

System.Text.Json Пространство имен

[Ссылка](#)

Предоставляет высокопроизводительные, не требовательные к памяти и соответствующие стандартам возможности обработки JSON, включая сериализацию объектов в текст JSON и десериализацию текста JSON в объекты за счет встроенной поддержки UTF-8. Оно также предоставляет типы для чтения и записи текста JSON в кодировке UTF-8 и для создания модели DOM в памяти для произвольного доступа к элементам JSON в структурированном представлении данных.

Классы

JsonDocument	Предоставляет механизм для проверки структурного содержимого значения JSON без автоматического создания экземпляров для значений данных.
JsonException	Определяет пользовательский объект исключения, который возникает при обнаружении недопустимого текста JSON, передаче заданной максимальной глубины или при несовместимом тексте JSON с типом свойства объекта.
JsonNamingPolicy	Определяет политику именования, используемую для преобразования строкового имени в другой формат, например Camel.
JsonSerializer	Предоставляет функциональные возможности сериализации объектов или типов значений в JSON и десериализации JSON в объекты или типы значений.
JsonSerializerOptions	Предоставляет параметры для использования с JsonSerializer .
Utf8JsonWriter	Предоставляет высокопроизводительный API для однонаправленной некэшированной записи текста JSON в кодировке UTF-8.

Структуры

JsonDocumentOptions	Предоставляет пользователю возможность определить пользовательское поведение при анализе JSON для создания JsonDocument .
JsonElement	Представляет определенное значение JSON в JsonDocument .

JsonElement.ArrayEnumerator	Представляет перечислитель для содержимого массива JSON.
JsonElement.Object Enumerator	Представляет перечислитель для свойств объекта JSON.
JsonEncodedText	Предоставляет методы для преобразования текста в кодировке UTF-8 или UTF-16 в форму, подходящую для JSON.
JsonProperty	Представляет одно свойство объекта JSON.
JsonReaderOptions	Предоставляет пользователю возможность определить пользовательское поведение при чтении JSON.
JsonReaderState	Определяет непрозрачный тип, содержащий и сохраняющий все соответствующие сведения о состоянии, которые необходимо предоставить Utf8JsonReader для продолжения чтения после обработки неполных данных.
JsonWriterOptions	Разрешает пользователю определить пользовательское поведение при чтении JSON с помощью Utf8JsonWriter .
Utf8JsonReader	Предоставляет высокопроизводительный API для одностороннего доступа только для чтения к тексту JSON в кодировке UTF-8.

Перечисления

JsonCommentHandling	Определяет, как структура Utf8JsonReader обрабатывает комментарии.
JsonSerializerDefaults	Задает параметры сериализации по умолчанию на основе сценариев, которые можно использовать для создания экземпляра JsonSerializerOptions .
JsonTokenType	Определяет различные токены JSON, составляющие текст JSON.
JsonValueKind	Задает тип данных значения JSON.

Комментарии

См. также [System.Text.Json.Serialization](#) пространство имен, содержащее атрибуты и API для расширенных сценариев и настройки, относящихся к сериализации и десериализации.

Дополнительные сведения см . в обзоре [System.Text.Json](#).

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonCommentHandling Перечисление

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Определяет, как структура [Utf8JsonReader](#) обрабатывает комментарии.

C#

```
public enum JsonCommentHandling
```

Наследование [Object](#) → [ValueType](#) → [Enum](#) → [JsonCommentHandling](#)

Поля

Allow	2	Разрешает комментарии внутри входных данных JSON и обрабатывает их как допустимые токены. При чтении вызывающий объект может обратиться к значениям комментариев.
Disallow	0	Запрещает комментарии внутри входных данных JSON. При обнаружении комментарии обрабатываются как недопустимая JSON, и возникает исключение JsonException . Это значение по умолчанию.
Skip	1	Разрешает комментарии внутри входных данных JSON и игнорирует их. Utf8JsonReader работает так, будто комментарии отсутствуют.

Комментарии

Дополнительные сведения см. в статье [Как разрешить некоторые типы недопустимых json с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocument Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет механизм для проверки структурного содержимого значения JSON без автоматического создания экземпляров для значений данных.

C#

```
public sealed class JsonDocument : IDisposable
```

Наследование [Object](#) → JsonDocument

Реализации [IDisposable](#)

Комментарии

Этот класс использует ресурсы из памяти в пуле, чтобы свести к минимуму влияние сборщика мусора (GC) в сценариях с высоким уровнем использования. Сбой правильного удаления этого объекта приведет к тому, что память не будет возвращена в пул, что приведет к увеличению влияния сборки мусора в различных частях платформы.

Дополнительные сведения см. в статье [Использование JSON DOM, Utf8JsonReader и Utf8JsonWriter в System.Text.Json](#).

Свойства

[RootElement](#)

Возвращает корневой элемент этого документа JSON.

Методы

[Dispose\(\)](#)

Освобождает ресурсы, используемые этим экземпляром

	JsonDocument.
Equals(Object)	Определяет, равен ли указанный объект текущему объекту. (Унаследовано от Object)
GetHashCode()	Служит хэш-функцией по умолчанию. (Унаследовано от Object)
GetType()	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object)
MemberwiseClone()	Создает неполную копию текущего объекта Object . (Унаследовано от Object)
Parse(ReadOnly Memory<Byte>, Json DocumentOptions)	Анализирует память как текст в кодировке UTF-8, представляющий одно значение JSON, в JsonDocument .
Parse(ReadOnly Memory<Char>, Json DocumentOptions)	Анализирует текст, представляющий одно значение JSON, в jsonDocument .
Parse(ReadOnly Sequence<Byte>, Json DocumentOptions)	Анализирует последовательность как текст в кодировке UTF-8, представляющий одно значение JSON в jsonDocument .
Parse(Stream, JsonDocument Options)	Анализирует Stream в виде данных в кодировке UTF-8, представляющих отдельное значение JSON в JsonDocument . Поток считывается до завершения.
Parse(String, JsonDocument Options)	Анализирует текст, представляющий отдельное строковое значение JSON в JsonDocument .
ParseAsync(Stream, Json DocumentOptions, CancellationToken)	Анализирует Stream в виде данных в кодировке UTF-8, представляющих отдельное значение JSON в JsonDocument . Поток считывается до завершения.
ParseValue(Utf8JsonReader)	Анализирует одно значение JSON (включая объекты или массивы) из указанного модуля чтения.
ToString()	Возвращает строку, представляющую текущий объект. (Унаследовано от Object)
TryParseValue(Utf8JsonReader, JsonDocument)	Пытается проанализировать одно значение JSON (включая объекты или массивы) из указанного модуля чтения.
WriteTo(Utf8JsonWriter)	Записывает документ в предоставленный модуль записи в виде значения JSON.

Методы расширения

Deserialize(JsonDocument, JsonTypeInfo)	Преобразует объект , JsonDocument представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(JsonDocument, Type, JsonSerializerOptions)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonDocument, Type, JsonSerializerContext)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>returnType</code> в .
Deserialize< TValue >(Json Document, JsonSerializer Options)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Document, JsonType Info< TValue >)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>TValue</code> в .

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocument.RootElement Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the root element of this JSON document.

C#

```
public System.Text.Json.JsonElement RootElement { get; }
```

Property Value

[JsonElement](#)

A [JsonElement](#) representing the value of the document.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonDocument.Dispose Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Освобождает ресурсы, используемые этим экземпляром [JsonDocument](#).

C#

```
public void Dispose();
```

Реализации

[Dispose\(\)](#)

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocument.Parse Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

Parse(ReadOnlySequence<Byte>, JsonDocumentOptions)	Parses a sequence as UTF-8-encoded text representing a single JSON value into a JsonDocument.
Parse(Stream, JsonDocumentOptions)	Parses a Stream as UTF-8-encoded data representing a single JSON value into a JsonDocument. The stream is read to completion.
Parse(ReadOnlyMemory<Byte>, JsonDocumentOptions)	Parses memory as UTF-8-encoded text representing a single JSON value into a JsonDocument.
Parse(ReadOnlyMemory<Char>, JsonDocumentOptions)	Parses text representing a single JSON value into a JsonDocument.
Parse(String, JsonDocumentOptions)	Parses text representing a single JSON string value into a JsonDocument.

Parse(ReadOnlySequence<Byte>, JsonDocumentOptions)

Parses a sequence as UTF-8-encoded text representing a single JSON value into a JsonDocument.

C#

```
public static System.Text.Json.JsonDocument Parse  
(System.Buffers.ReadOnlySequence<byte> utf8Json,  
System.Text.Json.JsonDocumentOptions options = default);
```

Parameters

utf8Json [ReadOnlySequence<Byte>](#)

The JSON text to parse.

options [JsonDocumentOptions](#)

Options to control the reader behavior during parsing.

Returns

[JsonDocument](#)

A JsonDocument representation of the JSON value.

Exceptions

[JsonException](#)

`utf8Json` does not represent a valid single JSON value.

[ArgumentException](#)

`options` contains unsupported options.

Remarks

The [ReadOnlySequence<T>](#) may be used for the entire lifetime of the JsonDocument object, and the caller must ensure that the data therein does not change during the object lifetime. Because the input is considered to be text, a UTF-8 Byte-Order-Mark (BOM) must not be present.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Parse(Stream, JsonDocumentOptions)

Parses a [Stream](#) as UTF-8-encoded data representing a single JSON value into a JsonDocument. The stream is read to completion.

C#

```
public static System.Text.Json.JsonDocument Parse (System.IO.Stream  
utf8Json, System.Text.Json.JsonDocumentOptions options = default);
```

Parameters

utf8Json [Stream](#)

The JSON data to parse.

options [JsonDocumentOptions](#)

Options to control the reader behavior during parsing.

Returns

[JsonDocument](#)

A JsonDocument representation of the JSON value.

Exceptions

[JsonException](#)

`utf8Json` does not represent a valid single JSON value.

[ArgumentException](#)

`options` contains unsupported options.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Parse(ReadOnlyMemory<Byte>, JsonDocumentOptions)

Parses memory as UTF-8-encoded text representing a single JSON value into a JsonDocument.

C#

```
public static System.Text.Json.JsonDocument Parse (ReadOnlyMemory<byte>
utf8Json, System.Text.Json.JsonDocumentOptions options = default);
```

Parameters

utf8Json `ReadOnlyMemory<Byte>`

The JSON text to parse.

options `JsonDocumentOptions`

Options to control the reader behavior during parsing.

Returns

`JsonDocument`

A `JsonDocument` representation of the JSON value.

Exceptions

`JsonException`

`utf8Json` does not represent a valid single JSON value.

`ArgumentException`

`options` contains unsupported options.

Remarks

The `ReadOnlyMemory<T>` value will be used for the entire lifetime of the `JsonDocument` object, and the caller must ensure that the data therein does not change during the object lifetime.

Because the input is considered to be text, a UTF-8 Byte-Order-Mark (BOM) must not be present.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Parse(ReadOnlyMemory<Char>, JsonDocumentOptions)

Parses text representing a single JSON value into a `JsonDocument`.

C#

```
public static System.Text.Json.JsonDocument Parse (ReadOnlyMemory<char>
    json, System.Text.Json.JsonDocumentOptions options = default);
```

Parameters

json `ReadOnlyMemory<Char>`

The JSON text to parse.

options `JsonDocumentOptions`

Options to control the reader behavior during parsing.

Returns

`JsonDocument`

A `JsonDocument` representation of the JSON value.

Exceptions

`JsonException`

`json` does not represent a valid single JSON value.

`ArgumentException`

`options` contains unsupported options.

Remarks

The `ReadOnlyMemory<T>` value may be used for the entire lifetime of the `JsonDocument` object, and the caller must ensure that the data therein does not change during the object lifetime.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Parse(String, JsonDocumentOptions)

Parses text representing a single JSON string value into a `JsonDocument`.

C#

```
public static System.Text.Json.JsonDocument Parse (string json,  
System.Text.Json.JsonDocumentOptions options = default);
```

Parameters

json `String`

The JSON text to parse.

options `JsonDocumentOptions`

Options to control the reader behavior during parsing.

Returns

`JsonDocument`

A `JsonDocument` representation of the JSON value.

Exceptions

`JsonException`

`json` does not represent a valid single JSON value.

`ArgumentException`

`options` contains unsupported options.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonDocument.ParseAsync Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует [Stream](#) в виде данных в кодировке UTF-8, представляющих отдельное значение JSON в JsonDocument. Поток считывается до завершения.

C#

```
public static System.Threading.Tasks.Task<System.Text.Json.JsonDocument>
ParseAsync (System.IO.Stream utf8Json, System.Text.Json.JsonDocumentOptions
options = default, System.Threading.CancellationToken cancellationToken =
default);
```

Параметры

utf8Json [Stream](#)

Анализируемые данные JSON.

options [JsonDocumentOptions](#)

Параметры для управления поведением модуля чтения во время анализа.

cancellationToken [CancellationToken](#)

Токен для отслеживания запросов отмены.

Возвращаемое значение

[Task<JsonDocument>](#)

Задача по созданию представления JsonDocument данного значения JSON.

Исключения

[JsonException](#)

utf8Json не представляет допустимое отдельное значение JSON.

[ArgumentException](#)

`options` содержит неподдерживаемые параметры.

OperationCanceledException

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как [ArgumentException](#), по-прежнему создаются синхронно. Хранимые исключения см. в разделе исключения, создаваемые [Parse\(Stream, JsonDocumentOptions\)](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocument.ParseValue(Utf8JsonReader) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует одно значение JSON (включая объекты или массивы) из указанного модуля чтения.

C#

```
public static System.Text.Json.JsonDocument ParseValue (ref  
System.Text.Json.Utf8JsonReader reader);
```

Параметры

reader [Utf8JsonReader](#)

Модуль чтения, используемый для чтения.

Возвращаемое значение

[JsonDocument](#)

JsonDocument, представляющий значение (и вложенные значения), считанное из модуля чтения.

Исключения

[ArgumentException](#)

reader содержит неподдерживаемые параметры.

-или-

Текущий маркер **reader** не запускается или не представляет значение.

[JsonException](#)

Не удалось считать значение из модуля чтения.

Комментарии

TokenType Если свойство имеет `reader` значение `JsonTokenType.PropertyName` или `JsonTokenType.None`, средство чтения будет продвигаться на один вызов `Utf8JsonReader.Read()` чтобы определить начало значения.

После завершения этого метода `reader` позиционируется в окончательном маркере в значении JSON. При возникновении исключения средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, с помощью функции чтения, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocument.TryParseValue(Utf8JsonReader, JsonDocument) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Attempts to parse one JSON value (including objects or arrays) from the provided reader.

C#

```
public static bool TryParseValue (ref System.Text.Json.Utf8JsonReader  
reader, out System.Text.Json.JsonDocument? document);
```

Parameters

reader [Utf8JsonReader](#)

The reader to read.

document [JsonDocument](#)

When the method returns, contains the parsed document.

Returns

[Boolean](#)

`true` if a value was read and parsed into a `JsonDocument`; `false` if the reader ran out of data while parsing. All other situations result in an exception being thrown.

Exceptions

[ArgumentException](#)

`reader` contains unsupported options.

-or-

The current `reader` token does not start or represent a value.

JsonException

A value could not be read from the reader.

Remarks

If the `TokenType` property of `reader` is `JsonTokenType.PropertyName` or `JsonTokenType.None`, the reader will advance by one call to `Utf8JsonReader.Read()` to determine the start of the value.

Upon completion of this method, `reader` is positioned at the final token in the JSON value. If an exception is thrown or `false` is returned, the reader is reset to the state it was in when the method was called.

This method makes a copy of the data the reader acted on, so there is no caller requirement to maintain data integrity beyond the return of this method.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonDocument.WriteTo(Utf8JsonWriter)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Записывает документ в предоставленный модуль записи в виде значения JSON.

C#

```
public void WriteTo (System.Text.Json.Utf8JsonWriter writer);
```

Параметры

writer [Utf8JsonWriter](#)

Модуль записи, в который необходимо записать документ.

Исключения

[ArgumentNullException](#)

Параметр `writer` имеет значение `null`.

[InvalidOperationException](#)

`ValueKind` этого [RootElement](#) приведет к недопустимому JSON.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocumentOptions Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет пользователю возможность определить пользовательское поведение при анализе JSON для создания [JsonDocument](#).

C#

```
public struct JsonDocumentOptions
```

Наследование [Object](#) → [ValueType](#) → [JsonDocumentOptions](#)

Комментарии

Дополнительные сведения см. в статье [Сериализация и десериализация JSON](#).

Свойства

AllowTrailingCommas	Возвращает или задает значение, которое указывает, разрешена ли (и игнорируется) лишняя запятая в конце списка значений JSON в объекте или массиве внутри считываемых полезных данных JSON.
CommentHandling	Возвращает или задает значение, определяющее способ, с помощью которого JsonDocument обрабатывает комментарии при чтении данных JSON.
MaxDepth	Возвращает или задает максимальную глубину, разрешенную при анализе данных JSON, при этом значение по умолчанию (то есть 0) указывает максимальную глубину 64.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocumentOptions.AllowTrailingCommas Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, которое указывает, разрешена ли (и игнорируется) лишняя запятая в конце списка значений JSON в объекте или массиве внутри считываемых полезных данных JSON.

C#

```
public bool AllowTrailingCommas { get; set; }
```

Значение свойства

[Boolean](#)

`true` Если в конце списка значений JSON в объекте или массиве разрешена дополнительная запятая; в противном случае — `false`. Значение по умолчанию — `false`.

Комментарии

По умолчанию `AllowTrailingCommas` имеет значение `false`, а при [JsonException](#) обнаружении запятой возникает исключение .

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonDocumentOptions.CommentHandling Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets a value that determines how the [JsonDocument](#) handles comments when reading through the JSON data.

C#

```
public System.Text.Json.JsonCommentHandling CommentHandling { get; set; }
```

Property Value

[JsonCommentHandling](#)

One of the enumeration values that indicates how comments are handled.

Exceptions

[ArgumentOutOfRangeException](#)

The comment handling enum is set to a value that is not supported (or not within the [JsonCommentHandling](#) enum range).

Remarks

By default, a [JsonException](#) is thrown if a comment is encountered.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonDocumentOptions.MaxDepth Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets the maximum depth allowed when parsing JSON data, with the default (that is, 0) indicating a maximum depth of 64.

C#

```
public int MaxDepth { get; set; }
```

Property Value

[Int32](#)

The maximum depth allowed when parsing JSON data.

Exceptions

[ArgumentOutOfRangeException](#)

The max depth is set to a negative value.

Remarks

Parsing past this depth will throw a [JsonException](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Представляет определенное значение JSON в [JsonDocument](#).

C#

```
public readonly struct JsonElement
```

Наследование [Object](#) → [ValueType](#) → [JsonElement](#)

Комментарии

Дополнительные сведения см. в статье [Использование JsonDocument для доступа к данным.](#)

Свойства

Item[Int32]	Возвращает значение по указанному индексу, когда текущее значение — Array .
ValueKind	Возвращает тип текущего значения JSON.

Методы

Clone()	Возвращает JsonElement, который можно безопасно хранить по истечении времени существования исходного JsonDocument .
EnumerateArray()	Возвращает перечислитель для перечисления значений в массиве JSON, представленном этим JsonElement.
EnumerateObject()	Возвращает перечислитель для перечисления свойств в объекте JSON, представленном этим JsonElement.

GetArrayLength()	Возвращает количество значений, содержащихся в текущем значении массива.
GetBoolean()	Возвращает значение элемента в виде <code>Boolean</code> .
GetByte()	Возвращает текущий номер JSON в виде <code>Byte</code> .
GetBytesFromBase64()	Возвращает значение элемента в виде массива байтов.
GetDateTime()	Возвращает значение элемента в виде <code>DateTime</code> .
GetDateTimeOffset()	Возвращает значение элемента в виде <code>DateTimeOffset</code> .
GetDecimal()	Возвращает текущий номер JSON в виде <code>Decimal</code> .
GetDouble()	Возвращает текущий номер JSON в виде <code>Double</code> .
GetGuid()	Возвращает значение элемента в виде <code>Guid</code> .
GetInt16()	Возвращает текущий номер JSON в виде <code>Int16</code> .
GetInt32()	Возвращает текущий номер JSON в виде <code>Int32</code> .
GetInt64()	Возвращает текущий номер JSON в виде <code>Int64</code> .
GetProperty(ReadOnly Span<Byte>)	Возвращает <code>JsonElement</code> , представляющий значение требуемого свойства, определяемого по <code>utf8PropertyName</code> .
GetProperty(ReadOnly Span<Char>)	Возвращает <code>JsonElement</code> , представляющий значение требуемого свойства, определяемого по <code>propertyName</code> .
GetProperty(String)	Возвращает <code>JsonElement</code> , представляющий значение требуемого свойства, определяемого по <code>propertyName</code> .
GetRawText()	Возвращает строку, представляющую исходные входные данные, на которых основано это значение.
GetSByte()	Возвращает текущий номер JSON в виде <code>SByte</code> .
GetSingle()	Возвращает текущий номер JSON в виде <code>Single</code> .
GetString()	Возвращает значение элемента в виде <code>String</code> .
GetUInt16()	Возвращает текущий номер JSON в виде <code>UInt16</code> .
GetUInt32()	Возвращает текущий номер JSON в виде <code>UInt32</code> .
GetUInt64()	Возвращает текущий номер JSON в виде <code>UInt64</code> .
ParseValue(Utf8JsonReader)	Анализирует одно значение JSON (включая объекты или массивы) из указанного модуля чтения.

ToString()	Возвращает строковое представление для текущего значения в соответствии с типом значения.
TryGetByte(Byte)	Пытается представить текущий номер JSON в виде Byte .
TryGetBytesFromBase64(Byte[])	Пытается представить текущую строку JSON в виде массива байтов, предполагая, что она в кодировке Base64.
TryGetDateTime(DateTime)	Пытается представить текущую строку JSON в виде DateTime .
TryGetDateTimeOffset(DateTimeOffset)	Пытается представить текущую строку JSON в виде DateTimeOffset .
TryGetDecimal(Decimal)	Пытается представить текущий номер JSON в виде Decimal .
TryGetDouble(Double)	Пытается представить текущий номер JSON в виде Double .
TryGetGuid(Guid)	Пытается представить текущую строку JSON в виде Guid .
TryGetInt16(Int16)	Пытается представить текущий номер JSON в виде Int16 .
TryGetInt32(Int32)	Пытается представить текущий номер JSON в виде Int32 .
TryGetInt64(Int64)	Пытается представить текущий номер JSON в виде Int64 .
TryGetProperty(ReadOnlySpan<Byte>, JsonElement)	Ищет свойство с именем <code>utf8PropertyName</code> в текущем объекте, возвращая значение, которое указывает, существует ли такое свойство. Если свойство существует, метод назначает его значение аргументу <code>value</code> .
TryGetProperty(ReadOnlySpan<Char>, JsonElement)	Ищет свойство с именем <code>propertyName</code> в текущем объекте, возвращая значение, которое указывает, существует ли такое свойство. Если свойство существует, метод назначает его значение аргументу <code>value</code> .
TryGetProperty(String, JsonElement)	Ищет свойство с именем <code>propertyName</code> в текущем объекте, возвращая значение, которое указывает, существует ли такое свойство. Если свойство существует, его значение назначается аргументу <code>value</code> .
TryGetSByte(SByte)	Пытается представить текущий номер JSON в виде SByte .
TryGetSingle(Single)	Пытается представить текущий номер JSON в виде Single .
TryGetUInt16(UInt16)	Пытается представить текущий номер JSON в виде UInt16 .
TryGetUInt32(UInt32)	Пытается представить текущий номер JSON в виде UInt32 .
TryGetUInt64(UInt64)	Пытается представить текущий номер JSON в виде UInt64 .
TryParseValue(Utf8JsonReader, Nullable<JsonElement>)	Пытается проанализировать одно значение JSON (включая объекты или массивы) из указанного модуля чтения.

ValueEquals(ReadOnlySpan<Byte>)	Сравнивает текст, представленный байтовым диапазоном в кодировке UTF-8, со строковым значением этого элемента.
ValueEquals(ReadOnlySpan<Char>)	Сравнивает указанный диапазон символов только для чтения со строковым значением этого элемента.
ValueEquals(String)	Сравнивает указанную строку со строковым значением этого элемента.
WriteTo(Utf8JsonWriter)	Записывает элемент в указанный модуль записи в виде значения JSON.

Методы расширения

Deserialize(JsonElement, JsonTypeInfo)	Преобразует объект , JsonElement представляющий одно значение JSON, в экземпляр, заданный . jsonTypeInfo
Deserialize(JsonElement, Type, JsonSerializerOptions)	Преобразует объект , JsonElement представляющий одно значение JSON, в . returnType
Deserialize(JsonElement, Type, JsonSerializerContext)	Преобразует объект , JsonElement представляющий одно значение JSON, в . returnType
Deserialize< TValue >(JsonElement, JsonSerializerOptions)	Преобразует объект , JsonElement представляющий одно значение JSON, в . TValue
Deserialize< TValue >(JsonElement, JsonTypeInfo< TValue >)	Преобразует объект , JsonElement представляющий одно значение JSON, в . TValue

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

Да

Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.Item[Int32] Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the value at the specified index if the current value is an [Array](#).

C#

```
public System.Text.Json.JsonElement this[int index] { get; }
```

Parameters

index [Int32](#)

The item index.

Property Value

[JsonElement](#)

The value at the specified index.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Array](#).

[IndexOutOfRangeException](#)

index is not in the range [0, [GetArrayLength\(\)](#)).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ValueKind Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает тип текущего значения JSON.

C#

```
public System.Text.Json.JsonValueKind ValueKind { get; }
```

Значение свойства

[JsonValueKind](#)

Тип текущего значения JSON.

Исключения

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.Clone Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает JsonElement, который можно безопасно хранить по истечении времени существования исходного [JsonDocument](#).

C#

```
public System.Text.Json.JsonElement Clone();
```

Возвращаемое значение

[JsonElement](#)

JsonElement, который можно безопасно хранить по истечении времени существования исходного [JsonDocument](#).

Комментарии

Если это [JsonElement](#) сам по себе результат предыдущего вызова `Clone` или значение, содержащееся в другом [JsonElement](#), которое было выходными данными предыдущего вызова, этот метод не приводит к выделению `Clone` дополнительного объема памяти.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.EnumerateArray Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets an enumerator to enumerate the values in the JSON array represented by this `JsonElement`.

C#

```
public System.Text.Json.JsonElement.ArrayEnumerator GetEnumerator();
```

Returns

[JsonElement.ArrayEnumerator](#)

An enumerator to enumerate the values in the JSON array represented by this `JsonElement`.

Exceptions

[InvalidOperationException](#)

This value's `ValueKind` is not `Array`.

[ObjectDisposedException](#)

The parent `JsonDocument` has been disposed.

Remarks

For more information, see [How to write custom serializers and deserializers with System.Text.Json](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.EnumerateObject Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает перечислитель для перечисления свойств в объекте JSON, представленном этим JsonElement.

C#

```
public System.Text.Json.JsonElement.ObjectEnumerator EnumerateObject();
```

Возвращаемое значение

[JsonElement.ObjectEnumerator](#)

Перечислитель для перечисления свойств в объекте JSON, представленном этим JsonElement.

Исключения

[InvalidOperationException](#)

ValueKind этого значения не является Object.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetArrayLength Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает количество значений, содержащихся в текущем значении массива.

C#

```
public int GetArrayLength();
```

Возвращаемое значение

[Int32](#)

Количество значений, содержащихся в текущем значении массива.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Array](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Дополнительные сведения см. в статье [Создание пользовательских сериализаторов и десериализаторов с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetBoolean Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение элемента в виде [Boolean](#).

C#

```
public bool GetBoolean();
```

Возвращаемое значение

[Boolean](#)

Значение элемента в виде [Boolean](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является ни [True](#), ни [False](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetByte Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the current JSON number as a [Byte](#).

C#

```
public byte GetByte();
```

Returns

[Byte](#)

The current JSON number as a [Byte](#).

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as a [Byte](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetBytesFromBase64

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение элемента в виде массива байтов.

C#

```
public byte[] GetBytesFromBase64 ();
```

Возвращаемое значение

[Byte\[\]](#)

Значение, декодированное как массив байтов.

Исключения

[InvalidOperationException](#)

`ValueKind` этого значения не является [String](#).

[FormatException](#)

Значение не закодировано в тексте Base64 и поэтому не может быть декодировано в байты.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не создает байтовое представление значений, отличных от строк JSON в кодировке Base64.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetDateTime Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the value of the element as a [DateTime](#).

C#

```
public DateTime GetDateTime();
```

Returns

[DateTime](#)

The value of the element as a [DateTime](#).

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [String](#).

[FormatException](#)

The value cannot be read as a [DateTime](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method only creates a [DateTime](#) representation of JSON strings that conform to the ISO 8601-1 extended format (see [DateTime and DateTimeOffset support in System.Text.Json](#)).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetDateTimeOffset Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение элемента в виде [DateTimeOffset](#).

C#

```
public DateTimeOffset GetDateTimeOffset();
```

Возвращаемое значение

[DateTimeOffset](#)

Значение элемента в виде [DateTimeOffset](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [String](#).

[FormatException](#)

Это значение не может быть прочитано как [DateTimeOffset](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод создает только представление `dateTimeOffset` строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetDecimal Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the current JSON number as a [Decimal](#).

C#

```
public decimal GetDecimal();
```

Returns

[Decimal](#)

The current JSON number as a [Decimal](#).

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as a [Decimal](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetDouble Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает текущий номер JSON в виде [Double](#).

C#

```
public double GetDouble();
```

Возвращаемое значение

[Double](#)

Текущий номер JSON в виде [Double](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[FormatException](#)

Значение невозможно представить в виде [Double](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

В .NET Core этот метод возвращает [Double.PositiveInfinity](#) значения, превышающие [Double.MaxValue](#), и [Double.NegativeInfinity](#) для значений меньше [Double.MinValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetGuid Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение элемента в виде [Guid](#).

C#

```
public Guid GetGuid();
```

Возвращаемое значение

[Guid](#)

Значение элемента в виде [Guid](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [String](#).

[FormatException](#)

Значение невозможно представить в виде [Guid](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не создает guid-представление значений, отличных от строк JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetInt16 Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the current JSON number as an [Int16](#).

