




# Lesson 18

25.07.2022


```
public class Ex1 {  
    public static void main(String[] args) {  
        long year = 2011;  
        System.out.print(year);  
    }  
}
```

```
public class Ex2 {  
    public static void main(String[] args) {  
        do  
            while(true)  
                ;  
        System.out.println("HELLO");  
    }  
}
```

```
public class Ex3 {  
    public static void main(String[] args) {  
        double $ = 0XD_EP2F;  
        System.out.print($);  
    }  
}
```



```
public class Ex4 {  
    public static void main(String[] args) {  
  
        ((Ex4) null).Одесса();  
    }  
  
    static void Одесса() {  
        System.out.println("Одесса город у моря ... ");  
    }  
}
```



```
public class Ex5 {  
    public static void main(String[] args) {  
        int[] mass = {1, 2};  
        List<String> list = new ArrayList(10);  
        list.add("03");  
        list.add("04");  
        System.out.println(mass.length + list.size() + ".");  
    }  
}
```



{JSON}



YAML



**XML** (*eXtensible Markup Language*) – универсальный и расширяемый язык разметки данных, который не зависит от операционной системы и среды обработки. Xml служит для представления неких данных в виде структуры, и эту структуру Вы можете сами разработать или подстроить под ту или иную программу или какой-то сервис. Именно поэтому данный язык называют расширяемый, и в этом является его главное достоинство, за которое его так ценят.

```
<?xml version="1.0"?>
- <job>
  - <production>
    <ApprovalType>WebCenter</ApprovalType>
    <Substrate>carton 150 gr</Substrate>
    <SheetSize>220-140</SheetSize>
    <press>SuperFlat2</press>
    <finishing>standard</finishing>
    <urgency>normal</urgency>
  </production>
  - <customer>
    <name>FruitCo</name>
    <number>2712</number>
    <currency>USD</currency>
  </customer>
</job>
```






## XML теги

Язык XML для разметки использует теги (*теги регистрозависимы*), но не такие теги как в html, а те, которые Вы придумаете сами, но у xml документа есть также четкая структура, т.е. есть открывающий тег и закрывающий, есть вложенные теги и есть, конечно же, значения, которые расположены в этих тегах. Другими словами, все, что нужно для начальных знаний xml - это просто придерживаться этим правилам. Все вместе открывающий, закрывающий тег и значение называется элементом и весь xml документ состоит именно из элементов, которые в совокупности образуют структуру данных. У xml документа может быть только один корневой элемент, это запомните, так как если Вы напишите два корневых элемента, то это будет ошибка.

```
<start>
  <name>
    Alex Stepurko
  </name>
</start>
```

Как видите, я здесь просто привел пример своего рода контакта, но я не объявлял этот документ, т.е. не писал XML декларацию, которая говорит приложению, которое будет обрабатывать эти данные, что здесь расположены данные именно xml и в какой кодировке они представлены. Также можно писать комментарии и атрибуты, так давайте приведем пример такого документа:

```
<?xml version="1.0" encoding="UTF-8"?>
<start>
  <name>
    Alex Stepurko
  </name>
</start>
```



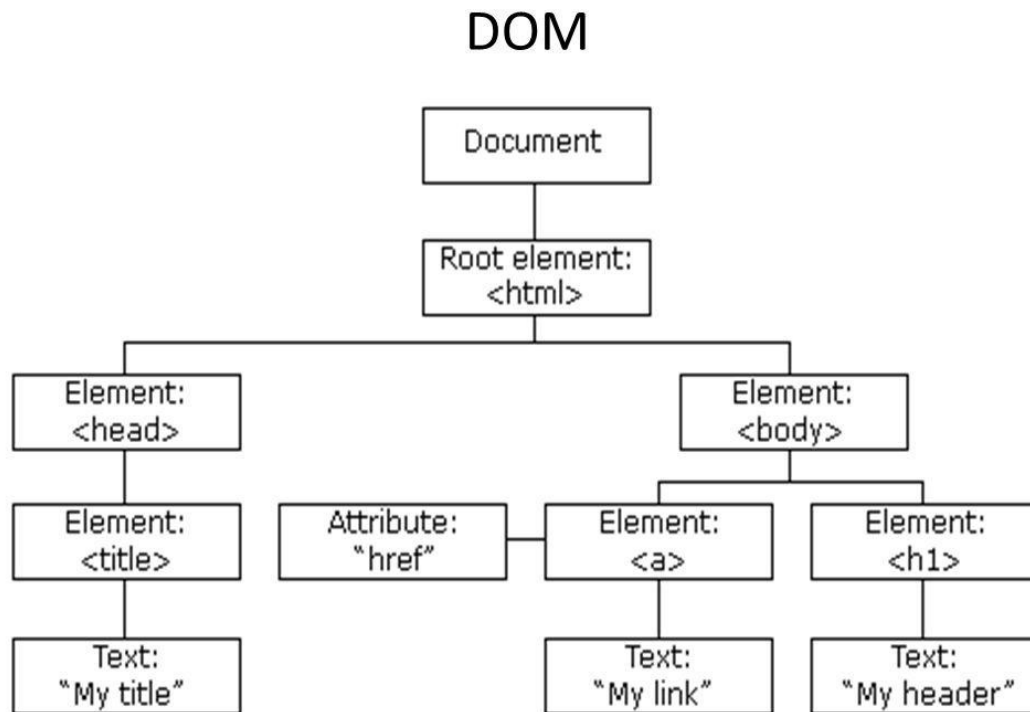
**SAX** ([англ.](#) «*Simple API for XML*») — последовательное чтение из источника XML. Документ читается последовательно по кусочкам (событиям).

