DOI: 10.15514/ISPRAS-2026-38(4)-3

Сравнительный анализ форматов XML и JSON в инфраструктуре современных научных коммуникаций: от документо-ориентированных стандартов к веб-ориентированным данным

О.И. Ефимова, ORCID 0009-0003-4503-057X <olgaefi2004@gmail.com>
Е.С. Оганесян, ORCID 0009-0006-4335-9824 <el.oganesyan@gmail.com>
П.А. Корсунская, ORCID 0009-0002-4412-8703 <polina_korsunskaya@mail.ru>

ФГБУ «Издательство «Наука»,
Россия, 121099, г. Москва Шубинский пер., д. 6, стр. 1.

Аннотация: В статье проводится сравнительный анализ форматов XML и JSON как ключевых стандартов для обмена и хранения научных данных. Актуальность исследования обусловлена ростом объемов научной информации и различиями между документо-ориентированными подходами (XML) и практиками веб-разработки (JSON). Цель работы – определить границы эффективного применения каждого формата в современных системах научных коммуникаций и оценить потенциал их интеграции. Методология включает анализ технических характеристик (синтаксис, типизация, метаданные), оценку производительности (объем данных, скорость анализа текста), сравнение инструментальных экосистем (XPath/JSONPath, XSD/JSON Schema, XSLT/jq), а также рассмотрение аспектов безопасности и конкретных примеров использования (JATS/TEI для XML; научные API и большие данные для JSON). Результаты представлены в виде критериев выбора формата для различных задач научных коммуникаций.

Ключевые слова: язык XML; язык JSON; научные коммуникации; JATS; TEI; JSON-LD; семантический веб; сравнительный анализ; большие данные; веб-API.

Для цитирования: Ефимова О.И., Оганесян Е.С., Корсунская П.А. Сравнительный анализ форматов XML и JSON в инфраструктуре современных научных коммуникаций: от документо-ориентированных стандартов к веб-ориентированным данным. Труды ИСП РАН, 2026, том 38, вып. 4, часть 1, стр. 49–68. DOI: 10.15514/ISPRAS-2026-38(4)-3.

Comparative Analysis of XML and JSON Formats in Modern Scientific Communication Infrastructure: From Document-Oriented Standards to Web-Oriented Data

O.I. Efimova, ORCID 0009-0003-4503-057X <olgaefi2004@gmail.com>
E.S. Oganesyan, ORCID 0009-0006-4335-9824 <el.oganesyan@gmail.com>
P.A. Korsunskaya, ORCID 0009-0002-4412-8703 <polina_korsunskaya@mail.ru>

Nauka Publishers,
6/1, Shubinskiy pereulok, Moscow, 121099, Russia.

Abstract: This article presents a comparative analysis of XML and JSON formats as fundamental standards for scientific data exchange and storage. The research relevance is determined by the increasing volumes of scientific information and the conceptual differences between document-oriented approaches (XML) and web development practices (JSON). The study aims to establish the boundaries of effective application for each format in modern scientific communication systems and assess their integration potential. The methodology encompasses analysis of technical characteristics (syntax, typing, metadata), performance evaluation (data volume, parsing speed), comparison of tool ecosystems (XPath/JSONPath, XSD/JSON Schema, XSLT/jq), along with examination of security aspects and specific use cases (JATS/TEI for XML; scientific APIs and big data for JSON). The research outcomes provide structured criteria for format selection tailored to various scientific communication requirements.

Keywords: XML, JSON, scientific communications, JATS, TEI, JSON-LD, semantic web, comparative analysis, big data, web API.

For citation: Efimova O.I., Oganesyan E.S., Korsunskaya P.A. Comparative Analysis of XML and JSON Formats in Modern Scientific Communication Infrastructure: From Document-Oriented Standards to Web-Oriented Data. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 4, part 1, pp. 49-68 (in Russian). DOI: 10.15514/ISPRAS-2026-38(4)-3.

1. Введение

Экспоненциальный рост объемов научных данных и развитие распределенных вычислительных систем актуализируют проблему выбора эффективных форматов для их структурированного представления, обмена и долговременного хранения. Оптимальный выбор формата данных является критическим фактором, определяющим скорость обработки информации, возможность ее автоматизированной интеграции и, как следствие, воспроизводимость научных исследований.

Исторически сложившимся стандартом для представления сложных документо-ориентированных данных в научной сфере является язык разметки XML (eXtensible Markup Language). Его выразительность, поддержка строгих онтологий, таких как JATS (Journal Article Tag Suite) и TEI (Text Encoding Initiative), а также механизмы валидации, например, XSD (XML Schema Definition), сделали его незаменимым инструментом в издательских системах, архивах (arXiv) и базах данных (PubMed). Язык XML обеспечивает описание семантической структуры документа, поддержку смешанного контента и сложных метаданных, что предопределило его использование в качестве формата для долговременного хранения и формального обмена.

Параллельно с развитием веб-технологий и микросервисных архитектур доминирующим форматом для обмена данными стал язык JSON (JavaScript Object Notation). Его ключевые преимущества – лаконичный синтаксис, высокая скорость синтаксического анализа (parsing), нативная поддержка в языках программирования и удобочитаемость человеком – определили его в качестве стандарта де-факто для интерфейса RESTful API (Representational State Transfer Application Programming Interface), конфигурационных файлов и систем реального времени.

Активное развитие семантического веба способствует дальнейшей популяризации JSON-ориентированных форматов, таких как JSON-LD (JSON for Linked Data) [1].

Возникает научная проблема, заключающаяся в наличии методологического разрыва между двумя подходами к организации данных: устоявшимися, но зачастую избыточными XML-стандартами научной коммуникации и легковесными, гибкими практиками веб-разработки (agile-ориентированными), основанными на JSON. Существующие сравнительные исследования, как правило, фокусируются на общих технических характеристиках форматов, но не дают комплексной оценки их применимости в конкретных научных контекстах, учитывающей всю совокупность критериев: от производительности и безопасности до зрелости инструментария и соответствия предметно-ориентированным требованиям.