C#

```
public short GetInt16();
```

Returns

[Int16](#)

The current JSON number as an [Int16](#).

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as an [Int16](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetInt32 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает текущий номер JSON в виде [Int32](#).

C#

```
public int GetInt32();
```

Возвращаемое значение

[Int32](#)

Текущий номер JSON в виде [Int32](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[FormatException](#)

Значение невозможно представить в виде [Int32](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetInt64 Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the current JSON number as an [Int64](#).

C#

```
public long GetInt64 ();
```

Returns

[Int64](#)

The current JSON number as an [Int64](#).

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as a [Int64](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetProperty Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

GetProperty(ReadOnlySpan<Byte>)	Возвращает JsonElement , представляющий значение требуемого свойства, определяемого по <code>utf8PropertyName</code> .
GetProperty(ReadOnlySpan<Char>)	Возвращает JsonElement , представляющий значение требуемого свойства, определяемого по <code>propertyName</code> .
GetProperty(String)	Возвращает JsonElement , представляющий значение требуемого свойства, определяемого по <code>propertyName</code> .

GetProperty(ReadOnlySpan<Byte>)

Возвращает [JsonElement](#), представляющий значение требуемого свойства, определяемого по `utf8PropertyName`.

C#

```
public System.Text.Json.JsonElement GetProperty (ReadOnlySpan<byte>  
utf8PropertyName);
```

Параметры

`utf8PropertyName` [ReadOnlySpan<Byte>](#)

Представление в кодировке UTF-8 (без метки порядка байтов) для имени возвращаемого свойства.

Возвращаемое значение

[JsonElement](#)

[JsonElement](#), представляющий значение запрошенного свойства.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Object](#).

[KeyNotFoundException](#)

Свойство с запрошенным именем не найдено.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Сопоставление имен свойств выполняется в виде порядкового сравнения с учетом регистра.

Если свойство определено несколько раз для одного и того же объекта, метод соответствует последнему такому определению.

Дополнительные сведения см. в статье [Создание пользовательских сериализаторов и десериализаторов с помощью System.Text.Json](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

GetProperty(ReadOnlySpan<Char>)

Возвращает [JsonElement](#), представляющий значение требуемого свойства, определяемого по `propertyName`.

C#

```
public System.Text.Json.JsonElement GetProperty (ReadOnlySpan<char>  
propertyName);
```

Параметры

<code>propertyName</code>	<code>ReadOnlySpan<Char></code>
---------------------------	---------------------------------------

Имя свойства, значение которого требуется возвратить.

Возвращаемое значение

<code>JsonElement</code>

`JsonElement`, представляющий значение запрошенного свойства.

Исключения

<code>InvalidOperationException</code>
--

`ValueKind` этого значения не является `Object`.

<code>KeyNotFoundException</code>

Свойство с запрошенным именем не найдено.

<code>ObjectDisposedException</code>

Родительский объект `JsonDocument` был удален.

Комментарии

Сопоставление имен свойств выполняется в виде порядкового сравнения с учетом регистра.

Если свойство определено несколько раз для одного и того же объекта, метод соответствует последнему такому определению.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

GetProperty(String)

Возвращает `JsonElement`, представляющий значение требуемого свойства, определяемого по `propertyName`.

C#

```
public System.Text.Json.JsonElement GetProperty (string propertyName);
```

Параметры

propertyName [String](#)

Имя свойства, значение которого требуется возвратить.

Возвращаемое значение

[JsonElement](#)

[JsonElement](#), представляющий значение запрошенного свойства.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Object](#).

[KeyNotFoundException](#)

Свойство с запрошенным именем не найдено.

[ArgumentNullException](#)

propertyName имеет значение `null`.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Сопоставление имен свойств выполняется в виде порядкового сравнения с учетом регистра.

Если свойство определено несколько раз для одного и того же объекта, метод соответствует последнему такому определению.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetRawText Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает строку, представляющую исходные входные данные, на которых основано это значение.

C#

```
public string GetRawText();
```

Возвращаемое значение

[String](#)

Исходные входные данные, на которых основано это значение.

Исключения

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetSByte Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Gets the current JSON number as an [SByte](#).

C#

```
[System.CLSCompliant(false)]
public sbyte GetSByte();
```

Returns

[SByte](#)

The current JSON number as an [SByte](#).

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as an [SByte](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetSingle Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает текущий номер JSON в виде [Single](#).

C#

```
public float GetSingle();
```

Возвращаемое значение

[Single](#)

Текущий номер JSON в виде [Single](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[FormatException](#)

Значение невозможно представить в виде [Single](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

В .NET Core этот метод возвращает [Single.PositiveInfinity](#) значения больше [Single.MaxValue](#) и [Single.NegativeInfinity](#) значения меньше [Single.MinValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение элемента в виде [String](#).

C#

```
public string? GetString();
```

Возвращаемое значение

[String](#)

Значение элемента в виде [String](#).

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является ни [String](#), ни [Null](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не создает строковое представление значений, отличных от строк JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.GetInt16 Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Gets the current JSON number as a [UInt16](#).

C#

```
[System.CLSCompliant(false)]
public ushort GetUInt16();
```

Returns

[UInt16](#)

The current JSON number as a [UInt16](#).

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as a [UInt16](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?



Yes



No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetInt32 Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Gets the current JSON number as a [UInt32](#).

C#

```
[System.CLSCompliant(false)]
public uint GetUInt32();
```

Returns

[UInt32](#)

The current JSON number as a [UInt32](#).

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[FormatException](#)

The value cannot be represented as a [UInt32](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?



Yes



No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.GetInt64 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Возвращает текущий номер JSON в виде [UInt64](#).

C#

```
[System.CLSCompliant(false)]
public ulong GetUInt64();
```

Возвращаемое значение

[UInt64](#)

Текущий номер JSON в виде [UInt64](#).

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[FormatException](#)

Значение невозможно представить в виде [UInt64](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ParseValue(Utf8JsonReader) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует одно значение JSON (включая объекты или массивы) из указанного модуля чтения.

C#

```
public static System.Text.Json.JsonElement ParseValue (ref  
System.Text.Json.Utf8JsonReader reader);
```

Параметры

reader [Utf8JsonReader](#)

Модуль чтения, используемый для чтения.

Возвращаемое значение

[JsonElement](#)

Объект [JsonElement](#), представляющий значение (и вложенные значения), считываемое из средства чтения.

Исключения

[ArgumentException](#)

reader использует неподдерживаемые параметры.

[ArgumentException](#)

Текущий маркер **reader** не запускается или не представляет значение.

[JsonException](#)

Не удалось считать значение из модуля чтения.

Комментарии

TokenType Если свойство имеет значение `PropertyName` или `None`, средство чтения будет расширено одним вызовом для `Read()` определения начала значения.

После завершения этого метода `reader` позиционируется в последнем токене в значении JSON. Если возникает исключение, средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, на которые действовал читатель, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Применяется к

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ToString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает строковое представление для текущего значения в соответствии с типом значения.

C#

```
public override string ToString();
```

Возвращаемое значение

[String](#)

Строковое представление для текущего значения в соответствии с типом значения.

Исключения

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

[String.Empty](#) Для [JsonValueKind.Null](#)возвращается значение .

[Boolean.TrueString](#) Для [JsonValueKind.True](#)возвращается значение .

[Boolean.FalseString](#) Для [JsonValueKind.False](#)возвращается значение .

Для [JsonValueKind.String](#)возвращается значение [GetString\(\)](#) .

Для других типов возвращается значение [GetRawText\(\)](#) .

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetByte(Byte) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Byte](#).

C#

```
public bool TryGetByte (out byte value);
```

Параметры

value [Byte](#)

При возврате этим методом содержит байтовый эквивалент текущего числа JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Byte](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetBytesFromBase64(Byte[]) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущую строку JSON в виде массива байтов, предполагая, что она в кодировке Base64.

C#

```
public bool TryGetBytesFromBase64 (out byte[]? value);
```

Параметры

value [Byte\[\]](#)

Если метод выполнен, содержит декодированное двоичное представление текста в кодировке Base64.

Возвращаемое значение

[Boolean](#)

`true`, если все значение токена закодировано как допустимый текст в Base64 и может быть успешно декодировано в байты; в противном случае — `false`.

Исключения

[InvalidOperationException](#)

`ValueKind` этого значения не является [String](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не создает представление массива байтов для значений, отличных от строк JSON в кодировке Base64.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetDateTime(DateTime)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущую строку JSON в виде [DateTime](#).

C#

```
public bool TryGetDateTime (out DateTime value);
```

Параметры

value [DateTime](#)

При возврате этого метода содержит значение даты и времени, эквивалентное текущей строке JSON, если преобразование выполнено успешно или [MinValue](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если строку можно представить в виде [DateTime](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [String](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод создает только представление DateTime строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetDateTimeOffset(DateTimeOffset) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущую строку JSON в виде [DateTimeOffset](#).

C#

```
public bool TryGetDateTimeOffset (out DateTimeOffset value);
```

Параметры

value [DateTimeOffset](#)

При возврате этим методом содержит значение даты и времени, эквивалентное текущей строке JSON, если преобразование выполнено успешно или [MinValue](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если строку можно представить в виде [DateTimeOffset](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [String](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод создает только представление `dateTimeOffset` строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetDecimal(Decimal) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Attempts to represent the current JSON number as a [Decimal](#).

C#

```
public bool TryGetDecimal (out decimal value);
```

Parameters

value [Decimal](#)

When this method returns, contains the decimal equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the number can be represented as a [Decimal](#); otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryGetDouble(Double)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Double](#).

C#

```
public bool TryGetDouble (out double value);
```

Параметры

value [Double](#)

При возврате этого метода содержит значение двойной точности с плавающей запятой, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Double](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

В .NET Core этот метод не возвращает `false` значения `Double.MaxValue` больше или меньше `Double.MinValue`. Вместо этого он возвращает `true` и назначает `Double.PositiveInfinity` или `Double.NegativeInfinity`.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetGuid(Guid) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущую строку JSON в виде [Guid](#).

C#

```
public bool TryGetGuid (out Guid value);
```

Параметры

value [Guid](#)

При возврате этим методом содержит ИДЕНТИФИКАТОР GUID, эквивалентный текущей строке JSON, если преобразование выполнено успешно или [Empty](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если строку можно представить в виде [Guid](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [String](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не создает представление guid значений, отличных от строк JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetInt16(Int16) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Int16](#).

C#

```
public bool TryGetInt16 (out short value);
```

Параметры

value [Int16](#)

При возврате этим методом содержит 16-разрядное целое значение, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось неудачно.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Int16](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetInt32(Int32) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Int32](#).

C#

```
public bool TryGetInt32 (out int value);
```

Параметры

value [Int32](#)

При возврате этого метода содержит 32-разрядное целое число, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось неудачно.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Int32](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetInt64(Int64) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Int64](#).

C#

```
public bool TryGetInt64 (out long value);
```

Параметры

value [Int64](#)

При возврате этого метода содержит 64-разрядное целое число, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось неудачно.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Int64](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetProperty Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

TryGetProperty(String, JsonElement)	Looks for a property named <code>propertyName</code> in the current object, returning a value that indicates whether or not such a property exists. When the property exists, its value is assigned to the <code>value</code> argument.
TryGetProperty(ReadOnlySpan<Byte>, JsonElement)	Looks for a property named <code>utf8PropertyName</code> in the current object, returning a value that indicates whether or not such a property exists. When the property exists, the method assigns its value to the <code>value</code> argument.
TryGetProperty(ReadOnlySpan<Char>, JsonElement)	Looks for a property named <code>propertyName</code> in the current object, returning a value that indicates whether or not such a property exists. When the property exists, the method assigns its value to the <code>value</code> argument.

TryGetProperty(String, JsonElement)

Looks for a property named `propertyName` in the current object, returning a value that indicates whether or not such a property exists. When the property exists, its value is assigned to the `value` argument.

C#

```
public bool TryGetProperty (string propertyName, out  
System.Text.Json.JsonElement value);
```

Parameters

propertyName [String](#)

The name of the property to find.

value [JsonElement](#)

When this method returns, contains the value of the specified property.

Returns

[Boolean](#)

`true` if the property was found; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Object](#).

[ArgumentNullException](#)

`propertyName` is `null`.

[ObjectDisposedException](#)

The parent [JsonDocument](#) has been disposed.

Remarks

Property name matching is performed as an ordinal, case-sensitive comparison.

If a property is defined multiple times for the same object, the method matches the last such definition.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

TryGetProperty(ReadOnlySpan<Byte>, JsonElement)

Looks for a property named `utf8PropertyName` in the current object, returning a value that indicates whether or not such a property exists. When the property exists,

the method assigns its value to the `value` argument.

C#

```
public bool TryGetProperty (ReadOnlySpan<byte> utf8PropertyName, out  
System.Text.Json.JsonElement value);
```

Parameters

utf8PropertyName `ReadOnlySpan<Byte>`

The UTF-8 (with no Byte-Order-Mark (BOM)) representation of the name of the property to return.

value `JsonElement`

Receives the value of the located property.

Returns

`Boolean`

`true` if the property was found; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's `ValueKind` is not `Object`.

[ObjectDisposedException](#)

The parent `JsonDocument` has been disposed.

Remarks

Property name matching is performed as an ordinal, case-sensitive comparison.

If a property is defined multiple times for the same object, the method matches the last such definition.

For more information, see [How to write custom serializers and deserializers with System.Text.Json](#).

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

TryGetProperty(ReadOnlySpan<Char>, JsonElement)

Looks for a property named `propertyName` in the current object, returning a value that indicates whether or not such a property exists. When the property exists, the method assigns its value to the `value` argument.

C#

```
public bool TryGetProperty (ReadOnlySpan<char> propertyName, out  
System.Text.Json.JsonElement value);
```

Parameters

propertyName `ReadOnlySpan<Char>`

The name of the property to find.

value `JsonElement`

When this method returns, contains the value of the specified property.

Returns

`Boolean`

`true` if the property was found; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's `ValueKind` is not `Object`.

[ObjectDisposedException](#)

The parent `JsonDocument` has been disposed.

Remarks

Property name matching is performed as an ordinal, case-sensitive comparison.

If a property is defined multiple times for the same object, the method matches the last such definition.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryGetSByte(SByte) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Attempts to represent the current JSON number as an [SByte](#).

C#

```
[System.CLSCompliant(false)]
public bool TryGetSByte (out sbyte value);
```

Parameters

value [SByte](#)

When this method returns, contains the signed byte equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the number can be represented as an [SByte](#); otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

ObjectDisposedException

The parent `JsonDocument` has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryGetSingle(Single)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается представить текущий номер JSON в виде [Single](#).

C#

```
public bool TryGetSingle (out float value);
```

Параметры

value [Single](#)

При возврате этого метода содержит значение с плавающей запятой одной точности, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось неудачно.

Возвращаемое значение

[Boolean](#)

Значение `true`, если номер можно представить в виде [Single](#), иначе — `false`.

Исключения

[InvalidOperationException](#)

[ValueKind](#) этого значения не является [Number](#).

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Комментарии

Этот метод не анализирует содержимое строкового значения JSON.

В .NET Core этот метод не возвращает `false` значения `Single.MaxValue` больше или меньше `Single.MinValue`). Вместо этого он возвращает `true` и назначает `Single.PositiveInfinity` аргументу или `.Single.NegativeInfinity value`

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.TryGetUInt16(UInt16) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Attempts to represent the current JSON number as a [UInt16](#).

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt16 (out ushort value);
```

Parameters

value [UInt16](#)

When this method returns, contains the unsigned 16-bit integer value equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the number can be represented as a [UInt16](#); otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

ObjectDisposedException

The parent `JsonDocument` has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryGetUInt32(UInt32) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Attempts to represent the current JSON number as a [UInt32](#).

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt32 (out uint value);
```

Parameters

value [UInt32](#)

When this method returns, contains unsigned 32-bit integer value equivalent to the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the number can be represented as a [UInt32](#); otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

ObjectDisposedException

The parent `JsonDocument` has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryGetUInt64(UInt64) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Attempts to represent the current JSON number as a [UInt64](#).

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt64 (out ulong value);
```

Parameters

value [UInt64](#)

When this method returns, contains unsigned 64-bit integer value equivalent to the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the number can be represented as a [UInt64](#); otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [Number](#).

ObjectDisposedException

The parent `JsonDocument` has been disposed.

Remarks

This method does not parse the contents of a JSON string value.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.TryParseValue(Utf8JsonReader, Nullable<JsonElement>) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Attempts to parse one JSON value (including objects or arrays) from the provided reader.

C#

```
public static bool TryParseValue (ref System.Text.Json.Utf8JsonReader  
reader, out System.Text.Json.JsonElement? element);
```

Parameters

reader [Utf8JsonReader](#)

The reader to read.

element [Nullable<JsonElement>](#)

Receives the parsed element.

Returns

[Boolean](#)

`true` if a value was read and parsed into a `JsonElement`; `false` if the reader ran out of data while parsing. All other situations result in an exception being thrown.

Exceptions

[ArgumentException](#)

`reader` is using unsupported options.

[ArgumentException](#)

The current `reader` token does not start or represent a value.

JsonException

A value could not be read from the reader.

Remarks

If the `TokenType` property of `reader` is `PropertyName` or `None`, the reader will be advanced by one call to `Read()` to determine the start of the value.

Upon completion of this method, `reader` is positioned at the final token in the JSON value. If an exception is thrown or `false` is returned, the reader is reset to the state it was in when the method was called.

This method makes a copy of the data the reader acted on, so there is no caller requirement to maintain data integrity beyond the return of this method.

Applies to

Product	Versions
.NET	6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ValueEquals Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

ValueEquals(ReadOnlySpan<Byte>)	Compares the text represented by a UTF8-encoded byte span to the string value of this element.
---	--

ValueEquals(ReadOnlySpan<Char>)	Compares a specified read-only character span to the string value of this element.
---	--

ValueEquals(String)	Compares a specified string to the string value of this element.
-------------------------------------	--

ValueEquals(ReadOnlySpan<Byte>)

Compares the text represented by a UTF8-encoded byte span to the string value of this element.

C#

```
public bool ValueEquals (ReadOnlySpan<byte> utf8Text);
```

Parameters

utf8Text [ReadOnlySpan<Byte>](#)

The UTF-8 encoded text to compare against.

Returns

[Boolean](#)

`true` if the string value of this element has the same UTF-8 encoding as `utf8Text`; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [String](#).

Remarks

This method is functionally equal to doing an ordinal comparison of the string produced by UTF-8 decoding `utf8Text` with the result of calling [GetString\(\)](#), but avoids creating the string instances.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

ValueEquals(ReadOnlySpan<Char>)

Compares a specified read-only character span to the string value of this element.

C#

```
public bool ValueEquals (ReadOnlySpan<char> text);
```

Parameters

text [ReadOnlySpan<Char>](#)

The text to compare against.

Returns

[Boolean](#)

`true` if the string value of this element matches `text`; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [String](#).

Remarks

This method is functionally equal to doing an ordinal comparison of `text` and the result of calling [GetString\(\)](#), but avoids creating the string instance.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

ValueEquals(String)

Compares a specified string to the string value of this element.

C#

```
public bool ValueEquals (string? text);
```

Parameters

text [String](#)

The text to compare against.

Returns

[Boolean](#)

`true` if the string value of this element matches `text`; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

This value's [ValueKind](#) is not [String](#).

Remarks

This method is functionally equal to doing an ordinal comparison of `text` and the result of calling [GetString\(\)](#), but avoids creating the string instance.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.WriteTo(Utf8JsonWriter) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Writes the element to the specified writer as a JSON value.

C#

```
public void WriteTo (System.Text.Json.Utf8JsonWriter writer);
```

Parameters

writer [Utf8JsonWriter](#)

The writer to which to write the element.

Exceptions

[ArgumentNullException](#)

The `writer` parameter is `null`.

[InvalidOperationException](#)

The `ValueKind` of this value is [Undefined](#).

[ObjectDisposedException](#)

The parent `JsonDocument` has been disposed.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator

Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Представляет перечислитель для содержимого массива JSON.

C#