Все события обрабатываются в **обработчике событий**, который нужно создать и **переопределить методы**.

**Преимущества:** высокая производительность благодаря "прямому" способу считывания данных, низкие затраты памяти.

**Недостатки:** ограниченная функциональность, а, значит, в нелинейных задачах дорабатывать её надо будет уже нам.

DOM (Document Object Model) - DOM-обработчик устроен так, что он считывает сразу весь XML и сохраняет его, создавая иерархию в виде дерева, по которой мы можем спокойно двигаться и получать доступ к нужным нам элементам.





Таким образом, мы можем, имея ссылку на верхний элемент, получить все ссылки на его внутренние элементы. При чем элементы, которые внутри элемента – дети этого элемента, а он – их родитель. Однажды считав весь XML в память, мы просто будем путешествовать по его структуре и выполнять нужные нам действия. Немного уже о программной части DOM в Java: в DOM есть множество интерфейсов, которые созданы, чтобы описывать разные данные. Все эти интерфейсы наследуют один общий интерфейс – Node (узел). Потому, по сути, самый частый тип данных в DOM – это Node (узел), который может быть всем.



У каждого Node есть следующие полезные методы для извлечения информации:

**getNodeName** – получить имя узла.

**getNodeValue** – получить значение узла.


**getNodeType** – получить тип узла.

**getParentNode** – получить узел, внутри которого находится данный узел.

**getChildNodes** – получить все производные.

**getAttributes** – получить все атрибуты узла.

**getTextContent** – возвращает весь текст внутри элемента и всех элементов внутри данного элемента, включая переносы строчек и пробелы.



**JSON** это сокращение от JavaScript Object Notation — формата передачи данных. Как можно понять из названия, JSON произошел из JavaScript, но он доступен для использования на многих других языках, включая Python, Ruby, PHP и Java

Сам по себе JSON использует расширение .json. Когда же он определяется в других файловых форматах, как .html, он появляется в кавычках как JSON строка или может быть объектом, назначенным на переменную. Такой формат легко передавать между сервером и клиентской частью, ну или браузером.

Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента. Это информативное руководство поможет вам быстрее разобраться с данными, которые вы можете использовать с JSON и основной структурой с синтаксисом этого же формата.

## Синтаксис и структура

Объект JSON это формат данных — ключ-значение, который обычно рендерится в фигурных скобках. Когда вы работаете с JSON, то вы скорее всего видите JSON объекты в .json файле, но они также могут быть и как JSON объект или строка уже в контексте самой программы.

```
{  
  "first_name" : "Sammy",  
  "last_name" : "Shark",  
  "location" : "Ocean",  
  "online" : true,  
  "followers" : 987  
}
```





В качестве значений в JSON могут быть использованы:

**Запись** — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.

**Массив (одномерный)** — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения.

**Число** (целое или вещественное).


**Литералы** true (логическое значение «истина»), false (логическое значение «ложь») и null.

**Строка** — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\»



```
<users>
  <user>
    <username>SammyShark</username> <location>Indian
Ocean</location>
  </user>
  <user>
    <username>JesseOctopus</username> <location>Pacific
Ocean</location>
  </user>
  <user>
    <username>DrewSquir</username> <location>Atlantic
Ocean</location>
  </user>
  <user>
    <username>JamieMantisShrimp</username> <location>Pacific
Ocean</location>
  </user>
</users>
```

```
{ "users": [
  { "username" : "SammyShark", "location" : "Indian Ocean"},
  { "username" : "JesseOctopus", "location" : "Pacific Ocean"},
  { "username" : "DrewSquid", "location" : "Atlantic Ocean"},
  { "username" : "JamieMantisShrimp", "location" : "Pacific Ocean"}
] }
```



**YAML** (акроним англ. «Yet Another Markup Language» — «Ещё один язык разметки», позже — рекурсивный акроним англ. «YAML Ain't Markup Language» — «YAML — не язык разметки») — «дружественный» формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования.

В трактовке названия отражена история развития: на ранних этапах YAML расшифровывался как Yet Another Markup Language («Ещё один язык разметки») и даже позиционировался как конкурент XML, но позже был переименован с целью акцентировать внимание на данных, а не на разметке документов



Согласно целям, озвученным Кларком Эвансом, YAML 1.0 призван:

- быть легко понятным человеку;
- поддерживать структуры данных, родные для языков программирования;
- быть переносимым между языками программирования;
- использовать цельную модель данных для поддержки обычного инструментария;
- поддерживать потоковую обработку;
- быть выразительным и расширяемым;
- быть лёгким в реализации и использовании;

# Синтаксические элементы

## Последовательности

```
--- # Список фильмов: последовательность в блочном формате
- Casablanca
- Spellbound
- Notorious
--- # Список покупок: последовательность в однострочном формате
[milk, bread, eggs, juice]
```

## Сопоставления имени и значения

```
--- # Блочный формат
name: John Smith
age: 33
--- # Однострочный формат
{name: John Smith, age: 33}
```



## Блочные литералы

### Переводы строк сохраняются

```
--- |
  There was a young fellow of Warwick
  Who had reason for feeling euphoric
    For he could, by election
    Have triune erection
  Ionic, Corinthian, and Doric
```

### Переводы строк исчезают

```
--- >
  Wrapped text
  will be folded
  into a single
  paragraph

  Blank lines denote
  paragraph breaks
```



## Последовательности из сопоставлений

```
- {name: John Smith, age: 33}  
- name: Mary Smith  
  age: 27
```

## Сопоставления из последовательностей

```
men: [John Smith, Bill Jones]  
women:  
  - Mary Smith  
  - Susan Williams
```