Целью настоящего исследования является проведение комплексного сравнительного анализа форматов XML и JSON для определения границ их эффективного применения в современных системах научных публикаций и обмена данными. В задачи работы входит:

1. Сравнительная оценка технических характеристик (синтаксис, типизация, поддержка метаданных).
2. Анализ производительности (объем данных, скорость синтаксического анализа) на основе актуальных бенчмарков.
3. Сопоставление экосистем и инструментария (XPath/JSONPath, XSD/JSON Schema, XSLT/jq).
4. Анализ конкретных примеров использования (JATS/TEI для XML; научные прикладные интерфейсы и большие данные для JSON).
5. Оценка потенциала гибридных подходов (JSON-LD, встраивание форматов) и выявление тенденций развития в контексте эволюции веб-технологий.

Методология исследования основана на системном анализе, обобщении и сопоставлении данных из научной литературы, технической документации и практики применения. Результаты работы обобщены в виде сводных таблиц и выводов, формулирующих критерии выбора формата в зависимости от решаемых задач.

Проведенный анализ позволит разработчикам научного программного обеспечения, издателям и кураторам данных увидеть технологические решения, оптимизирующие инфраструктуру научных коммуникаций.

2. Технические характеристики XML и JSON

Основные характеристики языков и нотаций XML и JSON приведены в табл. 1.

2.1. XML (Extensible Markup Language)

XML – это язык разметки, который использует строгую иерархическую структуру, представляющую собой дерево, где каждый элемент имеет свое четкое место.

2.1.1 Синтаксические особенности

1. Теги и элементы. Основные структурные единицы XML – это элементы, которые обозначаются тегами в угловых скобках. Элемент состоит из открывающего и закрывающего тега, а содержимое находится между ними. Например, элемент <title> с содержимым "Программирование": <title>Программирование</title>. Одиночные (самозакрывающиеся) теги также имеют место быть, если они не заполнены. Например, отсутствует приложенный файл или информация об авторе: <file/> или <otherinfo/>.
2. Атрибуты. Это дополнительные сведения об элементе, которые указываются внутри открывающего тега. Часто используются для метаданных, которые не являются основным содержимым. Например, в элементе <author> атрибут num указывает на конкретного автора, если в статье их несколько: <author num="001">.
3. Иерархия. Элементы должны быть корректно вложены друг в друга, формируя четкую древовидную структуру. Нарушение этого правила (например, неправильное закрытие тегов) делает документ недействительным.

   Пример корректной иерархии XML:

```
<?xml version='1.0' standalone='no'?>
<journal>
<journalInfo lang="RUS">
        <title>Программирование</title>
    </journalInfo>
        </journal>
```

4. Стандартизация и валидация. Разработкой и поддержкой XML занимается консорциум W3C [2]. Консорциум выпустил спецификации XML 1.0 и 1.1, а также вспомогательные стандарты для формирования запросов (XPath) и преобразований (XSLT). Для проверки корректности и соответствия структуре данных используются схемы. Устаревшим и более простым способом описать структуру документа является DTD (Document Type Definition). XML Schema (XSD) – более выразительный язык для определения структуры, типов данных и правил. Пример XSD-схемы, которая требует, чтобы элемент article содержал внутри себя элемент title типа "строка" выглядит следующим образом:

```
<xs:element name="article">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Табл. 1. Сравнение XML и JSON по основным характеристикам.
Table 1. Comparison of XML and JSON based on main characteristics.

Параметр сравнения	XML	JSON
Базовый синтаксис	Теги с атрибутами (<author num="001">)	Пары «ключ-значение» { "author": {"num": "001"})
Типизация	Текстовая; типы задаются и проверяются через XSD-схемы	Встроенные базовые типы (число, строка, булев, null, массив, объект)
Метаданные	Атрибуты элементов (оптимально для служебной информации)	Нет отдельного механизма; передаются как обычные поля в объекте
Обработка	Требует сложных парсеров (DOM – для дерева в памяти, SAX – для потоковой обработки)	Простые и высокопроизводительные парсеры, встроенные в большинство языков
Размер данных	Обычно больший из-за избыточности тегов (закрывающие теги, атрибуты)	Компактный за счет лаконичного синтаксиса

2.1.2 Типизация

Основной тип данных – текст. Все данные, заключенные между тегами, изначально воспринимаются синтаксическим анализатором (парсером) как текст (строки). Чтобы работать с числами, датами и другими типами, необходимо выполнять явное преобразование строк.

Пример с датой: <date>2023-01-15</date> – это просто строка. Чтобы работать с ней как с датой, нужно вручную указать ее формат ("ГГГГ-ММ-ДД") и выполнить преобразование.

Пространства имен (Namespaces) в документе XML решают проблему конфликтов имен. Они позволяют использовать одинаковые имена тегов из разных словарей в одном документе, избегая неоднозначности. В одном документе можно совместить математическую разметку и обычные данные. Для этого математическим тегам присваивается префикс (например, math), который связывается с уникальным идентификатором (URL).

```
<root xmlns:math="http://www.w3.org/1998/Math/MathML">
    <math:formula>...</math:formula>
    <text>Обычный текст</text>
</root>
```

Словари можно комбинировать. Это позволяет создавать комплексные документы, объединяющие, например, текст, формулы и графику.

2.2. JSON (JavaScript Object Notation)

JSON – это лаконичный формат для обмена данными, основанный на структурах данных в JavaScript, но в настоящее время не зависящий от него [3].

2.2.1 Синтаксические особенности

1. Базовые структуры.
   - Пары «ключ-значение»: данные организованы в виде записей вида "ключ": значение. Например:

     {"surname": "Ivanov", "address": "Moscow"}
   - Массивы: упорядоченные списки значений, ограничиваются квадратными скобками. Например:

     ["Institute for the History of Science and Technology of Russian Academy of Sciences", "Institute of Gene Biology of the Russian Academy of Sciences", "Yerevan State University"]
   - Вложенность: значениями могут быть другие объекты или массивы, что позволяет строить сложные иерархии. Например:

     {"author": {"surname": "Ivanov", "orgName": ["Institute of Gene Biology of the Russian Academy of Sciences", "Moscow State University"]}}
2. Синтаксические правила:
   - Ключи всегда заключаются в двойные кавычки.
   - Запятая ставится только между элементами, но не после последнего элемента в объекте или массиве.
   - Поддерживаются управляющие последовательности (escape-последовательности) для специальных символов в строках (например, \n для переноса текста на новую строку).

2.2.2 Стандартизация

- ECMA-404 – официальный международный стандарт;

- RFC 8259 – техническая спецификация, опубликованная IETF (Инженерным советом Интернета);
- JSON Schema – популярный, хотя и не официальный стандарт для описания и валидации структуры JSON-документов.

Пример схемы, которая требует, чтобы корневой элемент был объектом, содержащим свойство "temperature" типа "число":

