON

Числа представлены только в десятичном виде и в общем виде полная запись числа выглядит, как:

**[знак] [целая часть числа].[дробная часть числа][экспонента]**

Но нам, конечно же, никто не мешает упрощать форму до вида целых и дробных чисел



# Синтаксис YAML

---

# Основа YAML

Формат обмена данных YAML понравится любителям Python, но сложно даётся его противникам. Постараемся определить основные правила формирования yaml-файла:

- Файл должен начинаться с --- и заканчиваться на ...
- Каждый отдельный элемент коллекций стоит начинать с новой строки
- Каждый элемент коллекции list (массив) стоит начинать с символов “-” (тире и пробел)
- Комментарии начинаются с символа #



# Скаляры YAML

Скаляр, в случае с YAML, представляет собой единичный блок с информацией, которую можно записывать многострочно.

Существует два вида скаляров:

```
first:|
    Этот вид
    сохраняет все переходы на новую строку

second:>
    А этот
    преобразует каждый переход на новую строку
    в пробел
```

# Типы сущностей YAML

YAML поддерживает разнообразные типы данных. Например, целые числа могут быть:

```
12345 #Каноничные  
+1234 #Десятичные  
0o14 #Восьмеричные  
0xC #Шестандацтеричные
```

Числа с плавающей запятой:

```
1.2305e+3 #Каноничные  
21.2355e+02 #Экспонициальные  
1.4 #С фиксированной запятой  
+.inf #Бесконечность  
.NaN
```

# Типы сущностей YAML

Тут же присутствуют и другие типы:

```
null:  
booleans: [ true, false ]  
string: '1234'  
canon time: 2020-12-15T00:30:44.1Z  
date: 2020-07-31
```

# Коллекции YAML

По сути, это наборы данных в списке или словаре. Объявление списка имеет следующий синтаксис:


- Java
- Python
- Groovy

Чтобы объявить словарь нужно использовать другую конструкцию:

```
max: 100  
min: 10
```

Конечно же, оба эти типа можно комбинировать и использовать в разных вариациях, так например очень часто встречаются такие конструкции:

- name: Python  
 type: language  
 default: true  
 using: [ localhost, 7.7.7.7 ]



# **Возможности преобразования**

---

## Библиотеки yaml и json

Самый простой способ конвертации форматов - использовать методы библиотек yaml и json для Python:

- **Библиотека json** входит в стандартную поставку Python 3.x
- **Библиотеку yaml** необходимо установить: `python3 -m pip --user install pyyaml`

# Библиотека `yaml`

Библиотека позволяет загружать `yaml`-структуры, преобразуя их в стандартные объекты Python. В ней нас в первую очередь интересуют следующие методы:

- **`yaml.safe_load()`** - получает строку с `yaml` на вход и преобразует в объекты, с которыми может работать Python
- **`yaml.load_all()`** - считывает данные из файла и разделяет их на несколько
- **`yaml.dump()`** - получает объекты python на вход и преобразует в строку с `yaml`

# Библиотека json

Библиотека позволяет загружать json-структуры, преобразуя их в стандартные объекты Python. В этой библиотеке нас интересуют методы:

- **json.load()** - получает строку с json на вход и преобразует в объекты, с которыми может работать Python
- **json.dumps()** - получает объект python на вход и преобразует в строку с json
- **json.dump()** - получает объект python на вход, преобразует в json и записывает в файл



---

# Итоги

Сегодня мы:

- узнали, что такое языки разметки и для чего они нужны;
- познакомились с синтаксисами языков JSON и YAML;
- познакомились с возможностью их обработки при помощи Python



## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Алексей Метляков**