```
public struct JsonElement.ArrayEnumerator :  
    System.Collections.Generic.IEnumerable<System.Text.Json.JsonElement>,  
    System.Collections.Generic.IEnumerator<System.Text.Json.JsonElement>
```

Наследование [Object](#) → [ValueType](#) → [JsonElement.ArrayEnumerator](#)

Реализации [IEnumerable<JsonElement>](#) , [IEnumerator<JsonElement>](#) , [IEnumerable](#) , [IEnumerator](#) , [IDisposable](#)

Свойства

[Current](#)

Возвращает элемент коллекции, соответствующий текущей позиции перечислителя.

Методы

[Dispose\(\)](#)

Освобождает ресурсы, используемые этим экземпляром [JsonElement.ArrayEnumerator](#).

[GetEnumerator\(\)](#)

Возвращает перечислитель, который осуществляет итерацию по коллекции.

[MoveNext\(\)](#)

Перемещает перечислитель к следующему элементу коллекции.

Reset()	Устанавливает перечислитель в его начальное положение, т. е. перед первым элементом коллекции.
-------------------------	--

Явные реализации интерфейса

IEnumerable.GetEnumerator()	Возвращает перечислитель, который осуществляет итерацию по коллекции.
IEnumerable<JsonElement>.GetEnumerator()	Возвращает перечислитель, который осуществляет итерацию по коллекции.
IEnumerator.Current	Возвращает элемент коллекции, соответствующий текущей позиции перечислителя.

Методы расширения

ToFrozenDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает из FrozenDictionary<TKey,TValue> объекта в соответствии с указанной IEnumerable<T> функцией селектора ключей.
ToFrozenDictionary<TSource,TKey,TElement> (IEnumerable<TSource> , Func<TSource,TKey> , Func<TSource,TElement> , IEqualityComparer<TKey>)	Создает словарь FrozenDictionary<TKey,TValue> из объекта IEnumerable<T> в соответствии с заданными функциями селектора ключа и селектора элемента.
ToFrozenSet<T> (IEnumerable<T> , IEqualityComparer<T>)	Создает с FrozenSet<T> указанными значениями.
ToImmutableArray<TSource> (IEnumerable<TSource>)	Создает неизменяемый массив на основе указанной коллекции.
ToImmutableDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey>)	Создает неизменяемый словарь на основе существующей коллекции элементов, применяя функцию преобразования к исходным ключам.
ToImmutableDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает неизменяемый словарь на основе последовательности, подвергнутой определенному преобразованию.
ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource> , Func<TSource,TKey> , Func<TSource,TValue>)	Выполняет перечисление и преобразование последовательности и создает неизменяемый словарь на основе ее содержимого.

<code>ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TKey>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый словарь с использованием указанной функции сравнения ключей.
<code>ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый словарь с использованием указанных функций сравнения ключей и значений.
<code>ToImmutableHashSet<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает неизменяемый набор хэширования на основе ее содержимого.
<code>ToImmutableHashSet<TSource> (IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Выполняет перечисление последовательности, создает неизменяемый набор хэширования на основе ее содержимого и использует указанную функцию сравнения для типа набора.
<code>ToImmutableList<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает неизменяемый список на основе ее содержимого.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает неизменяемый отсортированный словарь на основе ее содержимого.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый отсортированный словарь с использованием указанной функции сравнения ключей.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>, IEqualityComparer<TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый отсортированный словарь с использованием указанных функций сравнения ключей и значений.
<code>ToImmutableSortedSet<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает

	неизменяемый отсортированный набор на основе ее содержимого.
ToImmutableSortedSet<TSource> (IEnumerable<TSource>, IComparer<TSource>)	Выполняет перечисление последовательности, создает неизменяемый отсортированный набор на основе ее содержимого и использует указанную функцию сравнения.
CopyToDataTable<T>(IEnumerable<T>)	Возвращает объект DataTable , содержащий копии объектов DataRow при заданном входном объекте IEnumerable<T> и универсальном параметре T , равном DataRow .
CopyToDataTable<T>(IEnumerable<T>, Data Table, LoadOption)	Копирует объекты DataRow в указанный объект DataTable с заданным входным объектом IEnumerable<T> , где универсальный параметр T имеет значение DataRow .
CopyToDataTable<T>(IEnumerable<T>, Data Table, LoadOption, FillErrorEventHandler)	Копирует объекты DataRow в указанный объект DataTable с заданным входным объектом IEnumerable<T> , где универсальный параметр T имеет значение DataRow .
Aggregate<TSource>(IEnumerable<TSource>, Func<TSource,TSource,TSource>)	Применяет к последовательности агрегатную функцию.
Aggregate<TSource,TAccumulate> (IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>)	Применяет к последовательности агрегатную функцию. Указанное начальное значение используется в качестве исходного значения агрегатной операции.
Aggregate<TSource,TAccumulate,TResult> (IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>, Func<TAccumulate,TResult>)	Применяет к последовательности агрегатную функцию. Указанное начальное значение служит исходным значением для агрегатной операции, а указанная функция используется для выбора результирующего значения.
All<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Проверяет, все ли элементы последовательности удовлетворяют условию.
Any<TSource>(IEnumerable<TSource>)	Проверяет, содержит ли последовательность какие-либо элементы.
Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Проверяет, удовлетворяет ли какой-либо элемент последовательности заданному

	условию.
Append<TSource>(IEnumerable<TSource>, TSource)	Добавляет значение в конец последовательности.
AsEnumerable<TSource>(IEnumerable<TSource>)	Возвращает входное значение, типизированное как IEnumerable<T> .
Average<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вычисляет среднее для последовательности значений типа Decimal , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вычисляет среднее для последовательности значений типа Double , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вычисляет среднее для последовательности значений типа Int32 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вычисляет среднее для последовательности значений типа Int64 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Вычисляет среднее для последовательности значений Decimal обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Вычисляет среднее для последовательности значений Double обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Вычисляет среднее для последовательности значений Int32 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.

Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Вычисляет среднее для последовательности значений Int64 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Вычисляет среднее для последовательности значений Single обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Вычисляет среднее для последовательности значений типа Single , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Cast<TResult>(IEnumerable)	Приводит элементы объекта IEnumerable к заданному типу.
Chunk<TSource>(IEnumerable<TSource>, Int32)	Разбивает элементы последовательности на блоки размером не более size .
Concat<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Объединяет две последовательности.
Contains<TSource>(IEnumerable<TSource>, TSource)	Определяет, содержится ли указанный элемент в последовательности, используя компаратор проверки на равенство по умолчанию.
Contains<TSource>(IEnumerable<TSource>, TSource, IEqualityComparer<TSource>)	Определяет, содержит ли последовательность заданный элемент, используя указанный компаратор IEqualityComparer<T> .
Count<TSource>(IEnumerable<TSource>)	Возвращает количество элементов в последовательности.
Count<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Возвращает число, представляющее количество элементов последовательности, удовлетворяющих заданному условию.
DefaultIfEmpty<TSource>(IEnumerable<TSource>)	Возвращает элементы указанной последовательности или одноэлементную коллекцию, содержащую значение параметра типа по умолчанию, если последовательность пуста.

<code>DefaultIfEmpty<TSource> (IEnumerable<TSource>, TSource)</code>	Возвращает элементы указанной последовательности или одноэлементную коллекцию, содержащую указанное значение, если последовательность пуста.
<code>Distinct<TSource>(IEnumerable<TSource>)</code>	Возвращает различающиеся элементы последовательности, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Distinct<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Возвращает различающиеся элементы последовательности, используя для сравнения значений указанный компаратор <code>IEqualityComparer<T></code> .
<code>DistinctBy<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)</code>	Возвращает отдельные элементы из последовательности в соответствии с указанной функцией селектора ключей.
<code>DistinctBy<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)</code>	Возвращает отдельные элементы из последовательности в соответствии с указанной функцией селектора ключей и с помощью указанного компаратора для сравнения ключей.
<code>ElementAt<TSource>(IEnumerable<TSource>, Index)</code>	Возвращает элемент по указанному индексу в последовательности.
<code>ElementAt<TSource>(IEnumerable<TSource>, Int32)</code>	Возвращает элемент по указанному индексу в последовательности.
<code>ElementAtOrDefault<TSource> (IEnumerable<TSource>, Index)</code>	Возвращает элемент последовательности по указанному индексу или значение по умолчанию, если индекс вне допустимого диапазона.
<code>ElementAtOrDefault<TSource> (IEnumerable<TSource>, Int32)</code>	Возвращает элемент последовательности по указанному индексу или значение по умолчанию, если индекс вне допустимого диапазона.
<code>Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)</code>	Находит разность множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Находит разность множеств, представленных двумя последовательностями, используя для

	сравнения значений указанный компаратор IEqualityComparer<T> .
ExceptBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TKey> , Func<TSource,TKey>)	Создает разность наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
ExceptBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TKey> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает разность наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
First<TSource> (IEnumerable<TSource>)	Возвращает первый элемент последовательности.
First<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Возвращает первый элемент последовательности, удовлетворяющий указанному условию.
FirstOrDefault<TSource> (IEnumerable<TSource>)	Возвращает первый элемент последовательности или значение по умолчанию, если последовательность не содержит элементов.
FirstOrDefault<TSource> (IEnumerable<TSource> , TSource)	Возвращает первый элемент последовательности или указанное значение по умолчанию, если последовательность не содержит элементов.
FirstOrDefault<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Возвращает первый элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если ни одного такого элемента не найдено.
FirstOrDefault<TSource> (IEnumerable<TSource> , Func<TSource,Boolean> , TSource)	Возвращает первый элемент последовательности, удовлетворяющий условию, или указанное значение по умолчанию, если такой элемент не найден.
GroupBy<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey>)	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа.
GroupBy<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и сравнивает ключи с помощью указанного компаратора.
GroupBy<TSource,TKey,TElement> (IEnumerable<TSource> , Func<TSource,TKey> ,	Группирует элементы последовательности в соответствии с заданной функцией

<code>Func<TSource,TElement>)</code>	селектора ключа и проецирует элементы каждой группы с помощью указанной функции.
<code>GroupBy<TSource,TKey,TElement></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с функцией селектора ключа. Ключи сравниваются с помощью компаратора, элементы каждой группы проецируются с помощью указанной функции.
<code>GroupBy<TSource,TKey,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TKey,IEnumerable<TSource>,TResult>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа.
<code>GroupBy<TSource,TKey,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TKey,IEnumerable<TSource>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Ключи сравниваются с использованием заданного компаратора.
<code>GroupBy<TSource,TKey,TElement,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>,</code> <code>Func<TKey,IEnumerable<TElement>,TResult>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Элементы каждой группы проецируются с помощью указанной функции.
<code>GroupBy<TSource,TKey,TElement,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>,</code> <code>Func<TKey,IEnumerable<TElement>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Значения ключей сравниваются с помощью указанного компаратора, элементы каждой группы проецируются с помощью указанной функции.
<code>GroupJoin<TOuter,TInner,TKey,TResult></code> <code>(IEnumerable<TOuter>, IEnumerable<TInner>,</code> <code>Func<TOuter,TKey>, Func<TInner,TKey>,</code> <code>Func<TOuter,IEnumerable<TInner>,TResult>)</code>	Устанавливает корреляцию между элементами двух последовательностей на основе равенства ключей и группирует результаты. Для сравнения ключей используется компаратор проверки на равенство по умолчанию.
<code>GroupJoin<TOuter,TInner,TKey,TResult></code> <code>(IEnumerable<TOuter>, IEnumerable<TInner>,</code> <code>Func<TOuter,TKey>, Func<TInner,TKey>,</code> <code>Func<TOuter,IEnumerable<TInner>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Устанавливает корреляцию между элементами двух последовательностей на основе равенства ключей и группирует результаты. Для сравнения ключей

	используется указанный компаратор IEqualityComparer<T> .
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Находит пересечение множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Находит пересечение множеств, представленных двумя последовательностями, используя для сравнения значений указанный компаратор IEqualityComparer<T> .
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>)	Создает набор пересечения двух последовательностей в соответствии с указанной функцией селектора ключей.
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Создает набор пересечения двух последовательностей в соответствии с указанной функцией селектора ключей.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>)	Устанавливает корреляцию между элементами двух последовательностей на основе сопоставления ключей. Для сравнения ключей используется компаратор проверки на равенство по умолчанию.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>, IEqualityComparer<TKey>)	Устанавливает корреляцию между элементами двух последовательностей на основе сопоставления ключей. Для сравнения ключей используется указанный компаратор IEqualityComparer<T> .
Last<TSource>(IEnumerable<TSource>)	Возвращает последний элемент последовательности.
Last<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает последний элемент последовательности, удовлетворяющий указанному условию.
LastOrDefault<TSource>(IEnumerable<TSource>)	Возвращает последний элемент последовательности или значение по умолчанию, если последовательность не содержит элементов.
LastOrDefault<TSource>(IEnumerable<TSource>, TSource)	Возвращает последний элемент последовательности или указанное

	значение по умолчанию, если последовательность не содержит элементов.
LastOrDefault<TSource> (IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает последний элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если ни одного такого элемента не найдено.
LastOrDefault<TSource> (IEnumerable<TSource>, Func<TSource,Boolean>, TSource)	Возвращает последний элемент последовательности, удовлетворяющий условию, или указанное значение по умолчанию, если такой элемент не найден.
LongCount<TSource>(IEnumerable<TSource>)	Возвращает значение типа Int64 , представляющее общее число элементов в последовательности.
LongCount<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает значение типа Int64 , представляющее число элементов последовательности, удовлетворяющих заданному условию.
Max<TSource>(IEnumerable<TSource>)	Возвращает максимальное значение, содержащееся в универсальной последовательности.
Max<TSource>(IEnumerable<TSource>, IComparer<TSource>)	Возвращает максимальное значение, содержащееся в универсальной последовательности.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Decimal .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Double .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Int32 .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Int64 .

<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Decimal>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Decimal</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Double>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Double</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int32>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Int32</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Int64</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Single</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Single>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа <code>Single</code> .
<code>Max<TSource, TResult>(IEnumerable<TSource>, Func<TSource, TResult>)</code>	Вызывает функцию преобразования для каждого элемента универсальной последовательности и возвращает максимальное результирующее значение.
<code>MaxBy<TSource, TKey>(IEnumerable<TSource>, Func<TSource, TKey>)</code>	Возвращает максимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей.
<code>MaxBy<TSource, TKey>(IEnumerable<TSource>, Func<TSource, TKey>, IComparer<TKey>)</code>	Возвращает максимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей и компаратором ключей.
<code>Min<TSource>(IEnumerable<TSource>)</code>	Возвращает минимальное значение, содержащееся в универсальной последовательности.
<code>Min<TSource>(IEnumerable<TSource>, IComparer<TSource>)</code>	Возвращает минимальное значение, содержащееся в универсальной

	последовательности.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Decimal</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Double>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Double</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Int32</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Int64</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Decimal</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Double</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Int32</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Int64</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Single</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Single>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Single</code> .

<code>Min<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)</code>	Вызывает функцию преобразования для каждого элемента универсальной последовательности и возвращает минимальное результирующее значение.
<code>MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Возвращает минимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей.
<code>MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Возвращает минимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей и компаратором ключей.
<code>OfType<TResult>(IEnumerable)</code>	Выполняет фильтрацию элементов объекта <code>IEnumerable</code> по заданному типу.
<code>Order<T>(IEnumerable<T>)</code>	Сортирует элементы последовательности в порядке возрастания.
<code>Order<T>(IEnumerable<T>, IComparer<T>)</code>	Сортирует элементы последовательности в порядке возрастания.
<code>OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Сортирует элементы последовательности в порядке возрастания ключа.
<code>OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Сортирует элементы последовательности в порядке возрастания с использованием указанного компаратора.
<code>OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Сортирует элементы последовательности в порядке убывания ключа.
<code>OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Сортирует элементы последовательности в порядке убывания с использованием указанного компаратора.
<code>OrderDescending<T>(IEnumerable<T>)</code>	Сортирует элементы последовательности в порядке убывания.
<code>OrderDescending<T>(IEnumerable<T>, IComparer<T>)</code>	Сортирует элементы последовательности в порядке убывания.
<code>Prepend<TSource>(IEnumerable<TSource>, TSource)</code>	Добавляет значение в начало последовательности.
<code>Reverse<TSource>(IEnumerable<TSource>)</code>	Изменяет порядок элементов последовательности на противоположный.
<code>Select<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)</code>	Проецирует каждый элемент последовательности в новую форму.

<code>Func<TSource, TResult>)</code>	
<code>Select<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, TResult>)</code>	Проецирует каждый элемент последовательности в новую форму, добавляя индекс элемента.
<code>SelectMany<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, IEnumerable<TResult>>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> и объединяет результирующие последовательности в одну последовательность.
<code>SelectMany<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, IEnumerable<TResult>>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> и объединяет результирующие последовательности в одну последовательность. Индекс каждого элемента исходной последовательности используется в проецированной форме этого элемента.
<code>SelectMany<TSource, TCollection, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, IEnumerable<TCollection>>,</code> <code>Func<TSource, TCollection, TResult>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> , объединяет результирующие последовательности в одну и вызывает функцию селектора результата для каждого элемента этой последовательности.
<code>SelectMany<TSource, TCollection, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, IEnumerable<TCollection>>,</code> <code>Func<TSource, TCollection, TResult>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> , объединяет результирующие последовательности в одну и вызывает функцию селектора результата для каждого элемента этой последовательности. Индекс каждого элемента исходной последовательности используется в промежуточной проецированной форме этого элемента.
<code>SequenceEqual<TSource></code> <code>(IEnumerable<TSource>,</code> <code>IEnumerable<TSource>)</code>	Определяет, совпадают ли две последовательности, используя для сравнения элементов компаратор проверки на равенство по умолчанию, предназначенный для их типа.
<code>SequenceEqual<TSource></code> <code>(IEnumerable<TSource>,</code> <code>IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Определяет, совпадают ли две последовательности, используя для сравнения элементов указанный компаратор <code>IEqualityComparer<T></code> .

<code>Single<TSource>(IEnumerable<TSource>)</code>	Возвращает единственный элемент последовательности и генерирует исключение, если число элементов последовательности отлично от 1.
<code>Single<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, и вызывает исключение, если таких элементов больше одного.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>)</code>	Возвращает единственный элемент последовательности или значение по умолчанию, если последовательность пуста; если в последовательности более одного элемента, генерируется исключение.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, TSource)</code>	Возвращает единственный элемент последовательности или указанное значение по умолчанию, если последовательность пуста; Этот метод создает исключение, если в последовательности содержится несколько элементов.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если такого элемента не существует; если условию удовлетворяет более одного элемента, вызывается исключение.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>, TSource)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, или заданное значение по умолчанию, если такого элемента не существует; Этот метод создает исключение, если условию удовлетворяет несколько элементов.
<code>Skip<TSource>(IEnumerable<TSource>, Int32)</code>	Пропускает заданное число элементов в последовательности и возвращает остальные элементы.
<code>SkipLast<TSource>(IEnumerable<TSource>, Int32)</code>	Возвращает новую перечислимую коллекцию, содержащую элементы из <code>source</code> с исключенными <code>count</code> элементами из конца исходной коллекции.

SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы.
SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Int32,Boolean>)	Пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы. Индекс элемента используется в логике функции предиката.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вычисляет сумму последовательности значений типа Decimal , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вычисляет сумму последовательности значений типа Double , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вычисляет сумму последовательности значений типа Int32 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вычисляет сумму последовательности значений типа Int64 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Вычисляет сумму последовательности значений Decimal обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Вычисляет сумму последовательности значений Double обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Вычисляет сумму последовательности значений Int32 обнуляемого типа, получаемой в результате применения

	функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Вычисляет сумму последовательности значений Int64 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Вычисляет сумму последовательности значений Single обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Вычисляет сумму последовательности значений типа Single , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Take<TSource>(IEnumerable<TSource>, Int32)	Возвращает указанное число подряд идущих элементов с начала последовательности.
Take<TSource>(IEnumerable<TSource>, Range)	Возвращает указанный диапазон смежных элементов из последовательности.
TakeLast<TSource>(IEnumerable<TSource>, Int32)	Возвращает новую перечислимую коллекцию, содержащую последние <code>count</code> элементов из <code>source</code> .
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Возвращает цепочку элементов последовательности, удовлетворяющих указанному условию.
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Int32, Boolean>)	Возвращает цепочку элементов последовательности, удовлетворяющих указанному условию. Индекс элемента используется в логике функции предиката.
ToArray<TSource>(IEnumerable<TSource>)	Создает массив из объекта IEnumerable<T> .
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>)	Создает словарь Dictionary<TKey, TValue> из объекта IEnumerable<T> в соответствии с заданной функцией селектора ключа.
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>, IEqualityComparer<TKey>)	Создает словарь Dictionary<TKey, TValue> из объекта IEnumerable<T> в соответствии с заданной функцией селектора ключа и компаратором ключей.

<code>ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)</code>	Создает словарь <code>Dictionary<TKey,TValue></code> из объекта <code>IEnumerable<T></code> в соответствии с заданными функциями селектора ключа и селектора элемента.
<code>ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Создает словарь <code>Dictionary<TKey,TValue></code> из объекта <code>IEnumerable<T></code> в соответствии с заданным компаратором и функциями селектора ключа и селектора элемента.
<code>ToHashSet<TSource>(IEnumerable<TSource>)</code>	Создает <code>HashSet<T></code> из <code>IEnumerable<T></code> .
<code>ToHashSet<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Создает <code>HashSet<T></code> из <code>IEnumerable<T></code> с использованием <code>comparer</code> для сравнения ключей.
<code>ToList<TSource>(IEnumerable<TSource>)</code>	Создает <code>List<T></code> из <code>IEnumerable<T></code> .
<code>ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданной функцией селектора ключа.
<code>ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданной функцией селектора ключа и компаратором ключей.
<code>ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданными функциями селектора ключа и селектора элемента.
<code>ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Создает объект <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданным компаратором и функциями селектора ключа и селектора элемента.
<code>TryGetNonEnumeratedCount<TSource> (IEnumerable<TSource>, Int32)</code>	Пытается определить количество элементов в последовательности без принудительного перечисления.
<code>Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)</code>	Находит объединение множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Находит объединение множеств, представленных двумя последовательностями, используя

	указанный компаратор IEqualityComparer<T> .
UnionBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TSource> , Func<TSource,TKey>)	Создает объединение наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
UnionBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает объединение наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
Where<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Выполняет фильтрацию последовательности значений на основе заданного предиката.
Where<TSource> (IEnumerable<TSource> , Func<TSource,Int32,Boolean>)	Выполняет фильтрацию последовательности значений на основе заданного предиката. Индекс каждого элемента используется в логике функции предиката.
Zip<TFirst,TSecond> (IEnumerable<TFirst> , IEnumerable<TSecond>)	Создает последовательность кортежей с элементами из двух указанных последовательностей.
Zip<TFirst,TSecond,TThird> (IEnumerable<TFirst> , IEnumerable<TSecond> , IEnumerable<TThird>)	Создает последовательность кортежей с элементами из трех указанных последовательностей.
Zip<TFirst,TSecond,TResult> (IEnumerable<TFirst> , IEnumerable<TSecond> , Func<TFirst,TSecond,TResult>)	Применяет указанную функцию к соответствующим элементам двух последовательностей, что дает последовательность результатов.
AsParallel(IEnumerable)	Позволяет осуществлять параллельный запрос.
AsParallel<TSource> (IEnumerable<TSource>)	Позволяет осуществлять параллельный запрос.
AsQueryable(IEnumerable)	Преобразовывает коллекцию IEnumerable в объект IQueryable .
AsQueryable<TElement> (IEnumerable<TElement>)	Преобразовывает универсальный объект IEnumerable<T> в универсальный объект IQueryable<T> .
Ancestors<T> (IEnumerable<T>)	Возвращает коллекцию элементов, содержащую предков каждого узла в исходной коллекции.

Ancestors<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию элементов, содержащую предков каждого узла в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
DescendantNodes<T>(IEnumerable<T>)	Возвращает коллекцию подчиненных узлов каждого документа и элемента в исходной коллекции.
Descendants<T>(IEnumerable<T>)	Возвращает коллекцию элементов, содержащую подчиненные элементы каждого элемента и документа в исходной коллекции.
Descendants<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию элементов, содержащую подчиненные элементы каждого элемента и документа в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
Elements<T>(IEnumerable<T>)	Возвращает коллекцию дочерних элементов каждого элемента и документа в исходной коллекции.
Elements<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию дочерних элементов каждого элемента и документа в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
InDocumentOrder<T>(IEnumerable<T>)	Возвращает коллекцию узлов, содержащую все узлы в исходной коллекции, отсортированные в порядке следования документов.
Nodes<T>(IEnumerable<T>)	Возвращает коллекцию дочерних узлов каждого документа и элемента в исходной коллекции.
Remove<T>(IEnumerable<T>)	Удаление каждого узла в исходной коллекции из родительского узла.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ArrayEnumerator.Current Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the element in the collection at the current position of the enumerator.

C#

```
public System.Text.Json.JsonElement Current { get; }
```

Property Value

[JsonElement](#)

The element in the collection at the current position of the enumerator.

Implements

[Current](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator.Dispose

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Освобождает ресурсы, используемые этим экземпляром [JsonElement.ArrayEnumerator](#).

C#

```
public void Dispose();
```

Реализации

[Dispose\(\)](#)

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ArrayEnumerator.Get Enumerator Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Returns an enumerator that iterates through a collection.

C#

```
public System.Text.Json.JsonElement.ArrayEnumerator GetEnumerator();
```

Returns

[JsonElement.ArrayEnumerator](#)

An enumerator that can be used to iterate through the array.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator.MoveNext Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Advances the enumerator to the next element of the collection.

C#

```
public bool MoveNext();
```

Returns

[Boolean](#)

`true` if the enumerator was successfully advanced to the next element; `false` if the enumerator has passed the end of the collection.

Implements

[MoveNext\(\)](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator.Reset Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Sets the enumerator to its initial position, which is before the first element in the collection.

C#

```
public void Reset();
```

Implements

[Reset\(\)](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator. IEnumerable<JsonElement>.Get Enumerator Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает перечислитель, который осуществляет итерацию по коллекции.

C#

```
System.Collections.Generic.IEnumerable<System.Text.Json.JsonElement>
IEnumerable<JsonElement>.GetEnumerator ();
```

Возвращаемое значение

[IEnumerator<JsonElement>](#)

Перечислитель массива элементов [JsonElement](#), который может использоваться для итерации по коллекции.

Реализации

[GetEnumerator\(\)](#)

Комментарии

Этот член представляет собой явную реализацию члена интерфейса. Его можно использовать, только если [JsonElement.ArrayEnumerator](#) экземпляр приведен к интерфейсу [\[IEnumerable<JsonElement>\]](#) ([`<xref:System.Collections.Generic.IEnumerable%601`](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ArrayEnumerator. IEnumerable.GetEnumerator Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Returns an enumerator that iterates through a collection.

C#

```
System.Collections.IEnumerator IEnumerable.GetEnumerator();
```

Returns

[IEnumerator](#)

An enumerator that can be used to iterate through the collection.

Implements

[GetEnumerator\(\)](#)

Remarks

This member is an explicit interface member implementation. It can be used only when the [JsonElement.ArrayEnumerator](#) instance is cast to an [IEnumerable](#) interface.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ArrayEnumerator. IEnumerator.Current Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the element in the collection at the current position of the enumerator.

C#

```
object System.Collections.IEnumerator.Current { get; }
```

Property Value

[Object](#)

The element in the collection at the current position of the enumerator.

Implements

[Current](#)

Remarks

This member is an explicit interface member implementation. It can be used only when the [JsonElement.ArrayEnumerator](#) instance is cast to an [IEnumerator](#) interface.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ObjectEnumerator

Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Представляет перечислитель для свойств объекта JSON.

C#

