**YAML**



YAML — это язык для хранения информации в формате понятном человеку.

YAML vs JSON vs XML

YAML (.yml)

Особенности YAML:

* понятный человеку код;
* минималистичный синтаксис;
* заточен под работу с данными;
* встроенный стиль, похожий на JSON (YAML является его надмножеством);
* поддерживает комментарии;
* поддерживает строки без кавычек;
* считается «чище», чем JSON;
* дополнительные возможности (расширяемые типы данных, относительные якоря и маппинг типов с сохранением порядка ключей).

**Применение:** YAML лучше всего подходит для приложений с большим объемом данных, которые используют DevOps конвейеры или виртуальные машины. Кроме того, улучшение читаемости данных пригодится в командах, где разработчики часто с ними взаимодействуют.

JSON

Особенности JSON:

* труднее читать;
* явные, строгие требования к синтаксису;
* встроенный стиль, похожий на YAML (некоторые парсеры YAML могут читать JSON-файлы);
* нет комментариев;
* строкам нужны двойные кавычки.

Применение: JSON используется в веб-разработке — это лучший формат для сериализации и передачи данных через HTTP-соединение.

XML

Особенности XML:

* труднее читать;
* более многословный;
* действует как язык разметки, а YAML как язык для форматирования данных;
* больше возможностей чем у YAML, например атрибуты тегов;
* более жёсткая схема документа.

**Применение**: XML идеален для сложных проектов, которым требуется тонкий контроль над валидацией, схемой и пространством имён. Язык обладает плохой читаемостью, требует большей пропускной способности и ёмкости хранилища, но обеспечивает беспрецедентный контроль.

## Характерные особенности YAML

### Поддержка мультидокументов

Вы можете объединить несколько YAML-документов в один YAML-файл для облегчения организации файлов и парсинга данных.

Документы разделяются тремя дефисами (---):

---

player: playerOne

action: attack (miss)

---

player: playerTwo

action: attack (hit)

---

### Поддержка комментариев

YAML позволяет добавлять комментарии после символа #, как в Python:

key: #Это однострочный комментарий

- value line 5

#Это

#многострочный комментарий

- value line 13

### Легко читаемый синтаксис

В синтаксисе YAML-файлов используется система отступов, как в Python. Необходимо использовать пробелы, а не табуляцию, чтобы избежать путаницы.

Это избавляет от лишних символов, которые есть в JSON и XML (кавычки, скобки, фигурные скобки).

В итоге читаемость файла значительно повышается.

### YAML

#YAML

Imaro:

author: Charles R. Saunders

language: English

publication-year: 1981

pages: 224

### JSON

{

"Imaro": {

"author": "Charles R. Saunders",

"language": "English",

"publication-year": "1981",

"pages": 224

}

}

### Явная и неявная типизация

YAML предлагает как автоопределение типов, так и возможность явно указать тип данных. Чтобы использовать конкретный тип, нужно написать !![тип] перед значением.

# Это значение преобразуется в int:

is-an-int: !!int 14.10

# Превращает любое значение в строку:

is-a-str: !!str 67.43

# Значение должно быть boolean:

is-a-bool: !!bool yes

### Отсутствие исполняемых файлов

YAML не содержит исполняемых файлов. Поэтому можно безопасно обмениваться YAML-файлами с третьей стороной.

Чтобы использовать исполняемые файлы, YAML нужно интегрировать с другими языками, например Perl или Java.

## Синтаксис YAML

В языке есть несколько базовых концепций, которые позволяют обрабатывать большинство данных.

### Пары ключ-значение

Большинство данных в YAML-файле хранятся в виде пары ключ-значение, где ключ — это имя пары, а значение — связанные данные.

### Скаляры и маппинг

Скаляр представляет собой одно значение, которому соответствует имя.

YAML поддерживает стандартные типы: int и float, boolean, string и null.

Они могут быть представлены в разных видах: шестнадцатеричном, восьмеричном или экспоненциальном. Также существуют специальные типы для математических сущностей, такие как: бесконечность, -бесконечность и NAN.

integer: 25

hex: 0x12d4 #равно 4820

octal: 023332 #равно 9946

float: 25.0

exponent: 12.3015e+05 #равно 1230150.0

boolean: Yes

string: "25"

infinity: .inf # преобразуется в бесконечность

neginf: -.Inf #преобразуется в минус бесконечность

not: .NAN #Not a Number

null: ~

### Строки

Строка — это коллекция символов, которая может содержать слово или предложение. Можно использовать либо |, для отдельных строк, либо >, для параграфов.

Кавычки в YAML не нужны.

str: Hello World

data: |

Это

Отдельные

Строки

data: >

Это

один параграф

текста

### Последовательности

Последовательности — это структуры данных похожие на списки или массивы, которые хранят несколько значений под одним ключом. Они определяются с помощью отступов или [].

shopping:

- milk

- eggs

- juice

Однострочные последовательности выглядят лаконичнее, но хуже читаются.

shopping: [milk, eggs, juice]

### Словари

Словари — это коллекции пар ключ-значение, которые хранятся под одним ключом. Они позволяют разделить данные на логические категории.

Employees:

- dan:

name: Dan D. Veloper

job: Developer

team: DevOps

- dora:

name: Dora D. Veloper

job: Project Manager

team: Web Subscriptions

Словари могут содержать более сложные структуры, что позволяет хранить сложные реляционные данные.