```
{ "$schema": "http://json-schema.org/draft-07/schema#",
    "type": "object",
    "properties": { "temperature": {"type": "number"} } }
```

2.2.3 Типы данных

JSON нативно поддерживает основные типы данных, что исключает необходимость их преобразования из строк:

- числа: 10, 3.14 (всегда с точкой в качестве разделителя);
- строки: "Hello World" (только в двойных кавычках);
- булевы значения: true или false;
- - специальное значение для обозначения пустоты: null.

2.2.4 Человекочитаемость

JSON часто проще для чтения и понимания человеком благодаря менее сложному синтаксису по сравнению с XML:

- меньше служебных символов: отсутствуют закрывающие теги, вместо них используются фигурные и квадратные скобки;
- более привычное отображение данных: структура напоминает объекты и массивы в большинстве языков программирования;
- удобство редактирования: JSON-файл проще создать или изменить вручную.

2.2.5 Наглядное сравнение объема и сложности

XML (обладает большей избыточностью):

```
<author num="001">
    <individInfo lang="ENG">
        <surname>Odegov</surname>
        <initials>D. O.</initials>
        <orgName>Institute of Gene Biology of the Russian Academy of Sciences
            </orgName>
        <address>Moscow, Russia</address>
        </individInfo>
    </author>
```

JSON (характеризуется меньшим объемом):

```
{ "author": {
    "num": "001",
    "individInfo": [ { "lang": "ENG",
        "surname": "Odegov",
        "initials": "D. O.",
        "orgName": "Institute of Gene Biology of the Russian Academy of Sciences",
        "address": "Moscow, Russia" } ]
    }
}
```

Как видно из примера, в JSON меньше служебных символов (например, отсутствуют закрывающие теги </surname>, </initials>). Компактный синтаксис обеспечивает меньший объем данных, а структура ближе к объектам в JavaScript/Python. Документ проще редактировать, вложенность данных визуально более очевидна.

2.2.6 Дополнительные особенности

- Отсутствие поддержки комментариев: спецификация JSON не позволяет добавлять комментарии (в отличие от XML). Это сделано для упрощения формата.
- Отсутствие атрибутов: в JSON нет аналога XML-атрибутов. Все данные, включая метаинформацию, должны быть переданы как отдельные пары «ключ-значение» внутри объекта.
- Расширение формата: JSON5 – расширение, добавляющее поддержку комментариев, необязательные кавычки для ключей и др. BSON (Binary JSON) – бинарный формат для хранения JSON-подобных документов, добавляющий дополнительные типы данных и оптимизированный для увеличения скорости обработки. JSON-LD – расширение для семантической разметки данных в сети, позволяющее создавать связанные данные.

3. Критерии сравнения

Эволюция веб-технологий и распределенных систем способствовала созданию множества форматов для обмена данными. Среди них XML и JSON остаются одними из наиболее популярных и широко используемых стандартов. Несмотря на общую цель – представление структурированных данных – их концептуальная основа, возможности и области применения существенно различаются [4, 5]. Условия и результаты тестирования показаны в табл. 2 и 3.

Табл. 2. Условия тестирования.
Table 2. Test conditions.

Параметр	Значение
Язык	Python 3.11, C++17
XML-парсеры	Python: lxml 4.9.2; C++: pugixml 1.14
JSON-парсеры	Python: orjson 3.9.10; C++: simdjson 3.6
Метрика	Медиана из 100 запусков (парсинг в DOM)

Табл. 3. Результаты тестирования.
Table 3. Test results.

Формат	Язык / Парсер	Время (мс)
XML	Python / lxml	2,4
JSON	Python / orjson	0,7
XML	C++ / pugixml	0,15
JSON	C++ / simdjson	0,08
Примечание. Источник данных для C++: бенчмарк из официального репозитория simdjson (см. [6], раздел Performance comparison).		

3.1. Производительность

3.1.1 Размер данных (критерий объема).

Объем передаваемых данных напрямую влияет на нагрузку на сеть и затраты на хранение. Анализ показывает, что JSON обладает преимуществом по компактности [7, 8].

Для идентичного набора данных представление в XML в среднем на 30-40% больше по объему из-за избыточности синтаксиса, требующего парных открывающих и закрывающих тегов. JSON использует более лаконичный синтаксис на основе пар «ключ-значение» и скобок ({}, []).

Пример сравнительного представления:

XML:
```
<author num="001">
    <individInfo lang="ENG">
        <surname>Pronozin</surname>
        <initials>A.Yu.</initials>
    </individInfo>
</author>
```

JSON
```
{   "author": {
        "num": "001",
        "individInfo": {
            "lang": "ENG",
            "surname": "Pronozin",
            "initials": "A.Yu."
        }
    }
}
```

3.1.2 Сравнение производительности

Для сравнения производительности выполнена трансформация реальной JATS XML-статьи (example article Xml.xml, 14 832 байт) в эквивалентный JSON (example article Json.json, 11 527 байт). JSON оказался компактнее на 22,3%.

Из данных табл. 3 видно, что JSON стабильно превосходит XML по скорости синтаксического анализа на обоих языках: на Python – в 3,4 раза, на C++ – в 1,9 раза.

3.2. Гибкость и расширяемость

3.2.1 Возможности XML

XML демонстрирует гибкость в работе со сложными и многоуровневыми структурами данных. Позволяет легко работать со смешанным контентом: комбинировать текст с другими типами данных в рамках одного элемента, что незаменимо в документо-ориентированных подходах. Ниже пример для одних и тех же данных в XML и JSON.

XML (допустим смешанный контент – тег <bold> внутри текста)