```
public struct JsonElement.ObjectEnumerator :  
    System.Collections.Generic.IEnumerable<System.Text.Json.JsonProperty>,  
    System.Collections.Generic.IEnumerator<System.Text.Json.JsonProperty>
```

Наследование [Object](#) → [ValueType](#) → [JsonElement.ObjectEnumerator](#)

Реализации [IEnumerable<JsonProperty>](#) , [IEnumerator<JsonProperty>](#) , [IEnumerable](#) ,
[IEnumerator](#) , [IDisposable](#)

Свойства

[Current](#)

Возвращает элемент коллекции, соответствующий текущей позиции перечислителя.

Методы

[Dispose\(\)](#)

Освобождает ресурсы, используемые этим экземпляром [JsonElement.ObjectEnumerator](#).

[GetEnumerator\(\)](#)

Возвращает перечислитель, который перебирает свойства объекта.

[MoveNext\(\)](#)

Перемещает перечислитель к следующему элементу коллекции.

Reset()	Устанавливает перечислитель в его начальное положение, т. е. перед первым элементом коллекции.
-------------------------	--

Явные реализации интерфейса

IEnumerable.GetEnumerator()	Возвращает перечислитель, который осуществляет итерацию по коллекции.
IEnumerable<JsonProperty>.GetEnumerator()	Возвращает перечислитель, который осуществляет итерацию по коллекции.
IEnumerator.Current	Возвращает элемент коллекции, соответствующий текущей позиции перечислителя.

Методы расширения

ToFrozenDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает из FrozenDictionary<TKey,TValue> объекта в соответствии с указанной IEnumerable<T> функцией селектора ключей.
ToFrozenDictionary<TSource,TKey,TElement> (IEnumerable<TSource> , Func<TSource,TKey> , Func<TSource,TElement> , IEqualityComparer<TKey>)	Создает словарь FrozenDictionary<TKey,TValue> из объекта IEnumerable<T> в соответствии с заданными функциями селектора ключа и селектора элемента.
ToFrozenSet<T> (IEnumerable<T> , IEqualityComparer<T>)	Создает с FrozenSet<T> указанными значениями.
ToImmutableArray<TSource> (IEnumerable<TSource>)	Создает неизменяемый массив на основе указанной коллекции.
ToImmutableDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey>)	Создает неизменяемый словарь на основе существующей коллекции элементов, применяя функцию преобразования к исходным ключам.
ToImmutableDictionary<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает неизменяемый словарь на основе последовательности, подвергнутой определенному преобразованию.
ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource> , Func<TSource,TKey> , Func<TSource,TValue>)	Выполняет перечисление и преобразование последовательности и создает неизменяемый словарь на основе ее содержимого.

<code>ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TKey>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый словарь с использованием указанной функции сравнения ключей.
<code>ToImmutableDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IEqualityComparer<TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый словарь с использованием указанных функций сравнения ключей и значений.
<code>ToImmutableHashSet<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает неизменяемый набор хэширования на основе ее содержимого.
<code>ToImmutableHashSet<TSource> (IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Выполняет перечисление последовательности, создает неизменяемый набор хэширования на основе ее содержимого и использует указанную функцию сравнения для типа набора.
<code>ToImmutableList<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает неизменяемый список на основе ее содержимого.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает неизменяемый отсортированный словарь на основе ее содержимого.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый отсортированный словарь с использованием указанной функции сравнения ключей.
<code>ToImmutableSortedDictionary<TSource,TKey,TValue> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TValue>, IComparer<TKey>, IEqualityComparer<TValue>)</code>	Выполняет перечисление и преобразование последовательности и создает на основе ее содержимого неизменяемый отсортированный словарь с использованием указанных функций сравнения ключей и значений.
<code>ToImmutableSortedSet<TSource> (IEnumerable<TSource>)</code>	Выполняет перечисление последовательности и создает

	неизменяемый отсортированный набор на основе ее содержимого.
ToImmutableSortedSet<TSource> (IEnumerable<TSource>, IComparer<TSource>)	Выполняет перечисление последовательности, создает неизменяемый отсортированный набор на основе ее содержимого и использует указанную функцию сравнения.
CopyToDataTable<T>(IEnumerable<T>)	Возвращает объект DataTable , содержащий копии объектов DataRow при заданном входном объекте IEnumerable<T> и универсальном параметре T , равном DataRow .
CopyToDataTable<T>(IEnumerable<T>, Data Table, LoadOption)	Копирует объекты DataRow в указанный объект DataTable с заданным входным объектом IEnumerable<T> , где универсальный параметр T имеет значение DataRow .
CopyToDataTable<T>(IEnumerable<T>, Data Table, LoadOption, FillErrorEventHandler)	Копирует объекты DataRow в указанный объект DataTable с заданным входным объектом IEnumerable<T> , где универсальный параметр T имеет значение DataRow .
Aggregate<TSource>(IEnumerable<TSource>, Func<TSource,TSource,TSource>)	Применяет к последовательности агрегатную функцию.
Aggregate<TSource,TAccumulate> (IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>)	Применяет к последовательности агрегатную функцию. Указанное начальное значение используется в качестве исходного значения агрегатной операции.
Aggregate<TSource,TAccumulate,TResult> (IEnumerable<TSource>, TAccumulate, Func<TAccumulate,TSource,TAccumulate>, Func<TAccumulate,TResult>)	Применяет к последовательности агрегатную функцию. Указанное начальное значение служит исходным значением для агрегатной операции, а указанная функция используется для выбора результирующего значения.
All<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Проверяет, все ли элементы последовательности удовлетворяют условию.
Any<TSource>(IEnumerable<TSource>)	Проверяет, содержит ли последовательность какие-либо элементы.
Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Проверяет, удовлетворяет ли какой-либо элемент последовательности заданному

	условию.
Append<TSource>(IEnumerable<TSource>, TSource)	Добавляет значение в конец последовательности.
AsEnumerable<TSource>(IEnumerable<TSource>)	Возвращает входное значение, типизированное как IEnumerable<T> .
Average<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вычисляет среднее для последовательности значений типа Decimal , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вычисляет среднее для последовательности значений типа Double , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вычисляет среднее для последовательности значений типа Int32 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вычисляет среднее для последовательности значений типа Int64 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Вычисляет среднее для последовательности значений Decimal обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Вычисляет среднее для последовательности значений Double обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Вычисляет среднее для последовательности значений Int32 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.

Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Вычисляет среднее для последовательности значений Int64 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Вычисляет среднее для последовательности значений Single обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Average<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Вычисляет среднее для последовательности значений типа Single , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Cast<TResult>(IEnumerable)	Приводит элементы объекта IEnumerable к заданному типу.
Chunk<TSource>(IEnumerable<TSource>, Int32)	Разбивает элементы последовательности на блоки размером не более size .
Concat<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Объединяет две последовательности.
Contains<TSource>(IEnumerable<TSource>, TSource)	Определяет, содержится ли указанный элемент в последовательности, используя компаратор проверки на равенство по умолчанию.
Contains<TSource>(IEnumerable<TSource>, TSource, IEqualityComparer<TSource>)	Определяет, содержит ли последовательность заданный элемент, используя указанный компаратор IEqualityComparer<T> .
Count<TSource>(IEnumerable<TSource>)	Возвращает количество элементов в последовательности.
Count<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Возвращает число, представляющее количество элементов последовательности, удовлетворяющих заданному условию.
DefaultIfEmpty<TSource>(IEnumerable<TSource>)	Возвращает элементы указанной последовательности или одноэлементную коллекцию, содержащую значение параметра типа по умолчанию, если последовательность пуста.

<code>DefaultIfEmpty<TSource> (IEnumerable<TSource>, TSource)</code>	Возвращает элементы указанной последовательности или одноэлементную коллекцию, содержащую указанное значение, если последовательность пуста.
<code>Distinct<TSource>(IEnumerable<TSource>)</code>	Возвращает различающиеся элементы последовательности, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Distinct<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Возвращает различающиеся элементы последовательности, используя для сравнения значений указанный компаратор <code>IEqualityComparer<T></code> .
<code>DistinctBy<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)</code>	Возвращает отдельные элементы из последовательности в соответствии с указанной функцией селектора ключей.
<code>DistinctBy<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)</code>	Возвращает отдельные элементы из последовательности в соответствии с указанной функцией селектора ключей и с помощью указанного компаратора для сравнения ключей.
<code>ElementAt<TSource>(IEnumerable<TSource>, Index)</code>	Возвращает элемент по указанному индексу в последовательности.
<code>ElementAt<TSource>(IEnumerable<TSource>, Int32)</code>	Возвращает элемент по указанному индексу в последовательности.
<code>ElementAtOrDefault<TSource> (IEnumerable<TSource>, Index)</code>	Возвращает элемент последовательности по указанному индексу или значение по умолчанию, если индекс вне допустимого диапазона.
<code>ElementAtOrDefault<TSource> (IEnumerable<TSource>, Int32)</code>	Возвращает элемент последовательности по указанному индексу или значение по умолчанию, если индекс вне допустимого диапазона.
<code>Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)</code>	Находит разность множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Except<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Находит разность множеств, представленных двумя последовательностями, используя для

	сравнения значений указанный компаратор IEqualityComparer<T> .
ExceptBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TKey> , Func<TSource,TKey>)	Создает разность наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
ExceptBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TKey> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает разность наборов двух последовательностей в соответствии с указанной функцией селектора ключей.
First<TSource> (IEnumerable<TSource>)	Возвращает первый элемент последовательности.
First<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Возвращает первый элемент последовательности, удовлетворяющий указанному условию.
FirstOrDefault<TSource> (IEnumerable<TSource>)	Возвращает первый элемент последовательности или значение по умолчанию, если последовательность не содержит элементов.
FirstOrDefault<TSource> (IEnumerable<TSource> , TSource)	Возвращает первый элемент последовательности или указанное значение по умолчанию, если последовательность не содержит элементов.
FirstOrDefault<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Возвращает первый элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если ни одного такого элемента не найдено.
FirstOrDefault<TSource> (IEnumerable<TSource> , Func<TSource,Boolean> , TSource)	Возвращает первый элемент последовательности, удовлетворяющий условию, или указанное значение по умолчанию, если такой элемент не найден.
GroupBy<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey>)	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа.
GroupBy<TSource,TKey> (IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и сравнивает ключи с помощью указанного компаратора.
GroupBy<TSource,TKey,TElement> (IEnumerable<TSource> , Func<TSource,TKey> ,	Группирует элементы последовательности в соответствии с заданной функцией

<code>Func<TSource,TElement>)</code>	селектора ключа и проецирует элементы каждой группы с помощью указанной функции.
<code>GroupBy<TSource,TKey,TElement></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с функцией селектора ключа. Ключи сравниваются с помощью компаратора, элементы каждой группы проецируются с помощью указанной функции.
<code>GroupBy<TSource,TKey,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TKey,IEnumerable<TSource>,TResult>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа.
<code>GroupBy<TSource,TKey,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TKey,IEnumerable<TSource>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Ключи сравниваются с использованием заданного компаратора.
<code>GroupBy<TSource,TKey,TElement,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>,</code> <code>Func<TKey,IEnumerable<TElement>,TResult>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Элементы каждой группы проецируются с помощью указанной функции.
<code>GroupBy<TSource,TKey,TElement,TResult></code> <code>(IEnumerable<TSource>, Func<TSource,TKey>,</code> <code>Func<TSource,TElement>,</code> <code>Func<TKey,IEnumerable<TElement>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Группирует элементы последовательности в соответствии с заданной функцией селектора ключа и создает результирующее значение для каждой группы и ее ключа. Значения ключей сравниваются с помощью указанного компаратора, элементы каждой группы проецируются с помощью указанной функции.
<code>GroupJoin<TOuter,TInner,TKey,TResult></code> <code>(IEnumerable<TOuter>, IEnumerable<TInner>,</code> <code>Func<TOuter,TKey>, Func<TInner,TKey>,</code> <code>Func<TOuter,IEnumerable<TInner>,TResult>)</code>	Устанавливает корреляцию между элементами двух последовательностей на основе равенства ключей и группирует результаты. Для сравнения ключей используется компаратор проверки на равенство по умолчанию.
<code>GroupJoin<TOuter,TInner,TKey,TResult></code> <code>(IEnumerable<TOuter>, IEnumerable<TInner>,</code> <code>Func<TOuter,TKey>, Func<TInner,TKey>,</code> <code>Func<TOuter,IEnumerable<TInner>,TResult>,</code> <code>IEqualityComparer<TKey>)</code>	Устанавливает корреляцию между элементами двух последовательностей на основе равенства ключей и группирует результаты. Для сравнения ключей

	используется указанный компаратор IEqualityComparer<T> .
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)	Находит пересечение множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
Intersect<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	Находит пересечение множеств, представленных двумя последовательностями, используя для сравнения значений указанный компаратор IEqualityComparer<T> .
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>)	Создает набор пересечения двух последовательностей в соответствии с указанной функцией селектора ключей.
IntersectBy<TSource,TKey>(IEnumerable<TSource>, IEnumerable<TKey>, Func<TSource,TKey>, IEqualityComparer<TKey>)	Создает набор пересечения двух последовательностей в соответствии с указанной функцией селектора ключей.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>)	Устанавливает корреляцию между элементами двух последовательностей на основе сопоставления ключей. Для сравнения ключей используется компаратор проверки на равенство по умолчанию.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>, IEqualityComparer<TKey>)	Устанавливает корреляцию между элементами двух последовательностей на основе сопоставления ключей. Для сравнения ключей используется указанный компаратор IEqualityComparer<T> .
Last<TSource>(IEnumerable<TSource>)	Возвращает последний элемент последовательности.
Last<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает последний элемент последовательности, удовлетворяющий указанному условию.
LastOrDefault<TSource>(IEnumerable<TSource>)	Возвращает последний элемент последовательности или значение по умолчанию, если последовательность не содержит элементов.
LastOrDefault<TSource>(IEnumerable<TSource>, TSource)	Возвращает последний элемент последовательности или указанное

	значение по умолчанию, если последовательность не содержит элементов.
LastOrDefault<TSource> (IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает последний элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если ни одного такого элемента не найдено.
LastOrDefault<TSource> (IEnumerable<TSource>, Func<TSource,Boolean>, TSource)	Возвращает последний элемент последовательности, удовлетворяющий условию, или указанное значение по умолчанию, если такой элемент не найден.
LongCount<TSource>(IEnumerable<TSource>)	Возвращает значение типа Int64 , представляющее общее число элементов в последовательности.
LongCount<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Возвращает значение типа Int64 , представляющее число элементов последовательности, удовлетворяющих заданному условию.
Max<TSource>(IEnumerable<TSource>)	Возвращает максимальное значение, содержащееся в универсальной последовательности.
Max<TSource>(IEnumerable<TSource>, IComparer<TSource>)	Возвращает максимальное значение, содержащееся в универсальной последовательности.
Max<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Decimal .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Double .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Int32 .
Max<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа Int64 .

<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Decimal>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Decimal</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Double>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Double</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int32>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Int32</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Int64</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение <code>Single</code> обнуляемого типа.
<code>Max<TSource>(IEnumerable<TSource>, Func<TSource, Single>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает максимальное значение типа <code>Single</code> .
<code>Max<TSource, TResult>(IEnumerable<TSource>, Func<TSource, TResult>)</code>	Вызывает функцию преобразования для каждого элемента универсальной последовательности и возвращает максимальное результирующее значение.
<code>MaxBy<TSource, TKey>(IEnumerable<TSource>, Func<TSource, TKey>)</code>	Возвращает максимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключа.
<code>MaxBy<TSource, TKey>(IEnumerable<TSource>, Func<TSource, TKey>, IComparer< TKey>)</code>	Возвращает максимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей и компаратором ключей.
<code>Min<TSource>(IEnumerable<TSource>)</code>	Возвращает минимальное значение, содержащееся в универсальной последовательности.
<code>Min<TSource>(IEnumerable<TSource>, IComparer<TSource>)</code>	Возвращает минимальное значение, содержащееся в универсальной

	последовательности.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Decimal</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Double>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Double</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Int32</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Int64</code> .
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Decimal</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Double</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Int32</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int64>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Int64</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Single>>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение <code>Single</code> обнуляемого типа.
<code>Min<TSource>(IEnumerable<TSource>, Func<TSource,Single>)</code>	Вызывает функцию преобразования для каждого элемента последовательности и возвращает минимальное значение типа <code>Single</code> .

<code>Min<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)</code>	Вызывает функцию преобразования для каждого элемента универсальной последовательности и возвращает минимальное результирующее значение.
<code>MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Возвращает минимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключа.
<code>MinBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Возвращает минимальное значение в универсальной последовательности в соответствии с указанной функцией селектора ключей и компаратором ключей.
<code>OfType<TResult>(IEnumerable)</code>	Выполняет фильтрацию элементов объекта <code>IEnumerable</code> по заданному типу.
<code>Order<T>(IEnumerable<T>)</code>	Сортирует элементы последовательности в порядке возрастания.
<code>Order<T>(IEnumerable<T>, IComparer<T>)</code>	Сортирует элементы последовательности в порядке возрастания.
<code>OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Сортирует элементы последовательности в порядке возрастания ключа.
<code>OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Сортирует элементы последовательности в порядке возрастания с использованием указанного компаратора.
<code>OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)</code>	Сортирует элементы последовательности в порядке убывания ключа.
<code>OrderByDescending<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>, IComparer<TKey>)</code>	Сортирует элементы последовательности в порядке убывания с использованием указанного компаратора.
<code>OrderDescending<T>(IEnumerable<T>)</code>	Сортирует элементы последовательности в порядке убывания.
<code>OrderDescending<T>(IEnumerable<T>, IComparer<T>)</code>	Сортирует элементы последовательности в порядке убывания.
<code>Prepend<TSource>(IEnumerable<TSource>, TSource)</code>	Добавляет значение в начало последовательности.
<code>Reverse<TSource>(IEnumerable<TSource>)</code>	Изменяет порядок элементов последовательности на противоположный.
<code>Select<TSource,TResult>(IEnumerable<TSource>, Func<TSource,TResult>)</code>	Проецирует каждый элемент последовательности в новую форму.

<code>Func<TSource, TResult>)</code>	
<code>Select<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, TResult>)</code>	Проецирует каждый элемент последовательности в новую форму, добавляя индекс элемента.
<code>SelectMany<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, IEnumerable<TResult>>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> и объединяет результирующие последовательности в одну последовательность.
<code>SelectMany<TSource, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, IEnumerable<TResult>>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> и объединяет результирующие последовательности в одну последовательность. Индекс каждого элемента исходной последовательности используется в проецированной форме этого элемента.
<code>SelectMany<TSource, TCollection, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, IEnumerable<TCollection>>,</code> <code>Func<TSource, TCollection, TResult>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> , объединяет результирующие последовательности в одну и вызывает функцию селектора результата для каждого элемента этой последовательности.
<code>SelectMany<TSource, TCollection, TResult></code> <code>(IEnumerable<TSource>,</code> <code>Func<TSource, Int32, IEnumerable<TCollection>>,</code> <code>Func<TSource, TCollection, TResult>)</code>	Проецирует каждый элемент последовательности в объект <code>IEnumerable<T></code> , объединяет результирующие последовательности в одну и вызывает функцию селектора результата для каждого элемента этой последовательности. Индекс каждого элемента исходной последовательности используется в промежуточной проецированной форме этого элемента.
<code>SequenceEqual<TSource></code> <code>(IEnumerable<TSource>,</code> <code>IEnumerable<TSource>)</code>	Определяет, совпадают ли две последовательности, используя для сравнения элементов компаратор проверки на равенство по умолчанию, предназначенный для их типа.
<code>SequenceEqual<TSource></code> <code>(IEnumerable<TSource>,</code> <code>IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Определяет, совпадают ли две последовательности, используя для сравнения элементов указанный компаратор <code>IEqualityComparer<T></code> .

<code>Single<TSource>(IEnumerable<TSource>)</code>	Возвращает единственный элемент последовательности и генерирует исключение, если число элементов последовательности отлично от 1.
<code>Single<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, и вызывает исключение, если таких элементов больше одного.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>)</code>	Возвращает единственный элемент последовательности или значение по умолчанию, если последовательность пуста; если в последовательности более одного элемента, генерируется исключение.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, TSource)</code>	Возвращает единственный элемент последовательности или указанное значение по умолчанию, если последовательность пуста; Этот метод создает исключение, если в последовательности имеется несколько элементов.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, или значение по умолчанию, если такого элемента не существует; если условию удовлетворяет более одного элемента, вызывается исключение.
<code>SingleOrDefault<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>, TSource)</code>	Возвращает единственный элемент последовательности, удовлетворяющий указанному условию, или указанное значение по умолчанию, если такого элемента не существует; Этот метод создает исключение, если условию удовлетворяет несколько элементов.
<code>Skip<TSource>(IEnumerable<TSource>, Int32)</code>	Пропускает заданное число элементов в последовательности и возвращает остальные элементы.
<code>SkipLast<TSource>(IEnumerable<TSource>, Int32)</code>	Возвращает новую перечислимую коллекцию, содержащую элементы из <code>source</code> с исключенными <code>count</code> элементами из конца исходной коллекции.

SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы.
SkipWhile<TSource>(IEnumerable<TSource>, Func<TSource,Int32,Boolean>)	Пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы. Индекс элемента используется в логике функции предиката.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Decimal>)	Вычисляет сумму последовательности значений типа Decimal , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Double>)	Вычисляет сумму последовательности значений типа Double , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int32>)	Вычисляет сумму последовательности значений типа Int32 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Int64>)	Вычисляет сумму последовательности значений типа Int64 , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Decimal>>)	Вычисляет сумму последовательности значений Decimal обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Double>>)	Вычисляет сумму последовательности значений Double обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource,Nullable<Int32>>)	Вычисляет сумму последовательности значений Int32 обнуляемого типа, получаемой в результате применения

	функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Int64>>)	Вычисляет сумму последовательности значений Int64 обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Nullable<Single>>)	Вычисляет сумму последовательности значений Single обнуляемого типа, получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Sum<TSource>(IEnumerable<TSource>, Func<TSource, Single>)	Вычисляет сумму последовательности значений типа Single , получаемой в результате применения функции преобразования к каждому элементу входной последовательности.
Take<TSource>(IEnumerable<TSource>, Int32)	Возвращает указанное число подряд идущих элементов с начала последовательности.
Take<TSource>(IEnumerable<TSource>, Range)	Возвращает указанный диапазон смежных элементов из последовательности.
TakeLast<TSource>(IEnumerable<TSource>, Int32)	Возвращает новую перечислимую коллекцию, содержащую последние <code>count</code> элементов из <code>source</code> .
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Boolean>)	Возвращает цепочку элементов последовательности, удовлетворяющих указанному условию.
TakeWhile<TSource>(IEnumerable<TSource>, Func<TSource, Int32, Boolean>)	Возвращает цепочку элементов последовательности, удовлетворяющих указанному условию. Индекс элемента используется в логике функции предиката.
ToArray<TSource>(IEnumerable<TSource>)	Создает массив из объекта IEnumerable<T> .
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>)	Создает словарь Dictionary<TKey, TValue> из объекта IEnumerable<T> в соответствии с заданной функцией селектора ключа.
ToDictionary<TSource, TKey> (IEnumerable<TSource>, Func<TSource, TKey>, IEqualityComparer<TKey>)	Создает словарь Dictionary<TKey, TValue> из объекта IEnumerable<T> в соответствии с заданной функцией селектора ключа и компаратором ключей.

<code>ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)</code>	Создает словарь <code>Dictionary<TKey,TValue></code> из объекта <code>IEnumerable<T></code> в соответствии с заданными функциями селектора ключа и селектора элемента.
<code>ToDictionary<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Создает словарь <code>Dictionary<TKey,TValue></code> из объекта <code>IEnumerable<T></code> в соответствии с заданным компаратором и функциями селектора ключа и селектора элемента.
<code>ToHashSet<TSource>(IEnumerable<TSource>)</code>	Создает <code>HashSet<T></code> из <code>IEnumerable<T></code> .
<code>ToHashSet<TSource>(IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Создает <code>HashSet<T></code> из <code>IEnumerable<T></code> с использованием <code>comparer</code> для сравнения ключей.
<code>ToList<TSource>(IEnumerable<TSource>)</code>	Создает <code>List<T></code> из <code>IEnumerable<T></code> .
<code>ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданной функцией селектора ключа.
<code>ToLookup<TSource,TKey> (IEnumerable<TSource>, Func<TSource,TKey>, IEqualityComparer<TKey>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданной функцией селектора ключа и компаратором ключей.
<code>ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>)</code>	Создает словарь <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданными функциями селектора ключа и селектора элемента.
<code>ToLookup<TSource,TKey,TElement> (IEnumerable<TSource>, Func<TSource,TKey>, Func<TSource,TElement>, IEqualityComparer<TKey>)</code>	Создает объект <code>Lookup<TKey,TElement></code> из объекта <code>IEnumerable<T></code> в соответствии с заданным компаратором и функциями селектора ключа и селектора элемента.
<code>TryGetNonEnumeratedCount<TSource> (IEnumerable<TSource>, Int32)</code>	Пытается определить количество элементов в последовательности без принудительного перечисления.
<code>Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>)</code>	Находит объединение множеств, представленных двумя последовательностями, используя для сравнения значений компаратор проверки на равенство по умолчанию.
<code>Union<TSource>(IEnumerable<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)</code>	Находит объединение множеств, представленных двумя последовательностями, используя

	указанный компаратор IEqualityComparer<T> .
UnionBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TSource> , Func<TSource,TKey>)	Создает объединение наборов двух последовательностей в соответствии с указанной функцией селектора ключа.
UnionBy<TSource,TKey> (IEnumerable<TSource> , IEnumerable<TSource> , Func<TSource,TKey> , IEqualityComparer<TKey>)	Создает объединение наборов двух последовательностей в соответствии с указанной функцией селектора ключа.
Where<TSource> (IEnumerable<TSource> , Func<TSource,Boolean>)	Выполняет фильтрацию последовательности значений на основе заданного предиката.
Where<TSource> (IEnumerable<TSource> , Func<TSource,Int32,Boolean>)	Выполняет фильтрацию последовательности значений на основе заданного предиката. Индекс каждого элемента используется в логике функции предиката.
Zip<TFirst,TSecond> (IEnumerable<TFirst> , IEnumerable<TSecond>)	Создает последовательность кортежей с элементами из двух указанных последовательностей.
Zip<TFirst,TSecond,TThird> (IEnumerable<TFirst> , IEnumerable<TSecond> , IEnumerable<TThird>)	Создает последовательность кортежей с элементами из трех указанных последовательностей.
Zip<TFirst,TSecond,TResult> (IEnumerable<TFirst> , IEnumerable<TSecond> , Func<TFirst,TSecond,TResult>)	Применяет указанную функцию к соответствующим элементам двух последовательностей, что дает последовательность результатов.
AsParallel(IEnumerable)	Позволяет осуществлять параллельный запрос.
AsParallel<TSource> (IEnumerable<TSource>)	Позволяет осуществлять параллельный запрос.
AsQueryable(IEnumerable)	Преобразовывает коллекцию IEnumerable в объект IQueryable .
AsQueryable<TElement> (IEnumerable<TElement>)	Преобразовывает универсальный объект IEnumerable<T> в универсальный объект IQueryable<T> .
Ancestors<T> (IEnumerable<T>)	Возвращает коллекцию элементов, содержащую предков каждого узла в исходной коллекции.

Ancestors<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию элементов, содержащую предков каждого узла в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
DescendantNodes<T>(IEnumerable<T>)	Возвращает коллекцию подчиненных узлов каждого документа и элемента в исходной коллекции.
Descendants<T>(IEnumerable<T>)	Возвращает коллекцию элементов, содержащую подчиненные элементы каждого элемента и документа в исходной коллекции.
Descendants<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию элементов, содержащую подчиненные элементы каждого элемента и документа в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
Elements<T>(IEnumerable<T>)	Возвращает коллекцию дочерних элементов каждого элемента и документа в исходной коллекции.
Elements<T>(IEnumerable<T>, XName)	Возвращает отфильтрованную коллекцию дочерних элементов каждого элемента и документа в исходной коллекции. В коллекцию включаются только элементы, соответствующие XName .
InDocumentOrder<T>(IEnumerable<T>)	Возвращает коллекцию узлов, содержащую все узлы в исходной коллекции, отсортированные в порядке следования документов.
Nodes<T>(IEnumerable<T>)	Возвращает коллекцию дочерних узлов каждого документа и элемента в исходной коллекции.
Remove<T>(IEnumerable<T>)	Удаление каждого узла в исходной коллекции из родительского узла.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ObjectEnumerator.Current Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the element in the collection at the current position of the enumerator.

C#

```
public System.Text.Json.JsonProperty Current { get; }
```

Property Value

[JsonProperty](#)

The element in the collection at the current position of the enumerator.

Implements

[Current](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ObjectEnumerator.Dispose

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Освобождает ресурсы, используемые этим экземпляром [JsonElement.ObjectEnumerator](#).

C#

```
public void Dispose();
```

Реализации

[Dispose\(\)](#)

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ObjectEnumerator.Get Enumerator Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает перечислитель, который перебирает свойства объекта.

C#

```
public System.Text.Json.JsonElement.ObjectEnumerator GetEnumerator();
```

Возвращаемое значение

[JsonElement.ObjectEnumerator](#)

Перечислитель, который можно использовать для перебора элементов объекта.

Комментарии

Перечислитель перечисляет свойства в порядке их объявления, и если объект имеет несколько определений одного свойства, они будут возвращены по отдельности (каждый в том порядке, в который они отображаются в содержимом).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ObjectEnumerator.MoveNext Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Advances the enumerator to the next element of the collection.

C#

```
public bool MoveNext();
```

Returns

[Boolean](#)

`true` if the enumerator was successfully advanced to the next element; `false` if the enumerator has passed the end of the collection.

Implements

[MoveNext\(\)](#)

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ObjectEnumerator.Reset

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Устанавливает перечислитель в его начальное положение, т. е. перед первым элементом коллекции.

C#

```
public void Reset();
```

Реализации

[Reset\(\)](#)

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ObjectEnumerator. IEnumerable<JsonProperty>.Get Enumerator Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает перечислитель, который осуществляет итерацию по коллекции.

C#

```
System.Collections.Generic.IEnumerable<System.Text.Json.JsonProperty>
IEnumerable<JsonProperty>.GetEnumerator ();
```

Возвращаемое значение

[IEnumerator<JsonProperty>](#)

Перечислитель объектов [JsonProperty](#), который может использоваться для итерации по коллекции.

Реализации

[GetEnumerator\(\)](#)

Комментарии

Этот член представляет собой явную реализацию члена интерфейса. Его можно использовать только в [JsonElement.ObjectEnumerator](#) том случае, если экземпляр приведен к интерфейсу [JsonProperty](#) [IEnumerator<](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonElement.ObjectEnumerator. IEnumerable.GetEnumerator Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Returns an enumerator that iterates through a collection.

C#

```
System.Collections.IEnumerator IEnumerable.GetEnumerator();
```

Returns

[IEnumerator](#)

An enumerator that can be used to iterate through the collection.

Implements

[GetEnumerator\(\)](#)

Remarks

This member is an explicit interface member implementation. It can be used only when the [JsonElement.ObjectEnumerator](#) instance is cast to an [IEnumerable](#) interface.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonElement.ObjectEnumerator. IEnumerator.Current Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает элемент коллекции, соответствующий текущей позиции перечислителя.

C#

```
object System.Collections.IEnumerator.Current { get; }
```

Значение свойства

[Object](#)

Элемент коллекции, соответствующий текущей позиции перечислителя.

Реализации

[Current](#)

Комментарии

Этот член представляет собой явную реализацию члена интерфейса. Он может использоваться, только если экземпляр [JsonElement.ObjectEnumerator](#) приведен к типу интерфейса [IEnumerator](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonEncodedText Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет методы для преобразования текста в кодировке UTF-8 или UTF-16 в форму, подходящую для JSON.

C#

```
public readonly struct JsonEncodedText :  
IEquatable<System.Text.Json.JsonEncodedText>
```

Наследование [Object](#) → [ValueType](#) → [JsonEncodedText](#)

Реализации [IEquatable<JsonEncodedText>](#)

Комментарии

Этот тип можно использовать для кэширования и хранения известных строк, используемых для записи JSON заранее, предварительно закодируя их заранее.

Свойства

[EncodedUtf8Bytes](#)

Возвращает представление предварительно закодированного текста JSON в кодировке UTF-8.

[Value](#)

Возвращает представление предварительно закодированного текста JSON в кодировке UTF-16 в [String](#) виде .

Методы

[Encode\(ReadOnlySpan<Byte>, JavaScriptEncoder\)](#) Кодирует текстовое значение UTF-8 в виде строки JSON.

Encode(ReadOnlySpan<Char>, JavaScriptEncoder)	Кодирует указанное текстовое значение в строку JSON.
Encode(String, JavaScriptEncoder)	Кодирует строковое текстовое значение в виде строки JSON.
Equals(JsonEncodedText)	Определяет, равны ли значения этого экземпляра и указанного экземпляра JsonEncodedText.
Equals(Object)	Определяет, равны ли значения этого экземпляра и указанного объекта, который также должен быть экземпляром JsonEncodedText.
GetHashCode()	Возвращает хэш-код для модуля чтения данных JsonEncodedText.
ToString()	Преобразует значение данного экземпляра в String.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonEncodedText.EncodedUtf8Bytes Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the UTF-8 encoded representation of the pre-encoded JSON text.

C#

```
public ReadOnlySpan<byte> EncodedUtf8Bytes { get; }
```

Property Value

[ReadOnlySpan<Byte>](#)

The UTF-8 encoded representation of the pre-encoded JSON text.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonEncodedText.Value Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает представление предварительно закодированного текста JSON в кодировке UTF-16 в [String](#) виде .

C#

```
public string Value { get; }
```

Значение свойства

[String](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonEncodedText.Encode Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

<code>Encode(ReadOnlySpan<Byte>, JavaScriptEncoder)</code>	Кодирует текстовое значение UTF-8 в виде строки JSON.
--	---

<code>Encode(ReadOnlySpan<Char>, JavaScriptEncoder)</code>	Кодирует указанное текстовое значение в строку JSON.
--	--

<code>Encode(String, JavaScriptEncoder)</code>	Кодирует строковое текстовое значение в виде строки JSON.
--	---

Encode(ReadOnlySpan<Byte>, JavaScriptEncoder)

Кодирует текстовое значение UTF-8 в виде строки JSON.

C#

```
public static System.Text.Json.JsonEncodedText Encode (ReadOnlySpan<byte> utf8Value, System.Text.Encodings.Web.JavaScriptEncoder? encoder = default);
```

Параметры

utf8Value `ReadOnlySpan<Byte>`

Текст в кодировке UTF-8 для преобразования в текст в кодировке JSON.

encoder `JavaScriptEncoder`

Кодировщик, используемый при экранировании строки. Укажите значение `null` для использования кодировщика по умолчанию.

Возвращаемое значение

JsonEncodedText

Текст в кодировке JSON.

Исключения

ArgumentException

`utf8Value` — слишком большое значение.

-ИЛИ-

`utf8Value` содержит недопустимые байты UTF-8.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Encode(ReadOnlySpan<Char>, JavaScriptEncoder)

Кодирует указанное текстовое значение в строку JSON.

C#

```
public static System.Text.Json.JsonEncodedText Encode (ReadOnlySpan<char>  
value, System.Text_ENCODINGS.Web.JavaScriptEncoder? encoder = default);
```

Параметры

value `ReadOnlySpan<Char>`

Значение для преобразования в текст в кодировке JSON.

encoder `JavaScriptEncoder`

Кодировщик, используемый при экранировании строки. Укажите значение `null` для использования кодировщика по умолчанию.

Возвращаемое значение

JsonEncodedText

Текст в кодировке JSON.

Исключения

ArgumentException

`value` — слишком большое значение.

-ИЛИ-

`value` содержит недопустимые символы UTF-16.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Encode(String, JavaScriptEncoder)

Кодирует строковое текстовое значение в виде строки JSON.

C#

```
public static System.Text.Json.JsonEncodedText Encode (string value,  
System.Text.Encodings.Web.JavaScriptEncoder? encoder = default);
```

Параметры

value String

Значение для преобразования в текст в кодировке JSON.

encoder JavaScriptEncoder

Кодировщик, используемый при экранировании строки. Укажите значение `null` для использования кодировщика по умолчанию.

Возвращаемое значение

JsonEncodedText

Текст в кодировке JSON.

Исключения

ArgumentNullException

`value` имеет значение `null`.

ArgumentException

`value` — слишком большое значение.

-ИЛИ-

`value` содержит недопустимые символы UTF-16.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonEncodedText.Equals Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

Equals(Object)	Determines whether this instance and a specified object, which must also be a JsonEncodedText instance, have the same value.
Equals(JsonEncodedText)	Determines whether this instance and another specified JsonEncodedText instance have the same value.

Equals(Object)

Determines whether this instance and a specified object, which must also be a [JsonEncodedText](#) instance, have the same value.

C#

```
public override bool Equals (object? obj);
```

Parameters

obj [Object](#)

The object to compare to this instance.

Returns

[Boolean](#)

`true` if the current instance and `obj` are equal; otherwise, `false`.

Remarks

If `obj` is `null`, the method returns `false`.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Equals(JsonEncodedText)

Determines whether this instance and another specified [JsonEncodedText](#) instance have the same value.

C#

```
public bool Equals (System.Text.Json.JsonEncodedText other);
```

Parameters

other [JsonEncodedText](#)

The object to compare to this instance.

Returns

[Boolean](#)

`true` if this instance and `other` have the same value; otherwise, `false`.

Implements

[Equals\(T\)](#)

Remarks

Default instances of [JsonEncodedText](#) are treated as equal.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonEncodedText.GetHashCode Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает хэш-код для модуля чтения данных [JsonEncodedText](#).

C#

```
public override int GetHashCode();
```

Возвращаемое значение

[Int32](#)

Хэш-код данного экземпляра.

Комментарии

Этот метод возвращает значение 0 для экземпляра [JsonEncodedText](#) по умолчанию .

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonEncodedText.ToString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Преобразует значение данного экземпляра в [String](#).

C#

```
public override string ToString();
```

Возвращаемое значение

[String](#)

Базовая строка в кодировке UTF-16.

Комментарии

Этот метод возвращает пустую строку в экземпляре [JsonEncodedText](#) по умолчанию.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonException Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Определяет пользовательский объект исключения, который возникает при обнаружении недопустимого текста JSON, передаче заданной максимальной глубины или при несовместимом тексте JSON с типом свойства объекта.

C#

```
public class JsonException : Exception
```

Наследование [Object](#) → [Exception](#) → [JsonException](#)

Комментарии

Дополнительные сведения см. в статье [Создание пользовательских преобразователей для сериализации JSON](#).

Конструкторы

JsonException()	Инициализирует новый экземпляр класса JsonException .
JsonException(Serialization Info, StreamingContext)	Является устаревшей. Создает новый объект исключения с сериализованными данными.
JsonException(String)	Инициализирует новый экземпляр класса JsonException с указанным сообщением об ошибке.
JsonException(String, Exception)	Инициализирует новый экземпляр класса JsonException с указанным сообщением об ошибке и ссылкой на внутреннее исключение, вызвавшее это исключение.
JsonException(String, String, Nullable<Int64>, Nullable<Int64>)	Создает объект исключения для передачи сведений об ошибке пользователю.

<code>JsonException(String, String, Nullable<Int64>, Nullable<Int64>, Exception)</code>	Создает объект исключения для передачи сведений об ошибке пользователю, включающий указанное внутреннее исключение.
---	---

Свойства

<code>BytePositionInLine</code>	Возвращает число байтов, считанных в пределах текущей строки (начиная с 0) перед исключением.
<code>Data</code>	Возвращает коллекцию пар «ключ-значение», предоставляющую дополнительные сведения об исключении. (Унаследовано от Exception)
<code>HelpLink</code>	Получает или задает ссылку на файл справки, связанный с этим исключением. (Унаследовано от Exception)
<code>HResult</code>	Возвращает или задает HRESULT — кодированное числовое значение, присвоенное определенному исключению. (Унаследовано от Exception)
<code>InnerException</code>	Возвращает экземпляр класса Exception , который вызвал текущее исключение. (Унаследовано от Exception)
<code>LineNumber</code>	Возвращает число считанных строк (начиная с 0) перед исключением.
<code>Message</code>	Возвращает сообщение, описывающее текущее исключение.
<code>Path</code>	Возвращает путь в JSON, где было обнаружено исключение.
<code>Source</code>	Возвращает или задает имя приложения или объекта, вызывавшего ошибку. (Унаследовано от Exception)
<code>StackTrace</code>	Получает строковое представление непосредственных кадров в стеке вызова. (Унаследовано от Exception)
<code>TargetSite</code>	Возвращает метод, создавший текущее исключение. (Унаследовано от Exception)

Методы

<code>Equals(Object)</code>	Определяет, равен ли указанный объект текущему объекту.
-----------------------------	---

(Унаследовано от [Object](#))

GetBaseException()	При переопределении в производном классе возвращает исключение Exception , которое является первопричиной одного или нескольких последующих исключений. (Унаследовано от Exception)
GetHashCode()	Служит хэш-функцией по умолчанию. (Унаследовано от Object)
GetObjectData(SerializationInfo, StreamingContext)	Является устаревшей. Заполняет объект SerializationInfo сведениями об исключении.
GetType()	Возвращает тип среды выполнения текущего экземпляра. (Унаследовано от Exception)
MemberwiseClone()	Создает неполную копию текущего объекта Object . (Унаследовано от Object)
ToString()	Создает и возвращает строковое представление текущего исключения. (Унаследовано от Exception)

События

SerializeObjectState	Является устаревшей. Возникает, когда исключение сериализовано для создания объекта состояния исключения, содержащего сериализованные данные об исключении. (Унаследовано от Exception)
--------------------------------------	---

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonException Конструкторы

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

JsonException()	Инициализирует новый экземпляр класса JsonException .
JsonException(String)	Инициализирует новый экземпляр класса JsonException с указанным сообщением об ошибке.
JsonException(SerializationInfo, StreamingContext)	Является устаревшей. Создает новый объект исключения с сериализованными данными.
JsonException(String, Exception)	Инициализирует новый экземпляр класса JsonException с указанным сообщением об ошибке и ссылкой на внутреннее исключение, вызвавшее это исключение.
JsonException(String, String, Nullable<Int64>, Nullable<Int64>)	Создает объект исключения для передачи сведений об ошибке пользователю.
JsonException(String, String, Nullable<Int64>, Nullable<Int64>, Exception)	Создает объект исключения для передачи сведений об ошибке пользователю, включающий указанное внутреннее исключение.

JsonException()

Инициализирует новый экземпляр класса [JsonException](#).

C#

```
public JsonException();
```

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonException(String)

Инициализирует новый экземпляр класса [JsonException](#) с указанным сообщением об ошибке.

C#