```
<article>
    <artTitle lang="ENG">TRANSFORMATION OF NONSTATIONARY NAVIER-STOKES EQUATIONS OF A VISCOUS COMPRESSIBLE FLUID UNDER AN ARBITRARY CONFORMAL MAPPING</artTitle>
    <abstract lang="ENG">It is shown that, <bold>the circulation of velocity and fluid flow</bold> on any closed or open contour are preserved under an arbitrary conformal mapping... The transformed unsteady Navier-Stokes, continuity and
```

```
        heat balance equations, which govern the aerodynamic parameters in the
        mapped region, are derived.</abstract>
        <author num="001">
            <individualInfo lang="ENG">
                <surname>Dynnikova</surname>
                <initials>G.Ya.</initials>
            </individualInfo>
        </author>
    </article>
```

JSON (смешанный контент не поддерживается; для чередования строк и элементов используются массивы)

```
{   "article": { "artTitle": {
                    "#text": "TRANSFORMATION OF NONSTATIONARY NAVIER-
                    STOKES EQUATIONS OF A VISCOUS COMPRESSIBLE FLUID
                    UNDER AN ARBITRARY CONFORMAL MAPPING",
                    "@lang": "ENG"
                },
                "abstract": {
                    "#text": "It is shown that, the circulation of velocity and fluid flow on
                    any closed or open contour are preserved... The transformed
                    unsteady Navier-Stokes, continuity and heat balance equations,
                    which govern the aerodynamic parameters in the mapped region,
                    are derived.",
                    "@lang": "ENG"
                }
        },
        "authors": [  { "author": { "@num": "001",
                                    "individualInfo": { "@lang": "ENG",
                                                        "surname": "Dynnikova",
                                                        "initials": "G.Ya."
                                    }
                        }
                      }
        ]
    }
```

В XML-фрагменте тег <bold> внутри текста абстракта является наглядным примером смешанного контента. В JSON это невозможно представить без потери семантики выделения, либо без создания сложной и нечитаемой структуры с массивами для чередующихся строк и элементов разметки.

Атрибуты XML (lang, num) в JSON приходится эмулировать с помощью специальных псевдополей, начинающихся с символа @ ("@lang", "@num"). Это распространенный, но не стандартизированный подход, который усложняет структуру.

Пример демонстрирует, что XML обладает большей выразительной способностью для разметки сложного текстового контента с метаданными, в то время как JSON для данной задачи становится более громоздким и менее интуитивным. JSON оптимален для четко структурированных данных прикладного интерфейса (API), например, для информации об авторе.

Механизмы расширения XML:

- Пространства имен (namespaces) позволяют комбинировать несколько словарей данных в одном документе.

- Модульность обеспечивается технологией XInclude.
- Специализированные применения реализованы в таких стандартах, как SVG (векторная графика), MathML (математические формулы), DocBook (техническая документация).

3.2.2 Возможности и ограничения JSON

JSON оптимизирован для простоты и эффективности в конкретных сценариях. Области применения: веб-API (RESTful сервисы), конфигурационные файлы, обмен данными между уровнями приложения.

Ограничения JSON – отсутствие поддержки атрибутов (метаданных для элементов), отсутствие комментариев (на уровне стандарта) и сложность представления смешанного контента – делают его менее пригодным для сложных документо-ориентированных задач.

Пример ограничения:

```
{   "text": "1. Introduction. As is known, the method of conformal mappings is widely
        used... In [6] conformal mapping was used for modeling of stationary plane-
        parallel flow... <bold>Transformed Equation:</bold> 3u'/3t + (u·∇')u' +
        (u^2/2)∇'J = -∇'p + ...",
    "formulas": {   "usual_equation": "3u/3t + (u·∇)u = -∇p + μ∇²u",
                    "transformed_equation": "3u'/3t + (u·∇')u' + (u^2/2)∇'J = -∇'p + ..." },
    "citations": "[1-5], [6], [7], [8], [9]"
}
```

Поле "text" содержит весь текст статьи как простую строку. Любая сложная разметка (например, <bold>Transformed Equation:</bold>) для выделения ключевого момента или специальные символы) не поддерживается на уровне формата JSON и будет отображена как простой текст, теряя семантическое значение. Математические формулы (поле "formulas") представлены как простые строки. JSON не имеет стандартного способа представления математических элементов. Ссылки на литературу (поле "citations") в тексте даны просто как строка; JSON не может выразить семантическую связь между этим текстом и массивом библиографических данных.

3.3 Поддержка инструментов и экосистема

3.3.1 Инструментарий XML

XML поддерживается набором стандартизированных инструментов (табл. 4):

- языки запросов: XPath (навигация по узлам дерева), XQuery (полноценный язык запросов) [9];
- трансформации: XSLT – язык преобразований, Тьюринг-полный;
- валидация: XML Schema (XSD), Schematron, RELAX NG.

Примеры:

Xpath
```
//author[individInfo/surname="Kosov"]/initials
```

Xquery
```
for $a in doc("article.xml")/article/authors/author
[individInfo/orgName[contains(., "System Dynamics")]]
return $a/individInfo/email
```

XSLT
```
<xsl:template match="artTitle">
    <h1 style="text-align: center;">
        <xsl:value-of select="."/>
    </h1>
</xsl:template>
```

3.3.2 Инструментарий JSON

Экосистема JSON развивалась вокруг веб-разработки и отличается практической ориентацией (табл. 4):

- языки запросов: JSONPath (аналог XPath), jq (утилитарный процессор для командной строки);
- валидация: JSON Schema;
- адресация: JSON Pointer (RFC 6901);
- семантическая разметка: JSON-LD;
- интеграция: нативная поддержка в браузерах и большинстве языков программирования (JavaScript, Python, Java и др.). Примеры:

JSONPath

```
$.authors[?(@.individInfo.surname == 'Kosov')].individInfo.email
```

jq (командная строка)

```
jq '.authors[] | select(.individInfo.orgName | contains("System Dynamics")) |
    .individInfo.email' article.json
```

JSON Pointer (RFC 6901)

```
/articles/authors/0/individInfo/email
```

Интеграция с языками программирования:

JavaScript

```
const article = JSON.parse('{"authors":[{"individInfo":{"surname":"Kosov"}}]}');
const email = article.authors[0].individInfo.email;
```

Python