```
public JsonException (string? message);
```

Параметры

message [String](#)

Контекстно-зависимое сообщение об ошибке.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonException(SerializationInfo, StreamingContext)

⊗ **Внимание!**

This API supports obsolete formatter-based serialization. It should not be called or extended by application code.

Создает новый объект исключения с сериализованными данными.

C#

```
[System.Obsolete("This API supports obsolete formatter-based
serialization. It should not be called or extended by application code.",
DiagnosticId="SYSLIB0051", UrlFormat="https://aka.ms/dotnet-
warnings/{0}")]
protected JsonException (System.Runtime.Serialization.SerializationInfo
info, System.Runtime.Serialization.StreamingContext context);
```

Параметры

info [SerializationInfo](#)

Данные сериализованного объекта о вызываемом исключении.

context [StreamingContext](#)

Объект, содержащий контекстные сведения об источнике или назначении.

Атрибуты [ObsoleteAttribute](#)

Исключения

[ArgumentNullException](#)

info имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии (Устарело)
.NET	Core 3.0, Core 3.1, 5, 6, 7 (8)

JsonException(String, Exception)

Инициализирует новый экземпляр класса [JsonException](#) с указанным сообщением об ошибке и ссылкой на внутреннее исключение, вызвавшее это исключение.

C#

```
public JsonException (string? message, Exception? innerException);
```

Параметры

message	<code>String</code>
----------------	---------------------

Контекстно-зависимое сообщение об ошибке.

innerException	<code>Exception</code>
-----------------------	------------------------

Исключение, которое вызвало текущее исключение.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonException(String, String, Nullable<Int64>, Nullable<Int64>)

Создает объект исключения для передачи сведений об ошибке пользователю.

C#

```
public JsonException (string? message, string? path, long? lineNumber,
long? bytePositionInLine);
```

Параметры

message	<code>String</code>
----------------	---------------------

Контекстно-зависимое сообщение об ошибке.

path	<code>String</code>
-------------	---------------------

Путь, по которому был обнаружен недопустимый JSON.

lineNumber	<code>Nullable<Int64></code>
-------------------	------------------------------------

Номер строки (начиная с 0), где обнаружен недопустимый JSON при десериализации.

bytePositionInLine	<code>Nullable<Int64></code>
---------------------------	------------------------------------

Число байтов (начиная с 0) в текущей строке, где обнаружен недопустимый JSON.

Комментарии

Обратите внимание, что `bytePositionInLine` подсчитывает количество байтов (т. е. единиц кода UTF-8), а не символов или скалярных знаков.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonException(String, String, Nullable<Int64>, Nullable<Int64>, Exception)

Создает объект исключения для передачи сведений об ошибке пользователю, включающий указанное внутреннее исключение.

C#

```
public JsonException (string? message, string? path, long? lineNumber,
long? bytePositionInLine, Exception? innerException);
```

Параметры

message `String`

Контекстно-зависимое сообщение об ошибке.

path `String`

Путь, по которому был обнаружен недопустимый JSON.

lineNumber `Nullable<Int64>`

Номер строки (начиная с 0), где обнаружен недопустимый JSON при десериализации.

bytePositionInLine `Nullable<Int64>`

Число байтов (начиная с 0) в текущей строке, где обнаружен недопустимый JSON.

innerException [Exception](#)

Исключение, которое вызвало текущее исключение.

Комментарии

Обратите внимание, что `bytePositionInLine` подсчитывает количество байтов (т. е. единиц кода UTF-8), а не символов или скалярных знаков.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonException.BytePositionInLine Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the zero-based number of bytes read within the current line before the exception.

C#

```
public long? BytePositionInLine { get; }
```

Property Value

[Nullable<Int64>](#)

The zero-based number of bytes read within the current line before the exception.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonException.LineNumber Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the zero-based number of lines read before the exception.

C#

```
public long? LineNumber { get; }
```

Property Value

[Nullable<Int64>](#)

The zero-based number of lines read before the exception.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonException.Message Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets a message that describes the current exception.

C#

```
public override string Message { get; }
```

Property Value

[String](#)

The error message that describes the current exception.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonException.Path Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets The path within the JSON where the exception was encountered.

C#

```
public string? Path { get; }
```

Property Value

[String](#)

The path within the JSON where the exception was encountered.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonException.GetObjectData(SerializationInfo, StreamingContext) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

⊗ Внимание!

This API supports obsolete formatter-based serialization. It should not be called or extended by application code.

Заполняет объект [SerializationInfo](#) сведениями об исключении.

C#

```
[System.Obsolete("This API supports obsolete formatter-based serialization.  
It should not be called or extended by application code.",  
DiagnosticId="SYSLIB0051", UrlFormat="https://aka.ms/dotnet-warnings/{0}")]  
public override void GetObjectData  
(System.Runtime.Serialization.SerializationInfo info,  
System.Runtime.Serialization.StreamingContext context);
```

Параметры

info [SerializationInfo](#)

Данные сериализованного объекта о вызываемом исключении.

context [StreamingContext](#)

Объект, содержащий контекстные сведения об источнике или назначении.

Атрибуты [ObsoleteAttribute](#)

Применяется к

Продукт	Версии (Устарело)
.NET	Core 3.0, Core 3.1, 5, 6, 7 (8)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Определяет политику именования, используемую для преобразования строкового имени в другой формат, например Camel.

C#

```
public abstract class JsonNamingPolicy
```

Наследование [Object](#) → JsonNamingPolicy

Комментарии

Дополнительные сведения см. в статье [Настройка имен и значений свойств с помощью System.Text.Json](#).

Конструкторы

[JsonNamingPolicy\(\)](#)

Инициализирует новый экземпляр [JsonNamingPolicy](#).

Свойства

CamelCase	Возвращает политику именования для Camel.
KebabCaseLower	Возвращает политику именования для кебаб- регистра в нижнем регистре.
KebabCaseUpper	Возвращает политику именования для кебаб- регистра в верхнем регистре.
SnakeCaseLower	Возвращает политику именования для строчных регистров.
SnakeCaseUpper	Возвращает политику именования для верхнего регистра со змеиным регистром.

Методы

ConvertName(String)	Если переопределено в производном классе, преобразует указанное имя согласно политике.
Equals(Object)	Определяет, равен ли указанный объект текущему объекту. (Унаследовано от Object)
GetHashCode()	Служит хэш-функцией по умолчанию. (Унаследовано от Object)
GetType()	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object)
MemberwiseClone()	Создает неполную копию текущего объекта Object . (Унаследовано от Object)
ToString()	Возвращает строку, представляющую текущий объект. (Унаследовано от Object)

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy Constructor

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Initializes a new instance of [JsonNamingPolicy](#).

C#

```
protected JsonNamingPolicy();
```

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonNamingPolicy.CamelCase Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает политику именования для Camel.

C#

```
public static System.Text.Json.JsonNamingPolicy CamelCase { get; }
```

Значение свойства

[JsonNamingPolicy](#)

Политика именования для верблюжьего регистра.

Комментарии

Дополнительные сведения см. в статье [Использование встроенной политики именования](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy.KebabCaseLower

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает политику именования для нижнего регистра kebab-casing.

C#

```
public static System.Text.Json.JsonNamingPolicy KebabCaseLower { get; }
```

Значение свойства

[JsonNamingPolicy](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy.KebabCaseUpper

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает политику именования для кебаб-регистра в верхнем регистре.

C#

```
public static System.Text.Json.JsonNamingPolicy KebabCaseUpper { get; }
```

Значение свойства

[JsonNamingPolicy](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy.SnakeCaseLower

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает политику именования для строчных регистров.

C#

```
public static System.Text.Json.JsonNamingPolicy SnakeCaseLower { get; }
```

Значение свойства

[JsonNamingPolicy](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy.SnakeCaseUpper

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает политику именования для верхнего регистра со змеиным регистром.

C#

```
public static System.Text.Json.JsonNamingPolicy SnakeCaseUpper { get; }
```

Значение свойства

[JsonNamingPolicy](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

Да

Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonNamingPolicy.ConvertName(String)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Если переопределено в производном классе, преобразует указанное имя согласно политике.

C#

```
public abstract string ConvertName (string name);
```

Параметры

name [String](#)

Имя для преобразования.

Возвращаемое значение

[String](#)

Преобразованное имя.

Комментарии

Дополнительные сведения см. в статье [Настройка имен и значений свойств с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Представляет одно свойство объекта JSON.

C#

```
public readonly struct JsonProperty
```

Наследование [Object](#) → [ValueType](#) → [JsonProperty](#)

Свойства

Name	Получает имя данного свойства.
Value	Возвращает значение свойства.

Методы

NameEquals(ReadOnlySpan<Byte>)	Сравнивает указанный текст в кодировке UTF-8 с именем этого свойства.
NameEquals(ReadOnlySpan<Char>)	Сравнивает указанный текст как диапазон символов с именем этого свойства.
NameEquals(String)	Сравнивает указанную строку с именем этого свойства.
ToString()	Предоставляет строковое представление свойства для целей отладки.
WriteTo(Utf8JsonWriter)	Записывает свойство в предоставленный модуль записи как свойство именованного объекта JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty.Name Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Получает имя данного свойства.

C#

```
public string Name { get; }
```

Значение свойства

[String](#)

Имя данного свойства.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty.Value Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение свойства.

C#

```
public System.Text.Json.JsonElement Value { get; }
```

Значение свойства

[JsonElement](#)

Значение этого свойства.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty.NameEquals Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

NameEquals(ReadOnlySpan<Byte>)	Сравнивает указанный текст в кодировке UTF-8 с именем этого свойства.
NameEquals(ReadOnlySpan<Char>)	Сравнивает указанный текст как диапазон символов с именем этого свойства.
NameEquals(String)	Сравнивает указанную строку с именем этого свойства.

NameEquals(ReadOnlySpan<Byte>)

Сравнивает указанный текст в кодировке UTF-8 с именем этого свойства.

C#

```
public bool NameEquals (ReadOnlySpan<byte> utf8Text);
```

Параметры

utf8Text [ReadOnlySpan<Byte>](#)

Текст в кодировке UTF-8, с которым производится сравнение.

Возвращаемое значение

[Boolean](#)

Значение `true`, если имя этого свойства имеет ту же кодировку UTF-8, что и `utf8Text`; в противном случае — значение `false`.

Исключения

InvalidOperationException

Type этого значения не является [PropertyName](#).

Комментарии

Этот метод функционально равен выполнению порядкового `utf8Text` сравнения и [Name](#), но он может избежать создания экземпляра строки.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

NameEquals(ReadOnlySpan<Char>)

Сравнивает указанный текст как диапазон символов с именем этого свойства.

C#

```
public bool NameEquals (ReadOnlySpan<char> text);
```

Параметры

text `ReadOnlySpan<Char>`

Текст, с которым выполняется сравнение.

Возвращаемое значение

`Boolean`

Значение `true`, если имя этого свойства совпадает с `text`; в противном случае — `false`.

Исключения

InvalidOperationException

Type этого значения не является [PropertyName](#).

Комментарии

Этот метод функционально равен выполнению порядкового `text` сравнения и [Name](#), но он может избежать создания экземпляра строки.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

NameEquals(String)

Сравнивает указанную строку с именем этого свойства.

C#

```
public bool NameEquals (string? text);
```

Параметры

text [String](#)

Текст, с которым выполняется сравнение.

Возвращаемое значение

[Boolean](#)

Значение `true`, если имя этого свойства совпадает с `text`; в противном случае — `false`.

Исключения

[InvalidOperationException](#)

`Type` этого значения не является [PropertyName](#).

Комментарии

Этот метод функционально равен выполнению порядкового сравнения `text` и `Name`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty.ToString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет строковое представление свойства для целей отладки.

C#

```
public override string ToString();
```

Возвращаемое значение

[String](#)

Строка, содержащая неинтерпретированное значение свойства, начинающаяся с объявляющей открывающей кавычки и заканчивающаяся последним символом, который является частью значения.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonProperty.WriteTo(Utf8JsonWriter) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Записывает свойство в предоставленный модуль записи как свойство именованного объекта JSON.

C#

```
public void WriteTo (System.Text.Json.Utf8JsonWriter writer);
```

Параметры

writer [Utf8JsonWriter](#)

Модуль записи, в который необходимо записать свойство.

Исключения

[ArgumentNullException](#)

writer имеет значение `null`.

[ArgumentException](#)

[Name](#) слишком велико, чтобы быть свойством объекта JSON.

[InvalidOperationException](#)

[ValueKind](#) этого свойства JSON [Value](#) приведет к недопустимому JSON.

[ObjectDisposedException](#)

Родительский объект [JsonDocument](#) был удален.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonReaderOptions Struct

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Provides the ability for the user to define custom behavior when reading JSON.

C#

```
public struct JsonReaderOptions
```

Inheritance Object → [ValueType](#) → JsonReaderOptions

Remarks

For more information, see [How to write custom serializers and deserializers with System.Text.Json](#).

Properties

AllowTrailingCommas	Gets or sets a value that defines whether an extra comma at the end of a list of JSON values in an object or array is allowed (and ignored) within the JSON payload being read.
CommentHandling	Gets or sets a value that determines how the Utf8JsonReader handles comments when reading through the JSON data.
MaxDepth	Gets or sets the maximum depth allowed when reading JSON, with the default (that is, 0) indicating a maximum depth of 64.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonReaderOptions.AllowTrailingCommas Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets a value that defines whether an extra comma at the end of a list of JSON values in an object or array is allowed (and ignored) within the JSON payload being read.

C#

```
public bool AllowTrailingCommas { get; set; }
```

Property Value

[Boolean](#)

`true` if an extra comma is allowed; otherwise, `false`.

Remarks

By default, this property is set to `false`, and a [JsonException](#) is thrown if a trailing comma is encountered.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

JsonReaderOptions.CommentHandling

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, определяющее способ, с помощью которого [Utf8JsonReader](#) обрабатывает комментарии при чтении данных JSON.

C#

```
public System.Text.Json.JsonCommentHandling CommentHandling { get; set; }
```

Значение свойства

[JsonCommentHandling](#)

Одно из значений перечисления, указывающее, как обрабатываются комментарии.

Исключения

[ArgumentOutOfRangeException](#)

Свойству присвоено значение, которое не является членом перечисления [JsonCommentHandling](#).

Комментарии

По умолчанию средство чтения создает исключение, [JsonException](#) если встречает комментарий.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonReaderOptions.MaxDepth Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets the maximum depth allowed when reading JSON, with the default (that is, 0) indicating a maximum depth of 64.

C#

```
public int MaxDepth { get; set; }
```

Property Value

[Int32](#)

The maximum depth allowed when reading JSON.

Exceptions

[ArgumentOutOfRangeException](#)

The maximum depth is being set to a negative value.

Remarks

Reading past this depth will throw a [JsonException](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

JsonReaderState Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Определяет непрозрачный тип, содержащий и сохраняющий все соответствующие сведения о состоянии, которые необходимо предоставить [Utf8JsonReader](#) для продолжения чтения после обработки неполных данных.

C#

```
public struct JsonReaderState
```

Наследование [Object](#) → [ValueType](#) → JsonReaderState

Комментарии

`JsonReaderState` требуется для поддержки повторного входа при чтении неполных данных и для продолжения чтения при появлении дополнительных данных.

[Utf8JsonReader](#) В отличие от структуры, которая представляет собой структуру ссылки, этот тип может существовать в пределах асинхронного или `await`, поэтому он должен обеспечить поддержку асинхронного чтения данных перед продолжением работы с новым экземпляром [Utf8JsonReader](#).

Конструкторы

[JsonReaderState\(JsonReader
Options\)](#)

Формирует новый экземпляр [JsonReaderState](#).

Свойства

[Options](#)

Возвращает пользовательское поведение, применяемое при чтении данных JSON с помощью структуры [Utf8JsonReader](#), которое может отклоняться от спецификации JSON, отражающей поведение по умолчанию.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonReaderState(JsonReaderOptions) Конструктор

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Формирует новый экземпляр [JsonReaderState](#).

C#

```
public JsonReaderState (System.Text.Json.JsonReaderOptions options =  
default);
```

Параметры

options [JsonReaderOptions](#)

Определяет пользовательское поведение [Utf8JsonReader](#), которое отличается от RFC по JSON (например, способом обработки комментариев или максимально допустимой глубиной при чтении). По умолчанию [Utf8JsonReader](#) строго следует RFC по JSON (то есть комментарии в JSON являются недопустимыми) и осуществляет чтение до максимальной глубины 64.

Исключения

[ArgumentException](#)

Для максимальной глубины задано значение, не являющееся положительным (< 0).

Комментарии

Экземпляр этого состояния должен быть передан конструктору [Utf8JsonReader](#) с данными JSON. [Utf8JsonReader](#) отличие от структуры , которая является структурой ссылки, состояние может существовать через границы `async/await`, поэтому этот тип требуется для обеспечения поддержки асинхронного чтения дополнительных данных перед продолжением работы с новым экземпляром [Utf8JsonReader](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonReaderState.Options Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the custom behavior to use when reading JSON data using the [Utf8JsonReader](#) struct that may deviate from strict adherence to the JSON specification, which is the default behavior.

C#

```
public System.Text.Json.JsonReaderOptions Options { get; }
```

Property Value

[JsonReaderOptions](#)

The custom behavior to use when reading JSON data.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonSerializer Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

ⓘ Важно!

Некоторые сведения относятся к предварительной версии продукта, в которую до выпуска могут быть внесены существенные изменения. Майкрософт не предоставляет никаких гарантий, явных или подразумеваемых, относительно приведенных здесь сведений.

Предоставляет функциональные возможности сериализации объектов или типов значений в JSON и десериализации JSON в объекты или типы значений.

C#

```
public static class JsonSerializer
```

Наследование [Object](#) → JsonSerializer

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Свойства

IsReflectionEnabledByDefault	Указывает, следует ли задать для ненастроенных JsonSerializerOptions экземпляров использование на основе DefaultJsonTypeInfoResolver отражения .
--	--

Методы

Deserialize(JsonDocument, JsonTypeInfo)	Преобразует объект , JsonDocument представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(JsonDocument, Type, JsonSerializerContext)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonDocument, Type, JsonSerializerOptions)	Преобразует объект , JsonDocument представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonElement, JsonTypeInfo)	Преобразует объект , JsonElement представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(JsonElement, Type, JsonSerializerContext)	Преобразует объект , JsonElement представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonElement, Type, JsonSerializerOptions)	Преобразует объект , JsonElement представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonNode, JsonTypeInfo)	Преобразует объект , JsonNode представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(JsonNode, Type, JsonSerializerContext)	Преобразует объект , JsonNode представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(JsonNode, Type, JsonSerializerOptions)	Преобразует объект , JsonNode представляющий одно значение JSON, <code>returnType</code> в .
Deserialize(ReadOnlySpan<Byte>, JsonTypeInfo)	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerContext)	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в . <code>returnType</code>
Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerOptions)	Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр указанного типа.
Deserialize(ReadOnlySpan<Char>, JsonTypeInfo)	Анализирует текст, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerContext)	Анализирует текст, представляющий одно значение JSON, в . <code>returnType</code>
Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerOptions)	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр указанного типа.
Deserialize(Stream, JsonTypeInfo)	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .

	Поток будет считан до завершения.
Deserialize(Stream, Type, JsonSerializerContext)	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в <code>.returnType</code> Поток будет считан до завершения.
Deserialize(Stream, Type, JsonSerializerOptions)	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в <code>.returnType</code> Поток будет считан до завершения.
Deserialize(String, JsonTypeInfo)	Анализирует текст, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(String, Type, JsonSerializerContext)	Анализирует текст, представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(String, Type, JsonSerializerOptions)	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр указанного типа.
Deserialize(Utf8JsonReader, JsonTypeInfo)	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в экземпляр, указанный в <code>jsonTypeInfo</code> .
Deserialize(Utf8JsonReader, Type, JsonSerializerContext)	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в <code>returnType</code> .
Deserialize(Utf8JsonReader, Type, JsonSerializerOptions)	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения и преобразует его в экземпляр указанного типа.
Deserialize< TValue >(Json Document, JsonSerializer Options)	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Document, JsonTypeInfo< TValue >)	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Element, JsonSerializer Options)	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Element, JsonTypeInfo< TValue >)	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Node, JsonSerializerOptions)	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, <code>TValue</code> в .
Deserialize< TValue >(Json Node, JsonTypeInfo< TValue >)	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, <code>TValue</code> в .

<code>Deserialize< TValue >(Read OnlySpan< Byte >, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(Read OnlySpan< Byte >, JsonType Info< TValue >)</code>	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в . <code>TValue</code>
<code>Deserialize< TValue >(Read OnlySpan< Char >, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(Read OnlySpan< Char >, JsonType Info< TValue >)</code>	Анализирует текст, представляющий одно значение JSON, в . <code>TValue</code>
<code>Deserialize< TValue >(Stream, JsonSerializerOptions)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в . <code>TValue</code> Поток будет считан до завершения.
<code>Deserialize< TValue >(Stream, JsonTypeInfo< TValue >)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в . <code>TValue</code> Поток будет считан до завершения.
<code>Deserialize< TValue >(String, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(String, JsonTypeInfo< TValue >)</code>	Анализирует текст, представляющий одно значение JSON, в . <code>TValue</code>
<code>Deserialize< TValue >(Utf8Json Reader, JsonSerializerOptions)</code>	Считывает одно значение JSON (включая объекты или массивы) из предоставленного модуля чтения в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(Utf8Json Reader, JsonTypeInfo< TValue >)</code>	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в <code>TValue</code> .
<code>DeserializeAsync(Stream, Json TypeInfo, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> . Поток будет считан до завершения.
<code>DeserializeAsync(Stream, Type, JsonSerializerContext, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в . <code>returnType</code> Поток будет считан до завершения.
<code>DeserializeAsync(Stream, Type, JsonSerializerOptions, CancellationToken)</code>	Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр указанного типа. Поток считывается до завершения.
<code>DeserializeAsync< TValue >(Stream, JsonSerializerOptions, CancellationToken)</code>	Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа,

<code>CancellationToken)</code>	заданного параметром универсального типа. Поток считывается до завершения.
<code>DeserializeAsync< TValue >(Stream, JsonType Info< TValue >, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в <code>.TValue</code> . Поток будет считан до завершения.
<code>DeserializeAsyncEnumerable< TValue >(Stream, JsonSerializerOptions, CancellationToken)</code>	Заключает текст в кодировке UTF-8 в <code>IAsyncEnumerable< T ></code> , который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.
<code>DeserializeAsyncEnumerable< TValue >(Stream, JsonTypeInfo< TValue >, CancellationToken)</code>	Заключает текст в кодировке UTF-8 в <code>IAsyncEnumerable< T ></code> , который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.
<code>Serialize(Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в <code>String</code> .
<code>Serialize(Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в <code>String</code> .
<code>Serialize(Object, Type, JsonSerializerOptions)</code>	Преобразует значение указанного типа в строку JSON.
<code>Serialize(Stream, Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .
<code>Serialize(Stream, Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .
<code>Serialize(Stream, Object, Type, JsonSerializerOptions)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .
<code>Serialize(Utf8JsonWriter, Object, JsonTypeInfo)</code>	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.
<code>Serialize(Utf8JsonWriter, Object, Type, JsonSerializerContext)</code>	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.
<code>Serialize(Utf8JsonWriter, Object, Type, JsonSerializerOptions)</code>	Записывает представление JSON указанного типа в предоставленный модуль записи.
<code>Serialize< TValue >(Stream, TValue, JsonSerializerOptions)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .
<code>Serialize< TValue >(Stream, TValue, JsonType Info< TValue >)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .

<code>Serialize< TValue >(TValue, JsonSerializerOptions)</code>	Преобразует значение типа, заданного параметром универсального типа, в строку JSON.
<code>Serialize< TValue >(TValue, JsonTypeInfo< TValue >)</code>	Преобразует предоставленное значение в String .
<code>Serialize< TValue >(Utf8Json Writer, TValue, JsonSerializer Options)</code>	Записывает представление JSON типа, указанного параметром универсального типа, в предоставленный модуль записи.
<code>Serialize< TValue >(Utf8Json Writer, TValue, JsonType Info< TValue >)</code>	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.
<code>SerializeAsync(Stream, Object, JsonTypeInfo, CancellationToken)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
<code>SerializeAsync(Stream, Object, Type, JsonSerializerContext, CancellationToken)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
<code>SerializeAsync(Stream, Object, Type, JsonSerializerOptions, CancellationToken)</code>	Асинхронно преобразует значение указанного типа в текст JSON в кодировке UTF-8 и записывает его в указанный поток.
<code>SerializeAsync< TValue > (Stream, TValue, JsonSerializer Options, CancellationToken)</code>	Асинхронно преобразует значение типа, заданного параметром универсального типа, в текст JSON в кодировке UTF-8 и записывает его в поток.
<code>SerializeAsync< TValue > (Stream, TValue, JsonType Info< TValue >, CancellationToken)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
<code>SerializeToDocument(Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToDocument(Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToDocument(Object, Type, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonDocument .
<code>Serialize ToDocument< TValue >(TValue, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonDocument .
<code>Serialize ToDocument< TValue >(TValue, JsonTypeInfo< TValue >)</code>	Преобразует предоставленное значение в JsonDocument .

<code>SerializeToElement(Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в JsonElement .
<code>SerializeToElement(Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToElement(Object, Type, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToElement< TValue > (TValue, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToElement< TValue > (TValue, JsonType Info< TValue >)</code>	Преобразует предоставленное значение в JsonDocument .
<code>SerializeToNode(Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode(Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode(Object, Type, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode< TValue > (TValue, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode< TValue > (TValue, JsonType Info< TValue >)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToUtf8Bytes(Object, JsonTypeInfo)</code>	Преобразует указанное значение в Byte массив.
<code>SerializeToUtf8Bytes(Object, Type, JsonSerializerContext)</code>	Преобразует указанное значение в Byte массив.
<code>SerializeToUtf8Bytes(Object, Type, JsonSerializerOptions)</code>	Преобразует значение указанного типа в строку JSON, закодированную как байты UTF-8.
<code>SerializeToUtf8Bytes< TValue > (TValue, JsonSerializerOptions)</code>	Преобразует значение типа, указанного параметром универсального типа, в строку JSON, закодированную как байты UTF-8.
<code>SerializeToUtf8Bytes< TValue > (TValue, JsonType Info< TValue >)</code>	Преобразует указанное значение в Byte массив.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonSerializer.IsReflectionEnabledByDefault Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

ⓘ Важно!

Некоторые сведения относятся к предварительной версии продукта, в которую до выпуска могут быть внесены существенные изменения. Майкрософт не предоставляет никаких гарантий, явных или подразумеваемых, относительно приведенных здесь сведений.

Указывает, следует ли задать для ненастроенных [JsonSerializerOptions](#) экземпляров использование на основе [DefaultJsonTypeInfoResolver](#) отражения .

C#

```
public static bool IsReflectionEnabledByDefault { get; }
```

Значение свойства

Boolean

Комментарии

Значение свойства поддерживается параметром [System.Text.Json.JsonSerializer.IsReflectionEnabledByDefault](#) и [AppContext](#) по умолчанию имеет значение , `true` если не задано.

Применяется к

Продукт	Версии
.NET	8

JsonSerializer.Deserialize Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

Deserialize(Stream, JsonTypeInfo)	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> . Поток будет считываться до завершения.
Deserialize(Utf8JsonReader, Type, JsonSerializerContext)	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в <code>returnType</code> .
Deserialize(Utf8JsonReader, Type, JsonSerializerOptions)	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения и преобразует его в экземпляр указанного типа.
Deserialize(JsonNode, Type, JsonSerializerOptions)	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(JsonElement, Type, JsonSerializerContext)	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(JsonElement, Type, JsonSerializerOptions)	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(JsonDocument, Type, JsonSerializerContext)	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(JsonDocument, Type, JsonSerializerOptions)	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(String, Type, JsonSerializerContext)	Анализирует текст, представляющий одно значение JSON, в <code>.returnType</code>
Deserialize(String, Type, JsonSerializerOptions)	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр указанного типа.
Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerContext)	Анализирует текст, представляющий одно значение JSON, в <code>.returnType</code>

<code>Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляре указанного типа.
<code>Deserialize(JsonNode, Type, JsonSerializerContext)</code>	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, в <code>.returnType</code>
<code>Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляре указанного типа.
<code>Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerContext)</code>	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в <code>.returnType</code>
<code>Deserialize(ReadOnlySpan<Byte>, JsonTypeInfo)</code>	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляре, указанный в <code>jsonTypeInfo</code> .
<code>Deserialize(String, JsonTypeInfo)</code>	Анализирует текст, представляющий одно значение JSON, в экземпляре, указанный в <code>jsonTypeInfo</code> .
<code>Deserialize(JsonDocument, JsonTypeInfo)</code>	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, в экземпляре, заданный <code>.jsonTypeInfo</code>
<code>Deserialize(ReadOnlySpan<Char>, JsonTypeInfo)</code>	Анализирует текст, представляющий одно значение JSON, в экземпляре, указанный в <code>jsonTypeInfo</code> .
<code>Deserialize(JsonNode, JsonTypeInfo)</code>	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, в экземпляре, заданный <code>.jsonTypeInfo</code>
<code>Deserialize(Utf8JsonReader, JsonTypeInfo)</code>	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в экземпляре, заданный <code>jsonTypeInfo</code> .
<code>Deserialize(Stream, Type, JsonSerializerOptions)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в <code>.returnType</code> Поток будет считываться до завершения.
<code>Deserialize(Stream, Type, JsonSerializerContext)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в <code>.returnType</code> Поток будет считываться до завершения.
<code>Deserialize(JsonElement, JsonTypeInfo)</code>	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, в экземпляре, заданный <code>.jsonTypeInfo</code>
<code>Deserialize< TValue >(Utf8JsonReader, JsonTypeInfo< TValue >)</code>	Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в <code>TValue</code> .
<code>Deserialize< TValue >(JsonDocument, JsonTypeInfo< TValue >)</code>	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, в <code>TValue</code>

<code>JsonTypeInfo< TValue ></code>	
<code>Deserialize< TValue >(Utf8JsonReader, JsonSerializerOptions)</code>	Считывает одно значение JSON (включая объекты или массивы) из предоставленного модуля чтения в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(JsonNode, JsonTypeInfo< TValue >)</code>	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(JsonNode, JsonSerializerOptions)</code>	Преобразует объект <code>JsonNode</code> представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(JsonElement, JsonTypeInfo< TValue >)</code>	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(JsonElement, JsonSerializerOptions)</code>	Преобразует объект <code>JsonElement</code> представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(JsonDocument, JsonSerializerOptions)</code>	Преобразует объект <code>JsonDocument</code> представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(ReadOnlySpan< Char >, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(String, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(ReadOnlySpan< Char >, JsonTypeInfo< TValue >)</code>	Анализирует текст, представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(ReadOnlySpan< Byte >, JsonTypeInfo< TValue >)</code>	Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в <code>. TValue</code>
<code>Deserialize< TValue >(ReadOnlySpan< Byte >, JsonSerializerOptions)</code>	Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.
<code>Deserialize< TValue >(Stream, JsonTypeInfo< TValue >)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в <code>. TValue</code> Поток будет считываться до завершения.
<code>Deserialize< TValue >(Stream, JsonSerializerOptions)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в <code>. TValue</code> Поток будет считываться до завершения.

`Deserialize< TValue >(String, JsonTypeInfo< TValue >)`

Анализирует текст, представляющий одно значение JSON, в `.TValue`

Deserialize(Stream, JsonTypeInfo)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`. Поток будет считываться до завершения.

C#

```
public static object? Deserialize (System.IO.Stream utf8Json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

`utf8Json` `Stream`

Данные JSON для анализа.

`jsonTypeInfo` `JsonTypeInfo`

Метаданные о преобразуемом типе.

Возвращаемое значение

`Object`

Представление `jsonTypeInfo` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `utf8Json` или `jsonTypeInfo` имеет значение `null`.

[JsonException](#)

Недопустимый код JSON или в потоке есть оставшиеся данные.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(Utf8JsonReader, Type, JsonSerializerContext)

Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в `returnType`.

C#

```
public static object? Deserialize (ref System.Text.Json.Utf8JsonReader  
reader, Type returnType,  
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

reader [Utf8JsonReader](#)

Модуль чтения, используемый для чтения.

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `returnType` или `context` имеет значение `null`.

[JsonException](#)

Недопустимый формат JSON, `returnType` несовместим с JSON, или не удалось прочитать значение из модуля чтения.

ArgumentException

`reader` использует неподдерживаемые параметры.

NotSupportedException

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

InvalidOperationException

Метод `GetTypeInfo(Type)` в предоставленном `context` объекте не вернул совместимый `JsonTypeInfo` для `returnType`.

Комментарии

`TokenType` Если свойство имеет `reader` значение `PropertyName` или `None`, средство чтения будет расширено одним вызовом `Read()` чтобы определить начало значения.

После завершения этого метода `reader` позиционируется в окончательном маркере в значении JSON. При возникновении исключения средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, с помощью функции чтения, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Объект `JsonReaderOptions` используемый для создания экземпляра `Utf8JsonReader`, имеет приоритет над `JsonSerializerOptions` когда они конфликтуют. Следовательно, `AllowTrailingCommas` при чтении используются, `MaxDepth` и `CommentHandling`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(Utf8JsonReader, Type, JsonSerializerOptions)

Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения и преобразует его в экземпляр указанного типа.

C#

```
public static object? Deserialize (ref System.Text.Json.Utf8JsonReader  
reader, Type returnType, System.Text.Json.JsonSerializerOptions? options  
= default);
```

Параметры

reader [Utf8JsonReader](#)

Модуль чтения, из которого читается JSON.

returnType [Type](#)

Тип объекта для преобразования и возврата.

options [JsonSerializerOptions](#)

Параметры для управления поведением сериализатора во время чтения.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Не удалось считать значение из модуля чтения.

ArgumentException

`reader` использует неподдерживаемые параметры.

NotSupportedException

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Комментарии

`TokenType` Если свойство имеет `reader` значение `JsonTokenType.PropertyName` или `JsonTokenType.None`, средство чтения будет расширено одним вызовом `Utf8JsonReader.Read()` чтобы определить начало значения.

После завершения этого метода `reader` будет располагаться в конечном токене в значении JSON. При возникновении исключения средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, с помощью функции чтения, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Объект `JsonReaderOptions` используемый для создания экземпляра `Utf8JsonReader`, имеет приоритет над `JsonSerializerOptions` когда они конфликтуют. Следовательно, `JsonReaderOptions.AllowTrailingCommas` при чтении используются `JsonReaderOptions.MaxDepth` и `JsonReaderOptions.CommentHandling`.

Дополнительные сведения см. в статье [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize(JsonNode, Type, JsonSerializerOptions)

Преобразует объект `JsonNode` представляющий одно значение JSON, в `returnType`

C#

```
public static object? Deserialize (this System.Text.Json.Nodes.JsonNode?  
node, Type returnType, System.Text.Json.JsonSerializerOptions? options =  
default);
```

Параметры

node `JsonNode`

Преобразуемый объект `JsonNode`.

returnType `Type`

Тип объекта для преобразования и возврата.

options `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

`Object`

Представление `returnType` данного значения JSON.

Исключения

`JsonException`

`returnType` несовместим с JSON.

`NotSupportedException`

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonElement, Type, JsonSerializerContext)

Преобразует объект `JsonElement` представляющий одно значение JSON, в `returnType`

C#

```
public static object? Deserialize (this System.Text.Json.JsonElement
element, Type returnType,
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

element `JsonElement`

Преобразуемый объект `JsonElement`.

returnType `Type`

Тип объекта для преобразования и возврата.

context `JsonSerializerContext`

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

`Object`

Представление `returnType` данного значения JSON.

Исключения

`ArgumentNullException`

`returnType` имеет значение `null`.

-ИЛИ-

`context` имеет значение `null`.

JsonException

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

InvalidOperationException

Метод `GetTypeInfo(Type)` предоставленного `context` возвращает `null` преобразуемый тип.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonElement, Type, JsonSerializerOptions)

Преобразует объект `JsonElement` представляющий одно значение JSON, в `returnType`

C#

```
public static object? Deserialize (this System.Text.Json.JsonElement
element, Type returnType, System.Text.Json.JsonSerializerOptions? options
= default);
```

Параметры

`element` `JsonElement`

Преобразуемый объект [JsonElement](#).

returnType [Type](#)

Тип объекта для преобразования и возврата.

options [JsonSerializerOptions](#)

Параметры для управления поведением во время анализа.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

[JsonException](#)

`returnType` несовместим с JSON.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonDocument, Type, JsonSerializerContext)

Преобразует объект `JsonDocument` представляющий одно значение JSON, в `returnType`.

C#

```
public static object? Deserialize (this System.Text.Json.JsonDocument
document, Type returnType,
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

document [JsonDocument](#)

Преобразуемый объект [JsonDocument](#).

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`document` имеет значение `null`.

-ИЛИ-

`returnType` имеет значение `null`.

-ИЛИ-

`context` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

NotSupportedException

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

InvalidOperationException

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonDocument, Type, JsonSerializerOptions)

Преобразует объект `, JsonDocument` представляющий одно значение JSON, в `.returnType`

C#

```
public static object? Deserialize (this System.Text.Json.JsonDocument
document, Type returnType, System.Text.Json.JsonSerializerOptions?
options = default);
```

Параметры

document [JsonDocument](#)

Преобразуемый объект [JsonDocument](#).

returnType [Type](#)

Тип объекта для преобразования и возврата.

options [JsonSerializerOptions](#)

Параметры для управления поведением во время анализа.

Возвращаемое значение

Object

Представление `returnType` данного значения JSON.

Исключения

ArgumentNullException

Параметр `document` или `returnType` имеет значение `null`.

JsonException

`returnType` несовместим с JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(String, Type, JsonSerializerContext)

Анализирует текст, представляющий одно значение JSON, в `.returnType`

C#

```
public static object? Deserialize (string json, Type returnType,  
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

json String

Текст JSON для анализа.

returnType Type

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `json` или `returnType` имеет значение `null`.

-ИЛИ-

`context` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

Комментарии

[String](#) Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(String, Type, JsonSerializerOptions)

Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр указанного типа.

C#

```
public static object? Deserialize (string json, Type returnType,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

json `String`

Анализируемый текст JSON.

returnType `Type`

Тип объекта для преобразования и возврата.

options `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

`Object`

Представление `returnType` данного значения JSON.

Исключения

`ArgumentNullException`

Параметр `json` или `returnType` имеет значение `null`.

`JsonException`

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Комментарии

`String` Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Дополнительные сведения см. в статье [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerContext)

Анализирует текст, представляющий одно значение JSON, в `.returnType`

C#

```
public static object? Deserialize (ReadOnlySpan<char> json, Type
returnType, System.Text.Json.Serialization.JsonSerializerContext
context);
```

Параметры

json `ReadOnlySpan<Char>`

Текст JSON для анализа.

returnType `Type`

Тип объекта для преобразования и возврата.

context `JsonSerializerContext`

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

`Object`

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `json` или `returnType` имеет значение `null`.

-ИЛИ-

`context` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

Комментарии

[String](#) Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(ReadOnlySpan<Char>, Type, JsonSerializerOptions)

Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляре указанного типа.

C#

```
public static object? Deserialize (ReadOnlySpan<char> json, Type  
returnType, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

json [ReadOnlySpan<Char>](#)

Анализируемый текст JSON.

returnType [Type](#)

Тип объекта для преобразования и возврата.

options [JsonSerializerOptions](#)

Параметры для управления поведением во время анализа.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в диапазоне больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Комментарии

Использование диапазона UTF-16 не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonNode, Type, JsonSerializerContext)

Преобразует объект `JsonNode` представляющий одно значение JSON, в `.returnType`

C#

```
public static object? Deserialize (this System.Text.Json.Nodes.JsonNode?  
node, Type returnType,  
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

node [JsonNode](#)

Преобразуемый объект [JsonNode](#).

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

-ИЛИ-

`context` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод `GetTypeInfo(Type)` предоставленного `context` возвращает `null` преобразуемый тип.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerOptions)

Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр указанного типа.

C#

```
public static object? Deserialize (ReadOnlySpan<byte> utf8Json, Type  
returnType, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

`utf8Json` `ReadOnlySpan<Byte>`

Анализируемый текст JSON.

`returnType` `Type`

Тип объекта для преобразования и возврата.

`options` `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

`Object`

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`returnType` несовместим с JSON.

-ИЛИ-

Остались данные в диапазоне больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `returnType` или его сериализуемых членов отсутствуют.

Комментарии

Дополнительные сведения см. в статье [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize(ReadOnlySpan<Byte>, Type, JsonSerializerContext)

Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в `.returnType`

C#

```
public static object? Deserialize (ReadOnlySpan<byte> utf8Json, Type  
returnType, System.Text.Json.Serialization.JsonSerializerContext  
context);
```

Параметры

utf8Json [ReadOnlySpan<Byte>](#)

Текст JSON для анализа.

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

`returnType` имеет значение `null`.

[JsonException](#)

`Json` является недопустимым, `returnType` несовместим с JSON или есть оставшиеся данные в Stream.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) в предоставленном `context` объекте не вернул совместимый [JsonTypeInfo](#) для `returnType`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(ReadOnlySpan<Byte>, JsonTypeInfo)

Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (ReadOnlySpan<byte> utf8Json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

`utf8Json` `ReadOnlySpan<Byte>`

Текст JSON для анализа.

`jsonTypeInfo` `JsonTypeInfo`

Метаданные о преобразуемом типе.

Возвращаемое значение

`Object`

Представление `jsonTypeInfo` данного значения JSON.

Исключения

`JsonException`

Json является недопустимым, или в буфере есть оставшиеся данные.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(String, JsonTypeInfo)

Анализирует текст, представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (string json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

json `String`

Текст JSON для анализа.

jsonTypeInfo `JsonTypeInfo`

Метаданные о преобразуемом типе.

Возвращаемое значение

`Object`

Представление `jsonTypeInfo` данного значения JSON.

Исключения

`ArgumentNullException`

`json` имеет значение `null`.

-ИЛИ-

`jsonTypeInfo` имеет значение `null`.

`JsonException`

Недопустимый JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

Комментарии

[String](#) Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(JsonDocument, JsonTypeInfo)

Преобразует объект , [JsonDocument](#) представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (this System.Text.Json.JsonDocument
document, System.Text.Json.Serialization.Metadata.JsonTypeInfo
jsonTypeInfo);
```

Параметры

document [JsonDocument](#)

Преобразуемый объект [JsonDocument](#).

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[Object](#)

Представление `jsonTypeInfo` данного значения JSON.

Исключения

ArgumentNullException

`document` имеет значение `null`.

-или-

`jsonTypeInfo` имеет значение `null`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(ReadOnlySpan<Char>, JsonTypeInfo)

Анализирует текст, представляющий одно значение JSON, в экземпляре, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (ReadOnlySpan<char> json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

`json` `ReadOnlySpan<Char>`

Текст JSON для анализа.

`jsonTypeInfo` `JsonTypeInfo`

Метаданные о преобразуемом типе.

Возвращаемое значение

`Object`

Представление `jsonTypeInfo` данного значения JSON.

Исключения

ArgumentNullException

`jsonTypeInfo` имеет значение `null`.

JsonException

Недопустимый JSON.

-или-

Остались данные в строке больше одного значения JSON.

Комментарии

String Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(JsonNode, JsonTypeInfo)

Преобразует объект `JsonNode` представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (this System.Text.Json.Nodes.JsonNode?  
node, System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

`node` `JsonNode`

Преобразуемый объект `JsonNode`.

`jsonTypeInfo` `JsonTypeInfo`

Метаданные о преобразуемом типе.

Возвращаемое значение

Object

Представление `jsonTypeInfo` данного значения JSON.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(Utf8JsonReader, JsonTypeInfo)

Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (ref System.Text.Json.Utf8JsonReader  
reader, System.Text.Json.Serialization.Metadata.JsonTypeInfo  
jsonTypeInfo);
```

Параметры

`reader` [Utf8JsonReader](#)

Модуль чтения, используемый для чтения.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

Object

Представление `jsonTypeInfo` данного значения JSON.

Исключения

[JsonException](#)

Недопустимый формат JSON, `jsonTypeInfo` несовместим с JSON, или не удалось прочитать значение из модуля чтения.

[ArgumentException](#)

`reader` использует неподдерживаемые параметры.

Комментарии

TokenType Если свойство имеет значение `PropertyName` или `None`, средство чтения будет расширено одним вызовом для `Read()` определения начала значения.

После завершения этого метода `reader` будет позиционироваться в последнем токене в значении JSON. Если возникает исключение, средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, на которые действовал читатель, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Объект `JsonReaderOptions` используемый для создания экземпляра `Utf8JsonReader` имеет приоритет над `JsonSerializerOptions`, когда они конфликтуют. Таким образом, `AllowTrailingCommas` при чтении используются, `MaxDepth` и `CommentHandling`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize(Stream, Type, JsonSerializerOptions)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON в `.returnType`. Поток будет считан до завершения.

C#

```
public static object? Deserialize (System.IO.Stream utf8Json, Type  
returnType, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

`utf8Json` [Stream](#)

Данные JSON для анализа.

`returnType` [Type](#)

Тип объекта для преобразования и возврата.

`options` [JsonSerializerOptions](#)

Параметры для управления поведением во время чтения.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `utf8Json` или `returnType` имеет значение `null`.

[JsonException](#)

Json недопустим `returnType`, несовместим с JSON или в потоке остались данные.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(Stream, Type, JsonSerializerContext)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON в `returnType`. Поток будет считан до завершения.

C#

```
public static object? Deserialize (System.IO.Stream utf8Json, Type  
returnType, System.Text.Json.Serialization.JsonSerializerContext  
context);
```

Параметры

utf8Json [Stream](#)

Данные JSON для анализа.

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Object](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Значение параметра `utf8Json`, `returnType` или `context` равно `null`.

[JsonException](#)

Json недопустим `returnType`, несовместим с JSON или в потоке остались данные.

NotSupportedException

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

InvalidOperationException

Метод [GetTypeInfo\(Type\)](#) в предоставленном `context` не вернул совместимый [JsonTypeInfo](#) для `returnType`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize(JsonElement, JsonTypeInfo)

Преобразует объект [JsonElement](#) представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`.

C#

```
public static object? Deserialize (this System.Text.Json.JsonElement  
element, System.Text.Json.Serialization.Metadata.JsonTypeInfo  
jsonTypeInfo);
```

Параметры

element [JsonElement](#)

Преобразуемый объект [JsonElement](#).

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[Object](#)

Представление `jsonTypeInfo` данного значения JSON.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Deserialize< TValue >(Utf8JsonReader, JsonTypeInfo< TValue >)

Считывает одно значение JSON (включая объекты или массивы) из предоставленного средства чтения в `TValue`.

C#

```
public static TValue? Deserialize< TValue > (ref
System.Text.Json.Utf8JsonReader reader,
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
jsonTypeInfo);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`reader` [Utf8JsonReader](#)

Модуль чтения, используемый для чтения.

`jsonTypeInfo` [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

[JsonException](#)

Недопустимый формат JSON, `TValue` несовместим с JSON, или не удалось прочитать значение из модуля чтения.

[ArgumentException](#)

`reader` использует неподдерживаемые параметры.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

`TokenType` Если свойство имеет `reader` значение `PropertyName` или `None`, средство чтения будет расширено одним вызовом для `Read()` определения начала значения.

После завершения этого метода `reader` позиционируется в последнем токене в значении JSON. Если возникает исключение, средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, на которые действовал читатель, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Объект `JsonReaderOptions` используемый для создания экземпляра `Utf8JsonReader` имеет приоритет над `JsonSerializerOptions`, когда они конфликтуют. Таким образом, `AllowTrailingCommas` при чтении используются, `MaxDepth` и `CommentHandling`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize< TValue >(JsonDocument, JsonTypeInfo< TValue >)

Преобразует объект [JsonDocument](#) представляющий одно значение JSON, [TValue](#) в .

C#

```
public static TValue? Deserialize< TValue > (this  
System.Text.Json.JsonDocument document,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >  
jsonTypeInfo);
```

Параметры типа

[TValue](#)

Тип для десериализации значения JSON.

Параметры

[document](#) [JsonDocument](#)

Преобразуемый объект [JsonDocument](#).

[jsonTypeInfo](#) [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[TValue](#)

Представление [TValue](#) данного значения JSON.

Исключения

[ArgumentNullException](#)

[document](#) имеет значение [null](#).

-или-

[jsonTypeInfo](#) имеет значение [null](#).

[JsonException](#)

`TValue` несовместим с JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(Utf8JsonReader, JsonSerializerOptions)

Считывает одно значение JSON (включая объекты или массивы) из предоставленного модуля чтения в экземпляр типа, заданного параметром универсального типа.

C#

```
public static TValue? Deserialize<TValue> (ref  
System.Text.Json.Utf8JsonReader reader,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Целевой тип значения JSON.

Параметры

reader Utf8JsonReader

Модуль чтения, из которого читается JSON.

options JsonSerializerOptions

Параметры для управления поведением сериализатора во время чтения.

Возвращаемое значение

TValue

Представление TValue данного значения JSON.

Исключения

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

TValue несовместим с JSON.

-ИЛИ-

Не удалось считать значение из модуля чтения.

[ArgumentException](#)

reader использует неподдерживаемые параметры.

[NotSupportedException](#)

Совместимые объекты JsonConverter для TValue или его сериализуемых членов отсутствуют.

Комментарии

TokenType Если свойство имеет reader значение [JsonTokenType.PropertyName](#) или [JsonTokenType.None](#), средство чтения будет расширено одним вызовом для [Utf8JsonReader.Read\(\)](#) определения начала значения.

После завершения этого метода reader будет позиционироваться в последнем токене в значении JSON. Если возникает исключение, средство чтения сбрасывается в состояние, в которое оно находилось при вызове метода.

Этот метод создает копию данных, на которые действовал читатель, поэтому не требуется, чтобы вызывающий объект поддерживал целостность данных после возврата этого метода.

Объект [JsonReaderOptions](#) используемый для создания экземпляра [Utf8JsonReader](#) имеет приоритет над [JsonSerializerOptions](#), когда они конфликтуют. Таким образом, [JsonReaderOptions.AllowTrailingCommas](#) при

чтении используются `JsonReaderOptions.MaxDepth` и `JsonReaderOptions.CommentHandling`.

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize< TValue >(JsonNode, JsonTypeInfo< TValue >)

Преобразует объект `JsonNode` представляющий одно значение JSON, `TValue`.

C#

```
public static TValue? Deserialize<TValue> (this
System.Text.Json.Nodes.JsonNode? node,
System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue>
jsonTypeInfo);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`node` `JsonNode`

Преобразуемый объект `JsonNode`.

`jsonTypeInfo` `JsonTypeInfo<TValue>`

Метаданные о преобразуемом типе.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

[JsonException](#)

`TValue` несовместим с JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(JsonNode, JsonSerializerOptions)

Преобразует объект `JsonNode` представляющий одно значение JSON, `TValue`.

C#

```
public static TValue? Deserialize<TValue> (this  
System.Text.Json.Nodes.JsonNode? node,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

node [JsonNode](#)

Преобразуемый объект [JsonNode](#).

options [JsonSerializerOptions](#)

Параметры для управления поведением во время анализа.

Возвращаемое значение

TValue

Представление **TValue** данного значения JSON.

Исключения

[JsonException](#)

TValue несовместим с JSON.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(JsonElement, JsonTypeInfo<TValue>)

Преобразует объект [JsonElement](#) представляющий одно значение JSON, **TValue** в .

C#

```
public static TValue? Deserialize<TValue> (this
System.Text.Json.JsonElement element,
System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue>
jsonTypeInfo);
```

Параметры типа

TValue

Тип для десериализации значения JSON.

Параметры

element [JsonElement](#)

Преобразуемый объект [JsonElement](#).

jsonTypeInfo [JsonTypeInfo<TValue>](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

TValue

Представление [TValue](#) данного значения JSON.

Исключения

[ArgumentNullException](#)

[jsonTypeInfo](#) имеет значение `null`.

[JsonException](#)

[TValue](#) несовместим с JSON.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для [TValue](#) или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(JsonElement, JsonSerializerOptions)

Преобразует объект `JsonElement` представляющий одно значение JSON,

`TValue`.

C#

```
public static TValue? Deserialize<TValue> (this  
System.Text.Json.JsonElement element,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`element` `JsonElement`

Преобразуемый объект `JsonElement`.

`options` `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

`JsonException`

`TValue` несовместим с JSON.

`NotSupportedException`

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(JsonDocument, JsonSerializerOptions)

Преобразует объект `JsonDocument` представляющий одно значение JSON, `TValue` в .

C#

```
public static TValue? Deserialize<TValue> (this  
System.Text.Json.JsonDocument document,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`document` `JsonDocument`

Преобразуемый объект `JsonDocument`.

`options` `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

`ArgumentNullException`

`document` имеет значение `null`.

`JsonException`

`TValue` несовместим с JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(ReadOnlySpan<Char>, JsonSerializerOptions)

Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.

C#

```
public static TValue? Deserialize<TValue> (ReadOnlySpan<char> json,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип для десериализации значения JSON.

Параметры

json `ReadOnlySpan<Char>`

Анализируемый текст JSON.

options `JsonSerializerOptions`

Параметры для управления поведением во время анализа.

Возвращаемое значение

TValue

Представление `TValue` данного значения JSON.

Исключения

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в диапазоне больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

Использование диапазона UTF-16 не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(String, JsonSerializerOptions)

Выполняет синтаксический анализ текста, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.

C#

```
public static TValue? Deserialize<TValue> (string json,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Целевой тип значения JSON.

Параметры

json String

Анализируемый текст JSON.

options JsonSerializerOptions

Параметры для управления поведением во время анализа.

Возвращаемое значение

TValue

Представление `TValue` данного значения JSON.

Исключения

ArgumentNullException

`json` имеет значение `null`.

JsonException

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

[String](#) Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize< TValue >(ReadOnlySpan< Char >, JsonTypeInfo< TValue >)

Анализирует текст, представляющий одно значение JSON, в `TValue`

C#

```
public static TValue? Deserialize< TValue > (ReadOnlySpan< char > json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >  
jsonTypeInfo);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`json` `ReadOnlySpan<Char>`

Текст JSON для анализа.

`jsonTypeInfo` `JsonTypeInfo< TValue >`

Метаданные о преобразуемом типе.

Возвращаемое значение

TValue

Представление `TValue` данного значения JSON.

Исключения

[ArgumentNullException](#)

`json` имеет значение `null`.

-ИЛИ-

`jsonTypeInfo` имеет значение `null`.

[JsonException](#)

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

`String` Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize< TValue >(ReadOnlySpan< Byte >, JsonTypeInfo< TValue >)

Анализирует текст в кодировке UTF-8, представляющий одно значение JSON, в [TValue](#)

C#

```
public static TValue? Deserialize< TValue > (ReadOnlySpan< byte > utf8Json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >  
jsonTypeInfo);
```

Параметры типа

[TValue](#)

Тип для десериализации значения JSON.

Параметры

[utf8Json](#) [ReadOnlySpan< Byte >](#)

Текст JSON для анализа.

[jsonTypeInfo](#) [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[TValue](#)

Представление [TValue](#) данного значения JSON.

Исключения

[JsonException](#)

Json является недопустимым, [TValue](#) несовместим с JSON или в потоке есть оставшиеся данные.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для [TValue](#) или его сериализуемых членов отсутствуют.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize<TValue>(ReadOnlySpan<Byte>, JsonSerializerOptions)

Выполняет синтаксический анализ текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа.

C#

```
public static TValue? Deserialize<TValue> (ReadOnlySpan<byte> utf8Json,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Целевой тип текста в кодировке UTF-8.

Параметры

utf8Json [ReadOnlySpan<Byte>](#)

Анализируемый текст JSON.

options [JsonSerializerOptions](#)

Параметры для управления поведением во время анализа.

Возвращаемое значение

TValue

Представление TValue данного значения JSON.

Исключения

JsonException

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в диапазоне больше одного значения JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Deserialize<TValue>(Stream, JsonTypeInfo<TValue>)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в `.TValue`. Поток будет считываться до завершения.

C#

```
public static TValue? Deserialize<TValue> (System.IO.Stream utf8Json,
System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue>
jsonTypeInfo);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`utf8Json Stream`

Данные JSON для анализа.

`jsonTypeInfo JsonTypeInfo< TValue >`

Метаданные о преобразуемом типе.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `utf8Json` или `jsonTypeInfo` имеет значение `null`.

[JsonException](#)

Json является недопустимым, `TValue` несовместим с JSON или есть оставшиеся данные в Stream.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize< TValue >(Stream, JsonSerializerOptions)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в `.TValue`. Поток будет считываться до завершения.

C#

```
public static TValue? Deserialize<TValue> (System.IO.Stream utf8Json,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`utf8Json` `Stream`

Данные JSON для анализа.

`options` `JsonSerializerOptions`

Параметры для управления поведением во время чтения.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

[JsonException](#)

`Json` является недопустимым, `TValue` несовместим с JSON или есть оставшиеся данные в `Stream`.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Deserialize< TValue >(String, JsonTypeInfo< TValue >)

Анализирует текст, представляющий одно значение JSON, в `TValue`

C#

```
public static TValue? Deserialize<TValue> (string json,
System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue>
jsonTypeInfo);
```

Параметры типа

`TValue`

Тип для десериализации значения JSON.

Параметры

`json` `String`

Текст JSON для анализа.

`jsonTypeInfo` `JsonTypeInfo<TValue>`

Метаданные о преобразуемом типе.

Возвращаемое значение

`TValue`

Представление `TValue` данного значения JSON.

Исключения

[ArgumentNullException](#)

`json` имеет значение `null`.

-или-

`jsonTypeInfo` имеет значение `null`.

JsonException

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

Остались данные в строке больше одного значения JSON.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

`String` Использование не так эффективно, как использование методов UTF-8, так как реализация изначально использует UTF-8.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.DeserializeAsync Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

<code>DeserializeAsync(Stream, Type, JsonSerializerContext, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в <code>.returnType</code> . Поток будет считан до завершения.
<code>DeserializeAsync(Stream, JsonTypeInfo, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в <code>jsonTypeInfo</code> . Поток будет считан до завершения.
<code>DeserializeAsync(Stream, Type, JsonSerializerOptions, CancellationToken)</code>	Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр указанного типа. Поток считывается до завершения.
<code>DeserializeAsync< TValue > (Stream, JsonTypeInfo< TValue >, CancellationToken)</code>	Считывает текст в кодировке UTF-8, представляющий одно значение JSON в <code>.TValue</code> . Поток будет считан до завершения.
<code>DeserializeAsync< TValue > (Stream, JsonSerializerOptions, CancellationToken)</code>	Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа. Поток считывается до завершения.

DeserializeAsync(Stream, Type, JsonSerializerContext, CancellationToken)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON в `.returnType`. Поток будет считан до завершения.

C#

```
public static System.Threading.Tasks.ValueTask<object?> DeserializeAsync  
(System.IO.Stream utf8Json, Type returnType,
```

```
System.Text.Json.Serialization.JsonSerializerContext context,  
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json [Stream](#)

Данные JSON для анализа.

returnType [Type](#)

Тип объекта для преобразования и возврата.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

cancellationToken [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции чтения.

Возвращаемое значение

[ValueTask<Object>](#)

Представление `returnType` данного значения JSON.

Исключения

[ArgumentNullException](#)

Значение параметра `utf8Json`, `returnType` или `context` равно `null`.

[JsonException](#)

Json недопустим `returnType`, несовместим с JSON или в потоке остались данные.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) в предоставленном `context` не вернул совместимый [JsonTypeInfo](#) для `returnType`.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как [ArgumentException](#), по-прежнему создаются синхронно. Хранимые исключения см. в разделе исключения, создаваемые [Deserialize\(Stream, Type, JsonSerializerContext\)](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

DeserializeAsync(Stream, JsonTypeInfo, CancellationToken)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON, в экземпляр, указанный в `jsonTypeInfo`. Поток будет считан до завершения.