```
import json
with open('article.json') as f:
    data = json.load(f)
    surname = data['authors'][0]['individInfo']['surname']
```

Табл. 4. Сравнение инструментария XML и JSON.
Table 4. Comparison of XML and JSON tooling.

Параметр сравнения	XML-инструменты	JSON-инструменты
Запросы	XPath, XQuery	JSONPath, jq
Трансформации	XSLT (Тьюринг-полный)	jq, Handlebars
Валидация	XSD, Schematron	JSON Schema
Производительность	Обычно ниже	Выше, оптимизированы в языках

4. Использование XML и JSON в научных публикациях и в работе с большими данными

Основные характеристики XML и JSON показаны в табл. 5.

4.1. Использование XML в научных публикациях: JATS и TEI

JATS и TEI представляют собой детализированные схемы разметки для структурирования текста. Они позволяют не только интерпретировать текст, но и извлекать его структуру и семантику каждого элемента.

Табл. 5. Применение XML и JSON.
Table 5. Using XML and JSON.

Параметр сравнения	XML	JSON
Стандартизация	Жесткие стандарты (JATS, CML*)	Гибкие схемы под задачу
Инструменты	Специализированные редакторы	Интеграция с веб-технологиями
Данные	Сложные структуры, метаданные	Массивы числовых данных
Производительность	Требует ресурсов	Оптимизирован для потоковой обработки
Обучение	Требует специфических знаний	Проще для новых исследователей
* CML (Chemical Markup Language) – химический язык разметки.		

JATS (Journal Article Tag Suite) – это стандартизированный формат сериализации для современных научных публикаций в журналах, обеспечивающий унификацию потока. Использование единой схемы для публикаций позволяет структурировать их на четко определенные логические разделы:

- <front> – выходные сведения: заголовок, авторы, аннотация, ключевые слова.
- <body> – основное содержание: введение, методы, результаты, обсуждение.
- <back> – заключительная часть: список литературы, приложения.

Благодаря такому подходу издатели могут автоматически агригировать статьи из разных источников и публиковать их на своих сайтах в единообразном виде, а программные агенты поисковых систем и баз данных (например, PubMed) могут извлекать семантику: идентифицировать фамилию автора, список литературы и т.д. Это обеспечивает точность поиска и возможность автоматической навигации по библиографическим ссылкам [10].

TEI (Text Encoding Initiative) – это исследовательский инструмент для работы с любыми текстами, особенно сложными и архивными, используемый для глубокого анализа и сохранения культурного наследия. Рукописи, письма, стихи, пьесы, исторические документы – все эти объекты имеют сложную структуру, которую TEI помогает описать. Исследователь самостоятельно выбирает необходимые элементы разметки для конкретного текста.

Примеры возможностей TEI:

- пометка исправлений в рукописи с помощью тегов <add> (добавление) и <del> (удаление);
- анализ персонажей через разметку имен тегом <persName> с последующей автоматической частной оценкой;
- описание повреждений в архивном документе с помощью тега <damage>.

TEI позволяет задавать тексту сложные вопросы (например, найти фрагмент текста, где один персонаж упоминается в диалогах с другим, или сравнить все правки, внесенные автором в черновик). Для филологов и историков такая разметка обеспечивает сохранение не только текста, но и всех его особенностей: пагинации, пометок на полях, особенностей шрифта и пр.

Ключевые преимущества XML для научных документов:

1. Структура и иерархия. XML точно передает вложенность и отношения между элементами. Очевидно, что <paragraph> находится внутри <section>, а <surname> – внутри <author>. Для программной обработки это представляет собой иерархическую древовидную модель.

2. Человекочитаемость. В отличие от бинарных форматов, XML-файл можно открыть в любом текстовом редакторе и интерпретировать содержимое без специальных средств. Теги обычно имеют осмысленные названия (<author>, <title>).
3. Отделение содержания от представления. XML описывает «что это» (заголовок, имя, цитата), а не «как это выглядит» (жирный, 14 пт, красный). Внешний вид задается отдельно (через CSS или XSLT). Это позволяет одному и тому же контенту выглядеть по-разному на сайте, в PDF и в мобильном приложении [11].
4. Машиночитаемость и автоматизация. Программа может выполнить однозначную выборку всех фамилий авторов в тысячах статей, потому что они помечены тегом <surname>. При работе с обычным текстом такой структурированный синтаксический анализ был бы затруднен.
5. Гибкость и расширяемость. Можно создавать собственные теги (TEI) или строго следовать готовому словарю (JATS). XML-схема (XSD) позволяет строго проверять соблюдение правил разметки.

4.2. Использование JSON в работе с большими данными

JSON обеспечивает непротиворечивую передачу данных, позволяя:

- включать в один файл различные виды данных, написанных на разных языках, таким образом, чтобы они корректно обрабатывались программным обеспечением;
- формировать машиночитаемые файлы при выдаче ответов на поисковые запросы через API;
- изменять конфигурацию сложных программ путем загрузки файла, содержащего необходимые параметры настроек;
- выполнять высокопроизводительную обработку больших массивов данных для аналитики.

В качестве примеров научных API, использующих JSON, можно назвать:

- NCBI API – предоставляет несколько API для программного доступа к базам данным и инструментам Национального центра биотехнологической информации США;
- NASA API – предоставляет доступ к исследованиям космоса, снимкам телескопов, данным о космических миссиях;
- arXiv API – позволяет получать метаданные и ссылки на препринты научных публикаций;
- OpenWeatherMap API – предоставляет доступ к архивным и прогнозным данным о погоде;
- UniProt REST API – предоставляет данные по белковым последовательностям.

Рассмотрим более подробно NCBI API. NCBI представляет собой цифровую библиотеку биомедицинской и генетической информации, включая базы данных белковых доменов, ДНК (GenBank) и РНК, научные публикации (PubMed) и таксономичную информацию (TaxBrowser). Например, для решения задачи по исследованию 5000 генов реализуется программный модуль (скрипт), который автоматически собирает данные через API NCBI и возвращает ответы в формате JSON. Программа формирует HTTP-запрос, NCBI обрабатывает его, находит данные на своих серверах и возвращает ответ. Полученный JSON легко десериализуется [12], и скрипт на Python или R извлекает необходимые элементы: название гена, последовательность ДНК, хромосому, ссылки на статьи и т.д.

5. Гибридные подходы к обоим форматам

Существует несколько способов интеграции форматов.

5.1 Вложение

При организации вложений один формат помещается внутрь другого в виде строки [13].

5.1.1 JSON внутри XML

Этот способ часто распространен для описания метаданных. Он используется, когда к статье в JATS XML необходимо добавить сложные, не жестко регламентированные метаданные для веб-системы (например, рейтинг статьи, дополнительные данные). В XML-файл статьи добавляется блок, внутри которого JSON хранится как обычный текст.