C#

```
public static System.Threading.Tasks.ValueTask<object?> DeserializeAsync  
(System.IO.Stream utf8Json,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo,  
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json `Stream`

Данные JSON для анализа.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

`cancellationToken` [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции чтения.

Возвращаемое значение

[ValueTask<Object>](#)

Представление `jsonTypeInfo` данного значения JSON.

Исключения

[ArgumentNullException](#)

Параметр `utf8Json` или `jsonTypeInfo` имеет значение `null`.

[JsonException](#)

Json недопустим или при наличии оставшихся данных в потоке.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

DeserializeAsync(Stream, Type, JsonSerializerOptions, CancellationToken)

Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр указанного типа. Поток считывается до завершения.

C#

```
public static System.Threading.Tasks.ValueTask<object?> DeserializeAsync  
(System.IO.Stream utf8Json, Type returnType,  
System.Text.Json.JsonSerializerOptions? options = default,  
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json Stream

Анализируемые данные JSON.

returnType Type

Тип объекта для преобразования и возврата.

options JsonSerializerOptions

Параметры для управления поведением во время чтения.

cancellationToken CancellationToken

Токен отмены, который может использоваться для отмены операции чтения.

Возвращаемое значение

ValueTask<Object>

Представление returnType данного значения JSON.

Исключения

ArgumentNullException

Параметр utf8Json или returnType имеет значение null.

JsonException

Недопустимый JSON.

-ИЛИ-

TValue несовместим с JSON.

-ИЛИ-

В потоке остались данные.

NotSupportedException

Совместимые объекты [JsonConverter](#) для `returnType` или его сериализуемых членов отсутствуют.

OperationCanceledException

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как [ArgumentException](#), по-прежнему создаются синхронно. Хранимые исключения см. в разделе [исключения, создаваемые Deserialize\(Stream, Type, JsonSerializerOptions\)](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

DeserializeAsync< TValue >(Stream, JsonTypeInfo< TValue >, CancellationToken)

Считывает текст в кодировке UTF-8, представляющий одно значение JSON в `. TValue`. Поток будет считан до завершения.

C#

```
public static System.Threading.Tasks.ValueTask< TValue?>
DeserializeAsync< TValue > (System.IO.Stream utf8Json,
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
jsonTypeInfo, System.Threading.CancellationToken cancellationToken =
default);
```

Параметры типа

TValue

Тип для десериализации значения JSON.

Параметры

utf8Json Stream

Данные JSON для анализа.

jsonTypeInfo JsonTypeInfo<TValue>

Метаданные о преобразуемом типе.

cancellationToken CancellationToken

Объект , CancellationToken который может использоваться для отмены операции чтения.

Возвращаемое значение

ValueTask<TValue>

Представление TValue данного значения JSON.

Исключения

ArgumentNullException

Параметр utf8Json или jsonTypeInfo имеет значение null.

JsonException

Json является недопустимым, TValue несовместим с JSON или в потоке есть оставшиеся данные.

NotSupportedException

Совместимые объекты JsonConverter для TValue или его сериализуемых членов отсутствуют.

OperationCanceledException

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

DeserializeAsync< TValue >(Stream, JsonSerializerOptions, CancellationToken)

Асинхронное чтение текста в кодировке UTF-8, представляющего одно значение JSON, в экземпляр типа, заданного параметром универсального типа. Поток считывается до завершения.

C#

```
public static System.Threading.Tasks.ValueTask<TValue?>
DeserializeAsync<TValue> (System.IO.Stream utf8Json,
System.Text.Json.JsonSerializerOptions? options = default,
System.Threading.CancellationToken cancellationToken = default);
```

Параметры типа

TValue

Целевой тип значения JSON.

Параметры

utf8Json [Stream](#)

Анализируемые данные JSON.

options [JsonSerializerOptions](#)

Параметры для управления поведением во время чтения.

cancellationToken [CancellationToken](#)

Токен, который можно использовать для отмены операции чтения.

Возвращаемое значение

[ValueTask](#)<TValue>

Представление **TValue** данного значения JSON.

Исключения

JsonException

Недопустимый JSON.

-ИЛИ-

`TValue` несовместим с JSON.

-ИЛИ-

В потоке остались данные.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

ArgumentNullException

`utf8Json` имеет значение `null`.

OperationCanceledException

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.DeserializeAsync Enumerable Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

<code>DeserializeAsyncEnumerable<TValue>(Stream, JsonSerializerOptions, CancellationToken)</code>	Заключает текст в кодировке UTF-8 в IAsyncEnumerable<T> объект , который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.
<code>DeserializeAsyncEnumerable<TValue>(Stream, JsonTypeInfo<TValue>, CancellationToken)</code>	Заключает текст в кодировке UTF-8 в IAsyncEnumerable<T> объект , который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.

DeserializeAsyncEnumerable<TValue>(Stream, JsonSerializerOptions, CancellationToken)

Заключает текст в кодировке UTF-8 в [IAsyncEnumerable<T>](#) объект , который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.

C#

```
public static System.Collections.Generic.IAsyncEnumerable<TValue?>
DeserializeAsyncEnumerable<TValue> (System.IO.Stream utf8Json,
System.Text.Json.JsonSerializerOptions? options = default,
System.Threading.CancellationToken cancellationToken = default);
```

Параметры типа

TValue

Тип элемента для асинхронной десериализации.

Параметры

`utf8Json` `Stream`

Данные JSON для анализа.

`options` `JsonSerializerOptions`

Параметры для управления поведением во время чтения.

`cancellationToken` `CancellationToken`

Объект `CancellationToken`, который может использоваться для отмены операции чтения.

Возвращаемое значение

`IAsyncEnumerable<TValue>`

Представление `IAsyncEnumerable<T>` предоставленного массива JSON.

Исключения

`ArgumentNullException`

`utf8Json` имеет значение `null`.

`OperationCanceledException`

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

DeserializeAsyncEnumerable<TValue>(Stream, JsonTypeInfo<TValue>, CancellationToken)

Заключает текст в кодировке UTF-8 в `IAsyncEnumerable<T>` объект, который можно использовать для десериализации массивов JSON корневого уровня в потоковой передаче.

C#

```
public static System.Collections.Generic.IAsyncEnumerable<TValue?>
DeserializeAsyncEnumerable<TValue> (System.IO.Stream utf8Json,
System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue>
jsonTypeInfo, System.Threading.CancellationToken cancellationToken =
default);
```

Параметры типа

TValue

Тип элемента для асинхронной десериализации.

Параметры

utf8Json [Stream](#)

Данные JSON для анализа.

jsonTypeInfo [JsonTypeInfo<TValue>](#)

Метаданные о преобразуемом типе элемента.

cancellationToken [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции чтения.

Возвращаемое значение

[IAsyncEnumerable<TValue>](#)

Представление [IAsyncEnumerable<T>](#) предоставленного массива JSON.

Исключения

[ArgumentNullException](#)

Параметр `utf8Json` или `jsonTypeInfo` имеет значение `null`.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8 и .NET 7

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.Serialize Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

Serialize(Utf8JsonWriter, Object, Type, JsonSerializerContext)	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.
Serialize(Stream, Object, Type, JsonSerializerContext)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
Serialize(Stream, Object, Type, JsonSerializerOptions)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
Serialize(Utf8JsonWriter, Object, JsonTypeInfo)	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.
Serialize(Utf8JsonWriter, Object, Type, JsonSerializerOptions)	Записывает представление JSON указанного типа в предоставленный модуль записи.
Serialize(Object, Type, JsonSerializerOptions)	Преобразует значение указанного типа в строку JSON.
Serialize(Stream, Object, JsonTypeInfo)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
Serialize(Object, JsonTypeInfo)	Преобразует предоставленное значение в String .
Serialize(Object, Type, JsonSerializerContext)	Преобразует предоставленное значение в String .
Serialize< TValue >(TValue, JsonSerializerOptions)	Преобразует значение типа, заданного параметром универсального типа, в строку JSON.
Serialize< TValue >(TValue, JsonTypeInfo< TValue >)	Преобразует предоставленное значение в String .
Serialize< TValue >(Stream, TValue, JsonSerializerOptions)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .

<code>Serialize<TValue>(Stream, TValue, JsonTypeInfo<TValue>)</code>	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в <code>Stream</code> .
<code>Serialize<TValue>(Utf8JsonWriter, TValue, JsonSerializerOptions)</code>	Записывает представление JSON типа, указанного параметром универсального типа, в предоставленный модуль записи.
<code>Serialize<TValue>(Utf8JsonWriter, TValue, JsonTypeInfo<TValue>)</code>	Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.

Serialize(Utf8JsonWriter, Object, Type, JsonSerializerContext)

Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.

C#

```
public static void Serialize (System.Text.Json.Utf8JsonWriter writer,
object? value, Type inputType,
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

writer `Utf8JsonWriter`

Модуль записи JSON, в который производится запись.

value `Object`

Значение для преобразования и записи.

inputType `Type`

Тип `value` для преобразования.

context `JsonSerializerContext`

Поставщик метаданных для сериализуемых типов.

Исключения

`ArgumentException`

Параметр `inputType` несовместим с параметром `value`.

[ArgumentException](#)

Параметр `writer` или `inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод `GetTypeInfo(Type)` предоставленного `context` возвращает `null` преобразуемый тип.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize(Stream, Object, Type, JsonSerializerContext)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в `Stream`.

C#

```
public static void Serialize (System.IO.Stream utf8Json, object? value,
Type inputType, System.Text.Json.Serialization.JsonSerializerContext
context);
```

Параметры

`utf8Json` `Stream`

UTF-8 `Stream` для записи.

`value` `Object`

Преобразуемое значение.

inputType Type

Тип `value` для преобразования.

context JsonSerializerContext

Поставщик метаданных для сериализуемых типов.

Исключения

ArgumentException

Параметр `inputType` несовместим с параметром `value`.

ArgumentNullException

Значение параметра `utf8Json`, `inputType` или `context` равно `null`.

NotSupportedException

Совместимые объекты `JsonConverter` для `inputType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize(Stream, Object, Type, JsonSerializerOptions)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static void Serialize (System.IO.Stream utf8Json, object? value,
    Type inputType, System.Text.Json.JsonSerializerOptions? options =
    default);
```

Параметры

utf8Json Stream

UTF-8 [Stream](#) для записи.

value Object

Преобразуемое значение.

inputType Type

Тип `value` для преобразования.

options JsonSerializerOptions

Параметры для управления поведением преобразования.

Исключения

ArgumentException

Параметр `inputType` несовместим с параметром `value`.

ArgumentNullException

Параметр `utf8Json` или `inputType` имеет значение `null`.

NotSupportedException

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize(Utf8JsonWriter, Object, JsonTypeInfo)

Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.

C#

```
public static void Serialize (System.Text.Json.Utf8JsonWriter writer,  
object? value, System.Text.Json.Serialization.Metadata.JsonTypeInfo
```

```
jsonTypeInfo);
```

Параметры

writer [Utf8JsonWriter](#)

Записываемый модуль.

value [Object](#)

Значение для преобразования и записи.

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Исключения

[ArgumentNullException](#)

Параметр `writer` или `jsonTypeInfo` имеет значение `null`.

[InvalidCastException](#)

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Serialize(Utf8JsonWriter, Object, Type, JsonSerializerOptions)

Записывает представление JSON указанного типа в предоставленный модуль записи.

C#

```
public static void Serialize (System.Text.Json.Utf8JsonWriter writer,  
    object? value, Type inputType, System.Text.Json.JsonSerializerOptions?  
    options = default);
```

Параметры

writer [Utf8JsonWriter](#)

Модуль записи JSON, в который производится запись.

value [Object](#)

Значение для преобразования и записи.

inputType [Type](#)

Тип `value` для преобразования.

options [JsonSerializerOptions](#)

Параметры управления поведением сериализации.

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`

[ArgumentNullException](#)

Параметр `writer` или `inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `inputType` или его сериализуемых членов отсутствуют.

Комментарии

Объект `JsonWriterOptions` используемый для создания экземпляра `Utf8JsonWriter` имеет приоритет над `JsonSerializerOptions`, когда они конфликтуют. Следовательно, `JsonWriterOptions.Indented` при записи используются, `JsonWriterOptions.SkipValidation` и `JsonWriterOptions.Encoder`.

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Serialize(Object, Type, JsonSerializerOptions)

Преобразует значение указанного типа в строку JSON.

C#

```
public static string Serialize (object? value, Type inputType,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[String](#)

Строковое представление JSON значения.

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`.

[ArgumentNullException](#)

`inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

Комментарии

[String](#) Использование не так эффективно, как использование кодировки UTF-8, так как реализация внутренне использует UTF-8. См. также [SerializeToUtf8Bytes\(Object, Type, JsonSerializerOptions\)](#) и [SerializeAsync\(Stream, Object, Type, JsonSerializerOptions, CancellationToken\)](#).

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Serialize(Stream, Object, JsonTypeInfo)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static void Serialize (System.IO.Stream utf8Json, object? value,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

utf8Json [Stream](#)

UTF-8 [Stream](#) для записи.

value [Object](#)

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

InvalidOperationException

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Serialize(Object, JsonTypeInfo)

Преобразует предоставленное значение в [String](#).

C#

```
public static string Serialize (object? value,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

`value` [Object](#)

Преобразуемое значение.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[String](#)

Представление [String](#) значения.

Исключения

ArgumentNullException

`jsonTypeInfo` имеет значение `null`.

InvalidOperationException

`value` не соответствует типу `jsonTypeInfo`.

Комментарии

[String](#) Использование не так эффективно, как использование кодировки UTF-8, так как реализация внутренне использует UTF-8. См. также [SerializeToUtf8Bytes\(Object, JsonTypeInfo\)](#) и [SerializeAsync\(Stream, Object, JsonTypeInfo, CancellationToken\)](#).

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Serialize(Object, Type, JsonSerializerContext)

Преобразует предоставленное значение в [String](#).

C#

```
public static string Serialize (object? value, Type inputType,  
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

`value` [Object](#)

Преобразуемое значение.

`inputType` [Type](#)

Тип `value` для преобразования.

`context` [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[String](#)

Представление [String](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

[ArgumentNullException](#)

Параметр `inputType` или `context` имеет значение `null`.

Комментарии

[String](#) Использование не так эффективно, как использование кодировки UTF-8, так как реализация внутренне использует UTF-8. Также см. [SerializeToUtf8Bytes\(Object, Type, JsonSerializerContext\)](#) и [SerializeAsync\(Stream, Object, Type, JsonSerializerContext, CancellationToken\)](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

[Serialize< TValue >\(TValue, JsonSerializerOptions\)](#)

Преобразует значение типа, заданного параметром универсального типа, в строку JSON.

C#

```
public static string Serialize< TValue> (TValue value,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

options JsonSerializerOptions

Параметры управления поведением сериализации.

Возвращаемое значение

String

Строковое представление JSON значения.

Исключения

NotSupportedException

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

[String](#) Использование не так эффективно, как использование кодировки UTF-8, так как реализация внутренне использует UTF-8. См. также [SerializeToUtf8Bytes\(Object, Type, JsonSerializerOptions\)](#) и [SerializeAsync\(Stream, Object, Type, JsonSerializerOptions, CancellationToken\)](#).

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Serialize< TValue >(TValue, JsonTypeInfo< TValue >)

Преобразует предоставленное значение в [String](#).

C#

```
public static string Serialize< TValue > (TValue value,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >  
jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[String](#)

Представление [String](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

Комментарии

`String` Использование не так эффективно, как использование кодировки UTF-8, так как реализация внутренне использует UTF-8. См. также `SerializeToUtf8Bytes< TValue >(TValue, JsonTypeInfo< TValue >)` и `SerializeAsync< TValue >(Stream, TValue, JsonTypeInfo< TValue >, CancellationToken)`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize< TValue >(Stream, TValue, JsonSerializerOptions)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в `Stream`.

C#

```
public static void Serialize< TValue > (System.IO.Stream utf8Json, TValue value, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

`TValue`

Тип сериализуемого значения.

Параметры

`utf8Json` `Stream`

UTF-8 `Stream` для записи.

`value` `TValue`

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize<TValue>(Stream, TValue, JsonTypeInfo<TValue>)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static void Serialize<TValue> (System.IO.Stream utf8Json, TValue value, System.Text.Json.Serialization.Metadata.JsonTypeInfo<TValue> jsonTypeInfo);
```

Параметры типа

`TValue`

Тип сериализуемого значения.

Параметры

`utf8Json` [Stream](#)

UTF-8 [Stream](#) для записи.

value TValue

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo](#)<TValue>

Метаданные о преобразуемом типе.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Serialize<TValue>(Utf8JsonWriter, TValue, JsonSerializerOptions)

Записывает представление JSON типа, указанного параметром универсального типа, в предоставленный модуль записи.

C#

```
public static void Serialize<TValue> (System.Text.Json.Utf8JsonWriter  
writer, TValue value, System.Text.Json.JsonSerializerOptions? options =  
default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

writer `Utf8JsonWriter`

Модуль записи JSON, в который производится запись.

value `TValue`

Значение для преобразования и записи.

options `JsonSerializerOptions`

Параметры управления поведением сериализации.

Исключения

`ArgumentNullException`

`writer` имеет значение `null`.

`NotSupportedException`

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

Комментарии

Объект , `JsonWriterOptions` используемый для создания экземпляра , `Utf8JsonWriter` имеет приоритет над `JsonSerializerOptions` , когда они конфликтуют. Следовательно, `JsonWriterOptions.Indented` при записи используются , `JsonWriterOptions.SkipValidation`и `JsonWriterOptions.Encoder` .

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Serialize< TValue >(Utf8JsonWriter, TValue, JsonTypeInfo< TValue >)

Записывает одно значение JSON (включая объекты или массивы) в предоставленный модуль записи.

C#

```
public static void Serialize< TValue > (System.Text.Json.Utf8JsonWriter writer, TValue value, System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue > jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

writer [Utf8JsonWriter](#)

Записываемый модуль.

value [TValue](#)

Значение для преобразования и записи.

jsonTypeInfo [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Исключения

[ArgumentNullException](#)

Параметр `writer` или `jsonTypeInfo` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.SerializeAsync Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

SerializeAsync(Stream, Object, Type, JsonSerializerContext, CancellationToken)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
SerializeAsync(Stream, Object, JsonTypeInfo, CancellationToken)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
SerializeAsync(Stream, Object, Type, JsonSerializerOptions, CancellationToken)	Асинхронно преобразует значение указанного типа в текст JSON в кодировке UTF-8 и записывает его в указанный поток.
SerializeAsync< TValue > (Stream, TValue, JsonTypeInfo< TValue >, CancellationToken)	Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в Stream .
SerializeAsync< TValue > (Stream, TValue, JsonSerializerOptions, CancellationToken)	Асинхронно преобразует значение типа, заданного параметром универсального типа, в текст JSON в кодировке UTF-8 и записывает его в поток.

SerializeAsync(Stream, Object, Type, JsonSerializerContext, CancellationToken)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static System.Threading.Tasks.Task SerializeAsync  
(System.IO.Stream utf8Json, object? value, Type inputType,
```

```
System.Text.Json.Serialization.JsonSerializerContext context,  
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json [Stream](#)

UTF-8 [Stream](#) для записи.

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип [value](#) для преобразования.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

cancellationToken [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции записи.

Возвращаемое значение

[Task](#)

Задача, представляющая асинхронную операцию записи.

Исключения

[ArgumentException](#)

Параметр [inputType](#) несовместим с параметром [value](#).

[ArgumentNullException](#)

Значение параметра [utf8Json](#), [inputType](#) или [context](#) равно `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для [inputType](#) или его сериализуемых членов отсутствуют.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как [ArgumentException](#), по-прежнему создаются синхронно. Хранимые исключения см. в разделе исключения, создаваемые [Serialize\(Stream, Object, Type, JsonSerializerContext\)](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeAsync(Stream, Object, JsonTypeInfo, CancellationToken)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static System.Threading.Tasks.Task SerializeAsync
(System.IO.Stream utf8Json, object? value,
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo,
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json [Stream](#)

UTF-8 [Stream](#) для записи.

value [Object](#)

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

cancellationToken [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции записи.

Возвращаемое значение

[Task](#)

Задача, представляющая асинхронную операцию записи.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

[InvalidOperationException](#)

`value` не соответствует типу `jsonTypeInfo`.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

SerializeAsync(Stream, Object, Type, JsonSerializerOptions, CancellationToken)

Асинхронно преобразует значение указанного типа в текст JSON в кодировке UTF-8 и записывает его в указанный поток.

C#

```
public static System.Threading.Tasks.Task SerializeAsync  
(System.IO.Stream utf8Json, object? value, Type inputType,  
System.Text.Json.JsonSerializerOptions? options = default,  
System.Threading.CancellationToken cancellationToken = default);
```

Параметры

utf8Json Stream

Поток UTF-8, в который требуется выполнить запись.

value Object

Преобразуемое значение.

inputType Type

Тип **value** для преобразования.

options JsonSerializerOptions

Параметры управления поведением сериализации.

cancellationToken CancellationToken

Токен, который можно использовать для отмены операции записи.

Возвращаемое значение

Task

Задача, представляющая асинхронную операцию записи.

Исключения

[ArgumentException](#)

Параметр **inputType** несовместим с параметром **value**.

[ArgumentNullException](#)

Параметр **utf8Json** или **inputType** имеет значение **null**.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **inputType** или его сериализуемых членов отсутствуют.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как [ArgumentException](#), по-прежнему создаются синхронно. Хранимые исключения см. в разделе исключения, создаваемые [Serialize\(Stream, Object, Type, JsonSerializerOptions\)](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

SerializeAsync< TValue >(Stream, TValue, JsonTypeInfo< TValue >, CancellationToken)

Преобразует предоставленное значение в текст JSON в кодировке UTF-8 и записывает его в [Stream](#).

C#

```
public static System.Threading.Tasks.Task SerializeAsync< TValue >
    (System.IO.Stream utf8Json, TValue value,
     System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
     jsonTypeInfo, System.Threading.CancellationToken cancellationToken =
     default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

utf8Json [Stream](#)

UTF-8 [Stream](#) для записи.

value [TValue](#)

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo<TValue>](#)

Метаданные о преобразуемом типе.

cancellationToken [CancellationToken](#)

Объект [CancellationToken](#), который можно использовать для отмены операции записи.

Возвращаемое значение

[Task](#)

Задача, представляющая асинхронную операцию записи.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `TValue` или его сериализуемых членов отсутствуют.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeAsync< TValue >(Stream, TValue, JsonSerializerOptions, CancellationToken)

Асинхронно преобразует значение типа, заданного параметром универсального типа, в текст JSON в кодировке UTF-8 и записывает его в поток.

C#

```
public static System.Threading.Tasks.Task SerializeAsync< TValue >(  
    System.IO.Stream utf8Json, TValue value,  
    System.Text.Json.JsonSerializerOptions? options = default,  
    System.Threading.CancellationToken cancellationToken = default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

utf8Json [Stream](#)

Поток UTF-8, в который требуется выполнить запись.

value [TValue](#)

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры управления поведением сериализации.

cancellationToken [CancellationToken](#)

Токен, который можно использовать для отмены операции записи.

Возвращаемое значение

[Task](#)

Задача, представляющая асинхронную операцию записи.

Исключения

ArgumentNullException

`utf8Json` имеет значение `null`.

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

OperationCanceledException

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.SerializeToDocument

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

SerializeToDocument(Object, JsonTypeInfo)	Преобразует предоставленное значение в JsonDocument .
SerializeToDocument(Object, Type, JsonSerializerOptions)	Преобразует предоставленное значение в JsonDocument .
SerializeToDocument(Object, Type, JsonSerializerContext)	Преобразует предоставленное значение в JsonDocument .
SerializeToDocument< TValue > (TValue, JsonSerializerOptions)	Преобразует предоставленное значение в JsonDocument .
SerializeToDocument< TValue > (TValue, JsonTypeInfo< TValue >)	Преобразует предоставленное значение в JsonDocument .

SerializeToDocument(Object, JsonTypeInfo)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonDocument SerializeToDocument (object? value, System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

value [Object](#)

Преобразуемое значение.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonDocument](#)

Представление [JsonDocument](#) значения .

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

[InvalidOperationException](#)

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

SerializeToDocument(Object, Type, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonDocument SerializeToDocument (object? value, Type inputType, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

`value` [Object](#)

Преобразуемое значение.

`inputType` Type

Тип `value` для преобразования.

`options` JsonSerializerOptions

Параметры для управления поведением преобразования.

Возвращаемое значение

[JsonDocument](#)

Представление [JsonDocument](#) значения .

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`.

[ArgumentNullException](#)

`inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToDocument(Object, Type, JsonSerializerContext)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonDocument SerializeToDocument (object?  
value, Type inputType,
```

```
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[JsonDocument](#)

Представление `JsonDocument` значения .

Исключения

[NotSupportedException](#)

Совместимые объекты `JsonConverter` для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод `GetTypeInfo(Type)` предоставленного `context` возвращает `null` преобразуемый тип.

[ArgumentNullException](#)

Параметр `inputType` или `context` имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToDocument< TValue >(TValue, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonDocument SerializeToDocument< TValue >(TValue value, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[JsonDocument](#)

Представление [JsonDocument](#) данного значения JSON.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToDocument< TValue >(TValue, JsonTypeInfo< TValue >)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonDocument SerializeToDocument< TValue >
(TValue value,
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonDocument](#)

Представление [JsonDocument](#) значения .

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

ArgumentNullException

`jsonTypeInfo` имеет значение `null`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.SerializeToElement Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

SerializeToElement(Object, JsonTypeInfo)	Преобразует предоставленное значение в JsonElement .
SerializeToElement(Object, Type, JsonSerializerOptions)	Преобразует предоставленное значение в JsonDocument .
SerializeToElement(Object, Type, JsonSerializerContext)	Преобразует предоставленное значение в JsonDocument .
SerializeToElement< TValue > (TValue, JsonTypeInfo< TValue >)	Преобразует предоставленное значение в JsonDocument .
SerializeToElement< TValue > (TValue, JsonSerializerOptions)	Преобразует предоставленное значение в JsonDocument .

SerializeToElement(Object, JsonTypeInfo)

Преобразует предоставленное значение в [JsonElement](#).

C#

```
public static System.Text.Json.JsonElement SerializeToElement (object? value, System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

value [Object](#)

Преобразуемое значение.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonElement](#)

Представление [JsonElement](#) значения.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

[InvalidOperationException](#)

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

SerializeToElement(Object, Type, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonElement SerializeToElement (object?  
value, Type inputType, System.Text.Json.JsonSerializerOptions? options =  
default);
```

Параметры

`value` [Object](#)

Преобразуемое значение.

`inputType` Type

Тип `value` для преобразования.

`options` JsonSerializerOptions

Параметры для управления поведением преобразования.

Возвращаемое значение

JsonElement

Представление JsonDocument значения.

Исключения

ArgumentException

Параметр `inputType` несовместим с параметром `value`.

ArgumentNullException

`inputType` имеет значение `null`.

NotSupportedException

Совместимые объекты JsonConverter для `inputType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToElement(Object, Type, JsonSerializerContext)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonElement SerializeToElement (object?  
value, Type inputType,
```

```
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[JsonElement](#)

Представление [JsonDocument](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

[ArgumentNullException](#)

Параметр `inputType` или `context` имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToElement< TValue >(TValue, JsonTypeInfo< TValue >)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonElement SerializeToElement< TValue >
(TValue value,
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

jsonTypeInfo JsonTypeInfo< TValue >

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonElement](#)

Представление [JsonDocument](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

[ArgumentNullException](#)

jsonTypeInfo имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToElement< TValue >(TValue, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonDocument](#).

C#

```
public static System.Text.Json.JsonElement SerializeToElement< TValue >(TValue value, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[JsonElement](#)

Представление [JsonDocument](#) данного значения JSON.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.SerializeToNode Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

<code>SerializeToNode(Object, JsonTypeInfo)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode(Object, Type, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode(Object, Type, JsonSerializerContext)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode< TValue > (TValue, JsonSerializerOptions)</code>	Преобразует предоставленное значение в JsonNode .
<code>SerializeToNode< TValue > (TValue, JsonTypeInfo< TValue >)</code>	Преобразует предоставленное значение в JsonNode .

SerializeToNode(Object, JsonTypeInfo)

Преобразует предоставленное значение в [JsonNode](#).

C#

```
public static System.Text.Json.Nodes.JsonNode? SerializeToNode (object?
value, System.Text.Json.Serialization.Metadata.JsonTypeInfo
jsonTypeInfo);
```

Параметры

value [Object](#)

Преобразуемое значение.

`jsonTypeInfo` [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonNode](#)

Представление [JsonNode](#) значения.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

[InvalidOperationException](#)

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

SerializeToNode(Object, Type, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonNode](#).

C#

```
public static System.Text.Json.Nodes.JsonNode? SerializeToNode (object?  
value, Type inputType, System.Text.Json.JsonSerializerOptions? options =  
default);
```

Параметры

`value` [Object](#)

Преобразуемое значение.

`inputType` Type

Тип `value` для преобразования.

`options` JsonSerializerOptions

Параметры для управления поведением преобразования.

Возвращаемое значение

[JsonNode](#)

Представление [JsonNode](#) значения.

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`.

[ArgumentNullException](#)

`inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToNode(Object, Type, JsonSerializerContext)

Преобразует предоставленное значение в [JsonNode](#).

C#

```
public static System.Text.Json.Nodes.JsonNode? SerializeToNode (object?  
value, Type inputType,
```

```
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[JsonNode](#)

Представление [JsonNode](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

[ArgumentNullException](#)

Параметр `inputType` или `context` имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToNode< TValue >(TValue, JsonSerializerOptions)

Преобразует предоставленное значение в [JsonNode](#).

C#

```
public static System.Text.Json.Nodes.JsonNode? SerializeToNode< TValue >(TValue value, System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[JsonNode](#)

Представление [JsonNode](#) данного значения JSON.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToNode< TValue >(TValue, JsonTypeInfo< TValue >)

Преобразует предоставленное значение в [JsonNode](#).