```
<article xmlns="http://jats.nlm.nih.gov">
    <front>
        <article-meta>
            <title>Новое открытие в физике</title>
            <abstract>...</abstract>
            <!-- Стандартные метаданные JATS -->
            <custom-meta>
                <meta-name>
                    web_metadata
                </meta-name>
                <meta-value>
                    <![CDATA[
                        {   "article_rating": 4.8,
                            "downloads_count": 12450,
                            "related_articles": ["doi:12345", "doi:67890"],
                            "semantic_tags": ["квантовая_телепортация", "кубит"] }
                    ]]>
                </meta-value>
            </custom-meta>
        </article-meta>
    </front>
    <body>...</body>
</article>
```

В результате стандартная JATS-структура не нарушается, но статья дополняется метаданными, необходимыми на платформе. Веб-система сначала извлекает эту строку из JATS, а затем десериализует данные из JSON.

5.1.2 XML внутри JSON

Этот способ применяется для сложных структур. Он используется, например, при разработке веб-API для научного архива. В JSON-файл в поле "article_xml" добавляется вся статья в виде XML-строки.

```
{   "search_query": "квантовая телепортация",
    "total_results": 42,
    "articles":  [ {  "id": "12345",              "title": "Новое открытие в физике",
                      "doi": "10.1000/12345",   "published_date": "2023-10-26",
                      "article_xml": "<article xmlns=\"http://jats.nlm.nih.gov\">
                          <front>...<title>Новое открытие в физике</title>...</front>
                          <body>...</body></article>"
                   } ]
}
```

В итоге API передает результаты поиска в формате JSON, а затем данные извлекаются из XML-строки. В качестве примера API, который предоставляет результаты поиска в виде метаданных статей в формате JSON, а полный текст – в JATS XML, можно привести PubMed Central (архив полнотекстовых биомедицинских публикаций со свободным доступом см. в [14]).

5.2 Преобразование

Данные из одного формата преобразуются в другой, сохраняя структуру и содержание.

5.2.1 XML (JATS) → JSON

Преобразование выполняется, например, для оптимизации обработки научных публикаций современными инструментами Data Science (Python, Pandas, JavaScript) путем создания правил (часто через XSLT или специальный парсер), которые отображают XML-элементы на JSON-объекты. Исходный JATS XML выглядит следующим образом:

```
<article>
    <front>
        <article-meta>
            <title-group><article-title>Заголовок статьи</article-title></title-group>
            <abstract><p>Аннотация статьи...</p></abstract>
        </article-meta>
    </front>
</article>
```

Результирующий JSON:

```
{ "article": {
    "front": {
        "article-meta":   { "title": "Заголовок статьи",
                            "abstract": "Аннотация статьи..." }
        }
    }
}
```

Таким образом значительно упрощается создание аналитических скриптов: вместо работы с DOM-парсером и XPath-запросами достаточно обратиться к полю article['front']['article-meta']['title'] на Python.

5.2.2 JSON → XML (JATS)

Преобразование выполняется, например, для создания формы подачи публикации автором через веб-интерфейс. Данные из формы преобразуются в JSON, а затем их необходимо преобразовать в валидный JATS XML для использования в публикационном процессе. Создается шаблон или скрипт, который извлекает данные из JSON и преобразует их в JATS XML. Входной JSON (из веб-формы) выглядит следующим образом:

```
{   "articleData": {
        "title": "Мои результаты",
        "authors": ["Иванов И.И.", "Петров П.П."],
        "abstractText": "Текст аннотации...",
        "bodySections": [ {"type": "methods", "content": "Текст методов..."},
                          {"type": "results", "content": "Текст результатов..."} ]
    }
}
```


Выходной XML (JATS):

```
<article>
    <front>
        <article-meta>
            <title-group><article-title>Мои результаты</article-title></title-group>
            <contrib-group>
            <contrib><name>Иванов И.И.</name></contrib>
                <contrib><name>Петров П.П.</name></contrib>
            </contrib-group>
            <abstract><p>Текст аннотации...</p></abstract>
        </article-meta>
    </front>
    <body>
        <sec sec-type="methods"><p>Текст методов...</p></sec>
        <sec sec-type="results"><p>Текст результатов...</p></sec>
    </body>
</article>
```

Такой подход позволяет автоматизировать процесс приема статей от авторов и их первоначальной подготовки к публикации.

5.3. Семантическая связь (JSON-LD внутри XML)

Данный способ используется, когда требуется, чтобы данные из публикации были машиночитаемыми и связанными с другими данными в семантическом вебе. В секцию <custom-meta> JATS XML внедряется JSON-LD скрипт, который описывает семантику публикации: автора, его аффилиацию, тематику:

```
<custom-meta>
    <meta-name>json-ld</meta-name>
    <meta-value>
        <![CDATA[
            {   "@context": "https://schema.org",
                "@type": "ScholarlyArticle",
                "headline": "Заголовок статьи",
                "author": {
                    "@type": "Person",
                    "name": "Иванов И.И.",
                    "affiliation": { "@type": "Organization", "name": "Университет" }
                },
                "description": "Аннотация статьи..."
            } ]]>
    </meta-value>
</custom-meta>
```

Посредством этого скрипта поисковые системы и научные агрегаторы могут извлекать эту структурированную информацию и использовать ее для улучшения поиска, построения графов знаний (базы данных, хранящих информацию в виде связей между сущностями) и создания расширенных описаний, отображаемых в поисковой выдаче.

Таким образом, интеграция XML и JSON представляет собой инструментарий для построения гибких, мощных и современных систем научных коммуникаций (табл. 6).

Табл. 6. Способы интеграции XML и JSON.
Table 6. Methods of XML and JSON integration.

Наименование способа	Содержание способа	Область применения
Вложение	Хранение одного формата как строки внутри другого	Добавление гибких метаданных к жесткой структуре
Преобразование	Конвертация данных из одного формата в другой	Интеграция с современными веб-API или анализаторами
Семантическая связь	Обогащение публикаций машиночитаемыми метаданными (JSON-LD)	Повышение видимости публикаций в семантическом вебе

6. Тенденции развития и применения XML и JSON для научных публикаций

Отметим несколько ключевых аспектов, связанных с применением форматов и спецификой их использования:

1. Доминирование гибридных архитектур («XML внутри, JSON снаружи») [15]. XML (JATS, TEI) используется для долгосрочного хранения и архивации благодаря строгости и поддержке валидации, а JSON – для обмена данными через API, веб-интерфейсы и потоковую передачу метаданных. На практике это выглядит следующим образом: издательские платформы хранят статьи в JATS XML, но все RESTful API для поиска и получения метаданных возвращают данные в JSON. Это сочетает надежность XML с легковесностью и скоростью разработки на JSON.
2. Рост значения семантической разметки и связанных данных. Акцент смещается с простого обмена данными на обеспечение их интеграции в глобальное информационное пространство (семантический веб). Для внедрения семантики в веб-среду используют стандарт JSON-LD [16]. Научные журналы все чаще размещают JSON-LD на своих веб-страницах, что повышает их видимость в поисковых системах через расширенные описания, отображаемые в поисковой выдаче, и позволяет интегрироваться в глобальные графы знаний, такие как Google Knowledge Graph.
3. Специализация форматов. Если ранее было принято использовать один формат для разных задач, то в настоящее время форматы выбираются под конкретную задачу: XML – для описания сложных, иерархических документов и в областях, где критически важна валидация (например, в юриспруденции или цифровых архивах); JSON – для веб-API, микросервисов, конфигурационных файлов и передачи данных от сенсоров, где важна высокая скорость разработки. Бинарные форматы (Apache Parquet, Avro) применяют для высокопроизводительной аналитики больших данных, где текстовые форматы (XML/JSON) оказываются слишком медленными и избыточными по объему.
4. Развитие инструментария для интеграции форматов. Распространение гибридных моделей стимулирует потребность в эффективных инструментах конвертации. Активно развиваются библиотеки и стандарты для преобразования XML в JSON и обратно (например, XSLT 3.0, библиотеки xml2js в Node.js), а также появляются стандартизированные методики такого преобразования.
5. Усиление роли валидации в JSON-экосистеме. JSON развивается до уровня XML в вопросах обеспечения качества данных. JSON Schema для описания структуры, валидации и документирования API-ответов стала промышленным стандартом в критически важных приложениях, заимствуя концепцию XSD из XML.
6. Оптимизация для машинной обработки. Оба формата все чаще ориентируются на машиночитаемые системы, а не на чтение человеком. Это проявляется в растущей популярности бинарных представлений, таких как EXI (для XML) или BSON и MessagePack (для JSON), которые снижают нагрузку на сеть и ускоряют синтаксический анализ.
7. Конвергенция в области запросов. Стираются границы между языками запросов к разным форматам. Появляются универсальные инструменты, позволяющие единообразно запрашивать данные. Например, язык JSONiq позволяет выполнять запросы к JSON, используя семантику, сходную с XQuery для XML.

7. Заключение

Проведенный анализ современных подходов к разметке и обмену научными данными демонстрирует отсутствие необходимости противопоставлять XML и JSON. Эти технологии не столько конкурируют, сколько эффективно дополняют друг друга, формируя гибридную экосистему, которая наилучшим образом отвечает разнородным задачам научной коммуникации.

Ключевые выводы работы заключаются в следующем:

1. Выбор между XML и JSON диктуется не техническим превосходством одного формата над другим, а конкретным этапом жизненного цикла научной публикации и целевой аудиторией, для которой предназначены данные.
2. XML (JATS, TEI) продолжает использоваться для долгосрочного, надежного хранения и осуществления публикационного процесса. Его строгая схематическая валидация, богатая семантика и развитая экосистема инструментов для преобразования, обеспечения целостности и сохранности научного контента делают его незаменимым для издателей.
3. JSON утвердился как универсальный язык данных для веб-ориентированных приложений, API и оперативного анализа. Его легковесность, скорость обработки и нативная поддержка в современных языках программирования делают его идеальным форматом для построения гибких API, быстрой передачи метаданных, интеграции научных данных в интерактивные веб-платформы и информационные панели.
4. Наиболее перспективным направлением является не выбор одного формата, а их интеграция. Методы вложения JSON в XML-документы для обогащения метаданными и преобразования между форматами на разных этапах публикационного процесса позволяют создать гибкую и эффективную архитектуру. В такой системе XML выступает как источник контента, а JSON – как инструмент для оперативной доставки и использования этих данных в цифровой среде.
5. Появление JSON-LD стало эффективным решением интеграции двух форматов. Встраивание машиночитаемых метаданных в формате JSON-LD непосредственно в XML-разметку статьи позволяет одновременно соблюдать издательские стандарты и соответствовать требованиям семантического веба, значительно повышая видимость научных результатов.

Таким образом, современная инфраструктура научных знаний строится не на замене устоявшихся стандартов, а на их интеграции с новыми технологиями. Будущее за гибридными моделями, где JATS XML обеспечивает надежность и долговечность, а JSON – скорость, гибкость и интеграцию с различными системами.

Список литературы / References

[1]. JSON-LD 1.1. A JSON-based Serialization for Linked Data. W3C Recommendation, 2020. Available at: https://www.w3.org/TR/json-ld11/, accessed 18.04.2026.

[2]. XML Essentials. W3C. Available at: http://www.w3.org/standards/xml/core, accessed 18.04.2026.

[3]. JSON and XML Comparison. Available at: https://www.json.org/json-en.html, accessed 18.04.2026.

[4]. Difference Between JSON and XML. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/difference-between-json-and-xml/, accessed 18.04.2026.

[5]. What is JSON? Amazon Web Services. Available at: https://aws.amazon.com/ru/documentdb/what-is-json/, accessed 18.04.2026.

[6]. simdjson: Parsing gigabytes of JSON per second. Available at: https://github.com/simdjson/simdjson, accessed 04.06.2026.

[7]. Rianto R., Rifansyah M., Gunawan R., Darmawan I., Rahmatulloh A. Comparison of JSON and XML Data Formats in Document Stored NoSql Database Replication Processes. International Journal on Advanced Science Engineering and Information Technology. 11, 2021, pp. 1150-1156. DOI: 10.18517/ijaseit.11.3.11570.

[8]. Канаев К.А., Фалеева Е.В., Пономарчук Ю.В. Сравнительный анализ форматов обмена данными, используемых в приложениях с клиент-серверной архитектурой. Фундаментальные исследования, 2-25, 2015, с. 5569-5572. Доступно по ссылке: https://fundamental-research.ru/ru/article/view?id=38464, обращение 18.04.2026. / Kanaev K.A., Faleeva E.V., Ponomarchuk Y.V. Comparative analysis of data exchange formats for applications with client-server architecture.Fundamental research, 2-25, 2015, c. 5569-5572. (In Russ.). Available at: https://fundamental-research.ru/ru/article/view?id=38464, accessed 18.04.2026.

[9]. XLXP, XSLT, XPath, XForms & XQuery Interview Questions You'll Most Likely Be Asked. Vibrant Publisher, CreateSpace Independent Publishing Platform, 2012. Available at: https://archive.org/details/xlxpxsltxpathxfo0000unse, accessed 18.04.2026.

[10]. Сон Хо, Тхэ Джин Цой, Со Хён Ким. Применение XML-разметки для научных статей, опубликованных в журналах на корейском языке. Доступно по ссылке: https://cyberleninka.ru/article/n/primenenie-xml-razmetki-dlya-nauchnyh-statey-opublikovannyh-v-zhurnalah-na-koreyskom-yazyke, обращение 18.04.2026. / Sun Huh, Tae Jin Choi, So-Hyeong Kim. Using Journal Article Tag Suite extensible markup language for scholarly journal articles written in Korean. Science editing, 1(1), 2014, pp. 19-23. DOI: 10.6087/kcse.2014.1.19.

[11]. Skornyakova R.Yu. An Approach to Creating an HTML Version of a Scientific Article from a Manuscript in MS Word Format for a Low-Budget Publisher. Russian Digital Libraries Journal 27(6), 2024, pp. 1064-1089. DOI: 10.26907/1562-5419-2024-27-6-1064-1089.

[12]. Шульман В.Д., Шабанов В.В., Сухов П.А., Чунихин А.О. Сравнительный анализ форматов сериализации и передачи данных JSON, XML, CBOR и GPB. Научно-образовательный журнал для студентов и преподавателей StudNet, 7, 2023, с. 1686-1697. Доступно по ссылке: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-formatov-serializatsii-i-peredachi-dannyh-json-xml-cbor-i-gpb/viewer, обращение 18.04.2026. / Shulman V., Shabanov V., Sukhov P., Chunikhin A. Comparatie analysis of JSON, XML, CBOR and GPB serialization and data transfer formats. StudNet, 7, 2023, pp. 1686-1697. Available at: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-formatov-serializatsii-i-peredachi-dannyh-json-xml-cbor-i-gpb/viewer, accessed 18.04.2026.

[13]. Мешков В.В., Ухарев Т.В. Связывание объектов в XML. Научная сессия МИФИ-2000, т. 3: Банки данных и анализ данных. Интеллектуальные системы и технологии. Технологии разработки программных систем, с. 20. Доступно по ссылке: https://scipeople.ru/publication/55701/, обращение 18.04.2026. / Meshkov V. V., Ukharov T. V. Object binding in XML. Scientific session of MEPHI-2000, vol. 3: Data banks and data analysis. Intellectual systems and technologies. Technologies for the development of software systems, p. 20. Available at: https://scipeople.ru/publication/55701/, accessed 18.04.2026.

[14]. U.S. National Institutes of Health's National Library of Medicine (NIH/NLM). PubMed Central (PMC). Available at: https://www.ncbi.nlm.nih.gov/pmc/, accessed 04.06.2026.

[15]. Bray T. (Ed.). The JavaScript Object Notation (JSON) Data Interchange Standard. RFC 8259. IETF, 2017. Available at: https://www.rfc-editor.org/rfc/rfc8259.html, accessed 18.04.2026.

[16]. Schema.org. Available at: https://schema.org/, accessed 18.04.2026.

Информация об авторах / Information about authors

Ольга Игоревна ЕФИМОВА – независимый исследователь, специалист по автоматизации издательских процессов ФГБУ издательства "НАУКА". Сфера научных интересов: применение нейросетей для парсинга и структурирования метаданных, автоматизация рабочих процессов на Python и Java, прикладная аналитика и архитектура информационных систем.

Olga Igorevna EFIMOVA– independent researcher, specialist in publishing process automation at the "NAUKA" Publishing House. Research interests: application of neural networks for parsing and structuring metadata, workflow automation with Python and Java, applied analytics, and information systems architecture.

Елена Сергеевна ОГАНЕСЯН – независимый исследователь, специалист по автоматизации издательских процессов ФГБУ издательства "НАУКА". Сфера научных интересов: применение нейросетей для парсинга и структурирования метаданных, обработка изображений, машинное обучение.

Elena Sergeevna OGANESYAN – independent researcher, specialist in publishing process automation at the "NAUKA" Publishing House. Research interests: application of neural networks for parsing and structuring metadata, image processing, machine learning.

Полина Александровна КОРСУНСКАЯ – научный сотрудник, руководитель группы организационно-правового обеспечения научно-издательской деятельности ИМЭМО РАН. Сфера научных интересов: цифровизация издательских процессов, архитектура издательских информационных систем.

Polina Aleksandrovna KORSUNSKAYA – Researcher, Head of the Group for Organizational and Legal Support of Scientific Publishing Activities of IMEMO. Research interests: digitalization of publishing processes, architecture of publishing information systems.