C#

```
public static System.Text.Json.Nodes.JsonNode? SerializeToNode< TValue >
(TValue value,
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >
jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo< TValue >](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[JsonNode](#)

Представление [JsonNode](#) значения.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

ArgumentNullException

`jsonTypeInfo` имеет значение `null`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializer.SerializeToUtf8Bytes

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

SerializeToUtf8Bytes(Object, JsonTypeInfo)	Преобразует указанное значение в Byte массив.
SerializeToUtf8Bytes(Object, Type, JsonSerializerOptions)	Преобразует значение указанного типа в строку JSON, закодированную как байты UTF-8.
SerializeToUtf8Bytes(Object, Type, JsonSerializerContext)	Преобразует указанное значение в Byte массив.
SerializeToUtf8Bytes< TValue > (TValue, JsonSerializerOptions)	Преобразует значение типа, указанного параметром универсального типа, в строку JSON, закодированную как байты UTF-8.
SerializeToUtf8Bytes< TValue > (TValue, JsonTypeInfo< TValue >)	Преобразует указанное значение в Byte массив.

SerializeToUtf8Bytes(Object, JsonTypeInfo)

Преобразует указанное значение в [Byte](#) массив.

C#

```
public static byte[] SerializeToUtf8Bytes (object? value,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo jsonTypeInfo);
```

Параметры

value [Object](#)

Преобразуемое значение.

jsonTypeInfo [JsonTypeInfo](#)

Метаданные о преобразуемом типе.

Возвращаемое значение

[Byte\[\]](#)

Представление значения в кодировке UTF-8.

Исключения

[ArgumentNullException](#)

`jsonTypeInfo` имеет значение `null`.

[InvalidOperationException](#)

`value` не соответствует типу `jsonTypeInfo`.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

SerializeToUtf8Bytes(Object, Type, JsonSerializerOptions)

Преобразует значение указанного типа в строку JSON, закодированную как байты UTF-8.

C#

```
public static byte[] SerializeToUtf8Bytes (object? value, Type inputType,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[Byte\[\]](#)

Строковое представление JSON значения, закодированное как байты UTF-8.

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`.

[ArgumentNullException](#)

`inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

[SerializeToUtf8Bytes\(Object, Type, JsonSerializerContext\)](#)

Преобразует указанное значение в [Byte](#) массив.

C#

```
public static byte[] SerializeToUtf8Bytes (object? value, Type inputType,  
System.Text.Json.Serialization.JsonSerializerContext context);
```

Параметры

value [Object](#)

Преобразуемое значение.

inputType [Type](#)

Тип `value` для преобразования.

context [JsonSerializerContext](#)

Поставщик метаданных для сериализуемых типов.

Возвращаемое значение

[Byte](#)[]

Представление значения в кодировке UTF-8.

Исключения

[ArgumentException](#)

Параметр `inputType` несовместим с параметром `value`.

[ArgumentNullException](#)

`inputType` имеет значение `null`.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `inputType` или его сериализуемых членов отсутствуют.

[InvalidOperationException](#)

Метод [GetTypeInfo\(Type\)](#) предоставленного `context` возвращает `null` преобразуемый тип.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

SerializeToUtf8Bytes< TValue >(TValue, JsonSerializerOptions)

Преобразует значение типа, указанного параметром универсального типа, в строку JSON, закодированную как байты UTF-8.

C#

```
public static byte[] SerializeToUtf8Bytes< TValue > (TValue value,  
System.Text.Json.JsonSerializerOptions? options = default);
```

Параметры типа

TValue

Тип значения.

Параметры

value **TValue**

Преобразуемое значение.

options [JsonSerializerOptions](#)

Параметры для управления поведением преобразования.

Возвращаемое значение

[Byte\[\]](#)

Строковое представление JSON значения, закодированное как байты UTF-8.

Исключения

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для **TValue** или его сериализуемых членов отсутствуют.

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

SerializeToUtf8Bytes< TValue >(TValue, JsonTypeInfo< TValue >)

Преобразует указанное значение в [Byte](#) массив.

C#

```
public static byte[] SerializeToUtf8Bytes< TValue > (TValue value,  
System.Text.Json.Serialization.Metadata.JsonTypeInfo< TValue >  
jsonTypeInfo);
```

Параметры типа

TValue

Тип сериализуемого значения.

Параметры

value TValue

Преобразуемое значение.

jsonTypeInfo JsonTypeInfo< TValue >

Метаданные о преобразуемом типе.

Возвращаемое значение

Byte[]

Представление значения в кодировке UTF-8.

Исключения

NotSupportedException

Совместимые объекты `JsonConverter` для `TValue` или его сериализуемых членов отсутствуют.

ArgumentNullException

`jsonTypeInfo` имеет значение `null`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerDefaults Перечисление

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Задает параметры сериализации по умолчанию на основе сценариев, которые можно использовать для создания экземпляра [JsonSerializerOptions](#).

C#

```
public enum JsonSerializerDefaults
```

Наследование [Object](#) → [ValueType](#) → [Enum](#) → JsonSerializerDefaults

Поля

General 0 Значения параметров общего назначения. Это те же параметры, которые применяются, если элемент [JsonSerializerDefaults](#) не указывается.

Сведения о применяемых значениях свойств по умолчанию см. в разделе [Свойства JsonSerializerOptions](#).

Web 1 Значения параметров, соответствующие веб-сценариям.

Этот элемент подразумевает, что:

- в именах свойств не учитывается регистр;
- для имен должно использоваться форматирование "camelCase";
- разрешены числа в кавычках (строки JSON для числовых свойств).

Применяется к

Продукт	Версии
.NET	5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет параметры для использования с [JsonSerializer](#).

C#

```
public sealed class JsonSerializerOptions
```

Наследование [Object](#) → JsonSerializerOptions

Комментарии

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Конструкторы

JsonSerializerOptions()	Инициализирует новый экземпляр класса JsonSerializerOptions .
JsonSerializerOptions(JsonSerializerDefaults)	Конструирует новый экземпляр JsonSerializerOptions с предопределенным набором параметров, определяемых указанным JsonSerializerDefaults .
JsonSerializerOptions(JsonSerializerOptions)	Копирует параметры из экземпляра JsonSerializerOptions в новый экземпляр.

Свойства

AllowTrailingCommas	Возвращает или задает значение, которое указывает, разрешена ли (и игнорируется) лишняя запятая в конце списка значений JSON в объекте или массиве внутри десериализуемых полезных данных JSON.
-------------------------------------	---

Converters	Возвращает список зарегистрированных пользовательских преобразователей.
Default	Возвращает одноэлементный экземпляр JsonSerializerOptions класса только для чтения, использующий конфигурацию по умолчанию.
DefaultBufferSize	Возвращает или задает размер буфера по умолчанию (в байтах), используемый при создании временных буферов.
DefaultIgnoreCondition	Возвращает или задает значение, определяющее, когда свойства со значениями по умолчанию игнорируются во время сериализации или десериализации. Значение по умолчанию — Never .
DictionaryKeyPolicy	Возвращает или задает политику, используемую для преобразования имени ключа IDictionary в другой формат, например Camel.
Encoder	Возвращает или устанавливает кодировщик, используемый при экранировании строк. Укажите значение <code>null</code> для использования кодировщика по умолчанию.
IgnoreNullValues	Является устаревшей. Возвращает или задает значение, указывающее, игнорируются ли <code>null</code> значения во время сериализации и десериализации. Значение по умолчанию — <code>false</code> .
IgnoreReadOnlyFields	Возвращает или задает значение, указывающее, игнорируются ли доступные только для чтения поля во время сериализации. Если поле помечено ключевым словом <code>readonly</code> , оно доступно только для чтения. Значение по умолчанию — <code>false</code> .
IgnoreReadOnlyProperties	Возвращает значение, указывающее, игнорируются ли свойства только для чтения во время сериализации. Значение по умолчанию — <code>false</code> .
IncludeFields	Возвращает или задает значение, указывающее, обрабатываются ли поля во время сериализации и десериализации. Значение по умолчанию — <code>false</code> .
IsReadOnly	Возвращает значение, указывающее, заблокирован ли текущий экземпляр для изменения пользователем.
MaxDepth	Возвращает или задает максимальную глубину, разрешенную при сериализации или десериализации JSON, при этом значение по умолчанию 0 указывает максимальную глубину 64.

NumberHandling	Возвращает или задает объект , указывающий способ обработки числовых типов при сериализации или десериализации.
PreferredObjectCreation Handling	Возвращает или задает предпочтительную обработку создания объектов для свойств при десериализации JSON.
PropertyNameCaseInsensitive	Возвращает или задает значение, указывающее, используется ли в имени свойства сравнение без учета регистра во время десериализации. Значение по умолчанию — <code>false</code> .
PropertyNamingPolicy	Возвращает или задает значение, указывающее политику, используемую для преобразования имени свойства объекта в другой формат, например "верблюжий" стиль, или <code>null</code> , чтобы оставить имена свойств без изменений.
ReadCommentHandling	Возвращает или задает значение, определяющее, как комментарии обрабатываются во время десериализации.
ReferenceHandler	Возвращает или задает объект , указывающий, как обрабатываются ссылки на объекты при чтении и записи JSON.
TypeInfoResolver	Возвращает или задает сопоставитель контрактов, JsonTypeInfo используемый этим экземпляром.
TypeInfoResolverChain	Возвращает список сопоставителей цепочек JsonTypeInfo контрактов, используемых этим экземпляром.
UnknownTypeHandling	Возвращает или задает объект , указывающий, как десериализация типа, объявленного как , Object обрабатывается во время десериализации.
UnmappedMemberHandling	Возвращает или задает объект , указывающий, как JsonSerializer обрабатывает свойства JSON, которые не могут быть сопоставлены с определенным элементом .NET при десериализации типов объектов.
WriteIndented	Возвращает или задает значение, указывающее, следует ли использовать в ФОРМАТЕ JSON красивую печать. По умолчанию JSON сериализуется без лишних пробелов.

Методы

AddContext<TContext>()	Является устаревшей. Добавляет новый JsonSerializerContext объект к разрешению метаданных текущего JsonSerializerOptions экземпляра .
--	--

Equals(Object)	Определяет, равен ли указанный объект текущему объекту. (Унаследовано от Object)
GetConverter(Type)	Возвращает преобразователь для указанного типа.
GetHashCode()	Служит хэш-функцией по умолчанию. (Унаследовано от Object)
GetType()	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object)
GetTypeInfo(Type)	Возвращает метаданные JsonTypeInfo контракта, разрешенные текущим JsonSerializerOptions экземпляром.
MakeReadOnly()	Помечает текущий экземпляр как доступный только для чтения, чтобы предотвратить дальнейшие изменения пользователей.
MakeReadOnly(Boolean)	Помечает текущий экземпляр как доступный только для чтения, предотвращая дальнейшие изменения пользователей.
MemberwiseClone()	Создает неполную копию текущего объекта Object . (Унаследовано от Object)
ToString()	Возвращает строку, представляющую текущий объект. (Унаследовано от Object)
TryGetTypeInfo(Type, JsonTypeInfo)	Пытается получить метаданные JsonTypeInfo контракта, разрешенные текущим JsonSerializerOptions экземпляром.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

Да

Нет

Отзыв о продукте [↗](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions Конструкторы

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

JsonSerializerOptions()	Инициализирует новый экземпляр класса JsonSerializerOptions .
JsonSerializerOptions(JsonSerializerDefaults)	Конструирует новый экземпляр JsonSerializerOptions с предопределенным набором параметров, определяемых указанным JsonSerializerDefaults .
JsonSerializerOptions(JsonSerializerOptions)	Копирует параметры из экземпляра JsonSerializerOptions в новый экземпляр.

JsonSerializerOptions()

Инициализирует новый экземпляр класса [JsonSerializerOptions](#).

C#

```
public JsonSerializerOptions ();
```

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

JsonSerializerOptions(JsonSerializerDefaults)

Конструирует новый экземпляр [JsonSerializerOptions](#) с предопределенным набором параметров, определяемых указанным [JsonSerializerDefaults](#).

C#

```
public JsonSerializerOptions (System.Text.Json.JsonSerializerDefaults
defaults);
```

Параметры

defaults [JsonSerializerDefaults](#)

[JsonSerializerDefaults](#) для анализа.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	5, 6, 7, 8

JsonSerializerOptions(JsonSerializerOptions)

Копирует параметры из экземпляра [JsonSerializerOptions](#) в новый экземпляр.

C#

```
public JsonSerializerOptions (System.Text.Json.JsonSerializerOptions
options);
```

Параметры

options [JsonSerializerOptions](#)

Настраиваемый экземпляр параметров, из которого копируются свойства.

Исключения

[ArgumentNullException](#)

options имеет значение `null`.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.AllowTrailingCommas Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, которое указывает, разрешена ли (и игнорируется) лишняя запятая в конце списка значений JSON в объекте или массиве внутри десериализуемых полезных данных JSON.

C#

```
public bool AllowTrailingCommas { get; set; }
```

Значение свойства

[Boolean](#)

`true` Если дополнительная запятая в конце списка значений JSON в объекте или массиве разрешена (и игнорируется); `false` Иначе.

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

Комментарии

По умолчанию `AllowTrailingCommas` имеет значение `false`, а [JsonException](#) во время десериализации при обнаружении запятой возникает исключение .

Дополнительные сведения см. в статье [Как разрешить некоторые типы недопустимых json с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.Converters

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает список зарегистрированных пользовательских преобразователей.

C#

```
public  
System.Collections.Generic.IList<System.Text.Json.Serialization.JsonConverte  
r> Converters { get; }
```

Значение свойства

[IList<JsonConverter>](#)

Список пользовательских преобразователей.

Комментарии

После сериализации или десериализации список нельзя изменить.

Дополнительные сведения см. в статье [How to write custom converters for JSON serialization \(marshalling\) in .NET](#) (Создание пользовательских преобразователей для сериализации JSON (маршалинг) в .NET).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.Default Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает одноэлементный экземпляр [JsonSerializerOptions](#) класса только для чтения, использующий конфигурацию по умолчанию.

C#

```
public static System.Text.Json.JsonSerializerOptions Default { get; }
```

Значение свойства

[JsonSerializerOptions](#)

Комментарии

Каждый [JsonSerializerOptions](#) экземпляр инкапсулирует собственные кэши метаданных сериализации, поэтому использование новых экземпляров по умолчанию при каждом необходимости может привести к избыточному пересчету преобразователей. Это свойство предоставляет общий экземпляр, который может использоваться любым количеством компонентов без необходимости перерасчета преобразователя.

Применяется к

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.DefaultBufferSize

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает размер буфера по умолчанию (в байтах), используемый при создании временных буферов.

C#

```
public int DefaultBufferSize { get; set; }
```

Значение свойства

[Int32](#)

Размер буфера по умолчанию в байтах.

Исключения

[ArgumentException](#)

Размер буфера меньше 1.

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

Комментарии

Размер буфера по умолчанию в байтах составляет 16 384. Для большинства рабочих нагрузок размер по умолчанию — это разумное количество JSON для буферизации при чтении из потока или записи в поток. То есть он хорошо работает без создания объектов в куче больших объектов для отслеживания сборщиком мусора (GC). В сценариях без потоковой передачи увеличение размера буфера по умолчанию может повысить производительность больших строк JSON

или массивов байтов UTF-8. Рекомендуется оставить это значение без изменений, если его изменение не изменит производительность.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.DefaultIgnoreCondition Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets a value that determines when properties with default values are ignored during serialization or deserialization. The default value is [Never](#).

C#

```
public System.Text.Json.Serialization.JsonIgnoreCondition  
DefaultIgnoreCondition { get; set; }
```

Property Value

[JsonIgnoreCondition](#)

Exceptions

[ArgumentException](#)

This property is set to [Always](#).

[InvalidOperationException](#)

This property is set after serialization or deserialization has occurred.

-or-

[IgnoreNullValues](#) has been set to `true`. These properties cannot be used together.

Applies to

Product	Versions
.NET	5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonSerializerOptions.DictionaryKeyPolicy Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает политику, используемую для преобразования имени ключа [IDictionary](#) в другой формат, например Camel.

C#

```
public System.Text.Json.JsonNamingPolicy? DictionaryKeyPolicy { get; set; }
```

Значение свойства

[JsonNamingPolicy](#)

Политика, используемая для преобразования [IDictionary](#) имени ключа в другой формат.

Комментарии

Политика именования не используется при десериализации.

Дополнительные сведения см. в статье [Использование политики именования для ключей словаря](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.Encoder Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или устанавливает кодировщик, используемый при экранировании строк. Укажите значение `null` для использования кодировщика по умолчанию.

C#

```
public System.Text_ENCODINGS.Web.JavaScriptEncoder? Encoder { get; set; }
```

Значение свойства

[JavaScriptEncoder](#)

Кодировка символов JavaScript.

Комментарии

Дополнительные сведения см. в статье [Настройка кодировки символов с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

👍 Да

👎 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.IgnoreNullValues

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

⊗ Внимание!

JsonSerializerOptions.IgnoreNullValues is obsolete. To ignore null values when serializing, set DefaultIgnoreCondition to JsonIgnoreCondition.WhenWritingNull.

Возвращает или задает значение, указывающее, игнорируются ли `null` значения во время сериализации и десериализации. Значение по умолчанию — `false`.

C#

```
[System.Obsolete("JsonSerializerOptions.IgnoreNullValues is obsolete. To
ignore null values when serializing, set DefaultIgnoreCondition to
JsonIgnoreCondition.WhenWritingNull.", DiagnosticId="SYSLIB0020",
UrlFormat="https://aka.ms/dotnet-warnings/{0}")]
public bool IgnoreNullValues { get; set; }
```

Значение свойства

[Boolean](#)

`true` Значение NULL игнорируется во время сериализации и десериализации; в противном случае — `false`.

Атрибуты [ObsoleteAttribute](#)

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

-ИЛИ-

Для `DefaultIgnoreCondition` задано значение, отличное от значения по умолчанию. Эти свойства не могут использоваться совместно.

Комментарии

Дополнительные сведения см. в статье [Как игнорировать свойства](#).

Применяется к

Продукт	Версии (Устарело)
.NET	Core 3.0, Core 3.1, 5 (6, 7, 8)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.IgnoreReadOnlyFields Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее, игнорируются ли доступные только для чтения поля во время сериализации. Если поле помечено ключевым словом `readonly`, оно доступно только для чтения. Значение по умолчанию — `false`.

C#

```
public bool IgnoreReadOnlyFields { get; set; }
```

Значение свойства

[Boolean](#)

`true` Значение , если поля, доступные только для чтения, игнорируются во время сериализации; `false` Иначе.

Исключения

[InvalidOperationException](#)

Это свойство задается после сериализации или десериализации.

Комментарии

Поля только для чтения не десериализируются независимо от этого параметра.

Применяется к

Продукт	Версии
.NET	5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.IgnoreReadOnlyProperties Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение, указывающее, игнорируются ли свойства только для чтения во время сериализации. Значение по умолчанию — `false`.

C#

```
public bool IgnoreReadOnlyProperties { get; set; }
```

Значение свойства

[Boolean](#)

`true` Значение , если свойства, доступные только для чтения, игнорируются во время сериализации; в противном случае — `false`.

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

Комментарии

Свойство доступно только для чтения, если оно содержит открытый метод получения, но не является общим методом задания.

Свойства только для чтения не десериализуются независимо от этого параметра.

Дополнительные сведения см. в статье [Как игнорировать свойства с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.IncludeFields

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее, обрабатываются ли поля во время сериализации и десериализации. Значение по умолчанию — `false`.

C#

```
public bool IncludeFields { get; set; }
```

Значение свойства

[Boolean](#)

`true` Значение , если поля включены во время сериализации; в противном случае — `false`.

Исключения

[InvalidOperationException](#)

Это свойство задается после сериализации или десериализации.

Применяется к

Продукт	Версии
.NET	5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.IsReadOnly

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение, указывающее, заблокирован ли текущий экземпляр для изменения пользователем.

C#

```
public bool IsReadOnly { get; }
```

Значение свойства

[Boolean](#)

Комментарии

Экземпляр [JsonSerializerOptions](#) может быть заблокирован, если он был передан одному из [JsonSerializer](#) методов, был связан с экземпляром [JsonSerializerContext](#) или пользователь явно вызвал [MakeReadOnly\(\)](#) методы в экземпляре.

Экземпляры только для чтения используют кэширование при запросе [JsonConverter](#) и [JsonTypeInfo](#) метаданных.

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.MaxDepth

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает максимальную глубину, разрешенную при сериализации или десериализации JSON, при этом значение по умолчанию 0 указывает максимальную глубину 64.

C#

```
public int MaxDepth { get; set; }
```

Значение свойства

[Int32](#)

Максимальная глубина, допустимая при сериализации или десериализации JSON.

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

[ArgumentOutOfRangeException](#)

Максимальной глубине присвоено отрицательное значение.

Комментарии

Если пройти мимо этой глубины, возникает исключение [JsonException](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.NumberHandling Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets an object that specifies how number types should be handled when serializing or deserializing.

C#

```
public System.Text.Json.Serialization.JsonNumberHandling NumberHandling {  
    get; set; }
```

Property Value

[JsonNumberHandling](#)

Exceptions

[InvalidOperationException](#)

This property is set after serialization or deserialization has occurred.

Applies to

Product	Versions
.NET	5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonSerializerOptions.PreferredObjectCreationHandling Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает предпочтительную обработку создания объектов для свойств при десериализации JSON.

C#

```
public System.Text.Json.Serialization.JsonObjectCreationHandling  
PreferredObjectCreationHandling { get; set; }
```

Значение свойства

[JsonObjectCreationHandling](#)

Если задано значение [Populate](#), будут заполнены все свойства, которые могут повторно использовать существующий экземпляр.

Комментарии

Учитывается только тип свойства. Например, если свойство имеет тип, [IEnumerable<T>](#) но ему назначено [List<T>](#), оно не будет заполнено, так как [IEnumerable<T>](#) оно не может заполнять. Кроме того, типы значений требуют заполнения метода задания.

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.PropertyNameCaseInsensitive Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее, используется ли в имени свойства сравнение без учета регистра во время десериализации. Значение по умолчанию — `false`.

C#

```
public bool PropertyNameCaseInsensitive { get; set; }
```

Значение свойства

[Boolean](#)

`true`, если имена свойств сравниваются без учета регистра; в противном случае — `false`.

Комментарии

Существуют затраты на производительность, связанные с сравнением без учета регистра (то есть, когда `PropertyNameCaseInsensitive` имеет значение `true`).

Дополнительные сведения см. в разделе [Включение сопоставления имен свойств без учета регистра с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.PropertyNamingPolicy Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее политику, используемую для преобразования имени свойства объекта в другой формат, например "верблюжий" стиль, или `null`, чтобы оставить имена свойств без изменений.

C#

```
public System.Text.Json.JsonNamingPolicy? PropertyNamingPolicy { get; set; }
```

Значение свойства

[JsonNamingPolicy](#)

Политика именования свойств или `null`, чтобы оставить имена свойств без изменений.

Комментарии

Полученное имя свойства должно соответствовать полезным данным JSON во время десериализации и будет использоваться при записи имени свойства во время сериализации.

Политика не используется для свойств, к которым применен объект [JsonPropertyNameAttribute](#).

Дополнительные сведения см. в статье [Настройка имен и значений свойств с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.ReadCommentHandling Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, определяющее, как комментарии обрабатываются во время десериализации.

C#

```
public System.Text.Json.JsonCommentHandling ReadCommentHandling { get; set; }
```

Значение свойства

[JsonCommentHandling](#)

Значение типа , указывающее, разрешены ли комментарии, запрещены или пропущены.

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

[ArgumentOutOfRangeException](#)

Перечисление обработки комментариев имеет значение, которое не поддерживается (или находится за пределами диапазона перечисления [JsonCommentHandling](#)).

Комментарии

По умолчанию при обнаружении [JsonException](#) комментария во время десериализации возникает исключение .

Дополнительные сведения см. в статье [Как разрешить некоторые типы недопустимых json с помощью System.Text.Json](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

Отзыв о продукте [↗](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.ReferenceHandler Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets an object that specifies how object references are handled when reading and writing JSON.

C#

```
public System.Text.Json.Serialization.ReferenceHandler? ReferenceHandler {  
    get; set; }
```

Property Value

[ReferenceHandler](#)

Remarks

By default, serialization does not support objects with cycles and does not preserve duplicate references. Metadata properties will not be written when serializing reference types and will be treated as regular properties on deserialize.

- On Serialize:
 - Treats duplicate object references as if they were unique and writes all their properties.
 - The serializer throws a [JsonException](#) if an object contains a cycle.
- On Deserialize:
 - Metadata properties (`$id`, `$values`, and `$ref`) will not be consumed and therefore will be treated as regular JSON properties.
 - The metadata properties can map to a real property on the returned object if the property names match, or will be added to the [JsonExtensionDataAttribute](#) overflow dictionary, if one exists; otherwise, they are ignored.

Use [Preserve](#) to enable unique object reference preservation on serialization and metadata consumption to read preserved references on deserialization.

Applies to

Product	Versions
.NET	5, 6, 7, 8

Feedback

Was this page helpful?



[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

JsonSerializerOptions.TypeInfoResolver Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets the [JsonTypeInfo](#) contract resolver used by this instance.

C#

```
public System.Text.Json.Serialization.Metadata.IJsonTypeInfoResolver?
TypeResolver { get; set; }
```

Property Value

[IJsonTypeInfoResolver](#)

Exceptions

[InvalidOperationException](#)

The property is set after serialization or deserialization has occurred.

Remarks

A `null` setting is equivalent to using the reflection-based [DefaultJsonTypeInfoResolver](#). The property will be populated automatically once used with one of the [JsonSerializer](#) methods.

Applies to

Product	Versions
.NET	7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonSerializerOptions.TypeInfoResolverChain Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает список цепочки [JsonTypeInfo](#) сопоставителей контрактов, используемых этим экземпляром.

C#

```
public  
System.Collections.Generic.IList<System.Text.Json.Serialization.Metadata.IJsonTypeInfoResolver> TypeInfoResolverChain { get; }
```

Значение свойства

[IList<IJsonTypeInfoResolver>](#)

Комментарии

Порядок цепочки имеет важное значение: [JsonSerializerOptions](#) запросит каждый из сопоставителей в указанном порядке, возвращая первый результат, отличный от `NULL`. Если все сопоставители в цепочке возвращают `null`, то [JsonSerializerOptions](#) также возвращает `null`.

Это свойство является вспомогательным для и синхронизируется со свойством [TypeInfoResolver](#). Любое изменение, внесенное в это свойство, отражается [TypeInfoResolver](#) и наоборот.

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.UnknownTypeHandling Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает объект , указывающий, как десериализация типа, объявленного как , [Object](#) обрабатывается во время десериализации.

C#

```
public System.Text.Json.Serialization.JsonUnknownTypeHandling  
UnknownTypeHandling { get; set; }
```

Значение свойства

[JsonUnknownTypeHandling](#)

Применяется к

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.UnmappedMemberHandling Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает объект , указывающий [JsonSerializer](#) способ обработки свойств JSON, которые не могут быть сопоставлены с определенным членом .NET при десериализации типов объектов.

C#

```
public System.Text.Json.Serialization.JsonUnmappedMemberHandling  
UnmappedMemberHandling { get; set; }
```

Значение свойства

[JsonUnmappedMemberHandling](#)

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.WriteIndented Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее, следует ли использовать в ФОРМАТЕ JSON красивую печать. По умолчанию JSON сериализуется без лишних пробелов.

C#

```
public bool WriteIndented { get; set; }
```

Значение свойства

Boolean

`true` Если JSON довольно напечатан при сериализации; в противном случае — `false`. Значение по умолчанию — `false`.

Исключения

[InvalidOperationException](#)

Это свойство было задано после сериализации или десериализации.

Комментарии

Красивая печать включает в себя:

- Отступ вложенных токенов JSON.
- Добавление новых строк
- Добавление пробелов между именами и значениями свойств.

Дополнительные сведения см. в разделе [Сериализация и десериализация JSON](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.AddContext<TContext> Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

⊗ Внимание!

JsonSerializerOptions.AddContext is obsolete. To register a JsonSerializerContext, use either the TypeInfoResolver or TypeInfoResolverChain properties.

Добавляет новый [JsonSerializerContext](#) объект к разрешению метаданных текущего [JsonSerializerOptions](#) экземпляра.

C#

```
[System.Obsolete("JsonSerializerOptions.AddContext is obsolete. To register a JsonSerializerContext, use either the TypeInfoResolver or TypeInfoResolverChain properties.", DiagnosticId="SYSLIB0049",
UrlFormat="https://aka.ms/dotnet-warnings/{0}")]
public void AddContext<TContext> () where TContext :
System.Text.Json.Serialization.JsonSerializerContext, new();
```

Параметры типа

TContext

Универсальное определение указанного типа контекста.

Атрибуты [ObsoleteAttribute](#)

Комментарии

При сериализации и десериализации типов с помощью экземпляра options метаданные для типов будут извлекаться из экземпляра контекста.

Методы поддерживают добавление нескольких контекстов для каждого экземпляра параметров. Метаданные будут разрешаться в порядке конфигурации, аналогично тому, как [Combine\(IJsonTypeInfoResolver\[\]\)](#) разрешается метаданные.

Применяется к

Продукт	Версии (Устарело)
.NET	6, 7 (8)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.GetConverter(Type) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает преобразователь для указанного типа.

C#

```
public System.Text.Json.Serialization.JsonConverter GetConverter (Type typeToConvert);
```

Параметры

typeToConvert [Type](#)

Тип, для которого возвращается преобразователь.

Возвращаемое значение

[JsonConverter](#)

Первый преобразователь, поддерживающий данный тип.

Исключения

[InvalidOperationException](#)

Объект [JsonConverter](#), настроенный для `typeToConvert`, вернул недопустимый преобразователь.

[NotSupportedException](#)

Совместимые объекты [JsonConverter](#) для `typeToConvert` или его сериализуемых членов отсутствуют.

Комментарии

Дополнительные сведения см. в статье [Создание пользовательских преобразователей для сериализации \(маршалинга\) JSON в .NET](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.GetTypeInfo(Type)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает метаданные [JsonTypeInfo](#) контракта, разрешенные текущим [JsonSerializerOptions](#) экземпляром.

C#

```
public System.Text.Json.Serialization.Metadata.JsonPropertyInfo GetTypeInfo  
(Type type);
```

Параметры

type [Type](#)

Тип для разрешения метаданных контракта.

Возвращаемое значение

[JsonPropertyInfo](#)

Метаданные контракта, разрешенные для **type**.

Исключения

[ArgumentNullException](#)

type имеет значение `null`.

[ArgumentException](#)

type не является допустимым для сериализации.

Комментарии

Возвращаемые метаданные могут быть понижены `JsonTypeInfo<T>` в и использоваться с соответствующими `JsonSerializer` перегрузками.

`JsonSerializerOptions` Если экземпляр заблокирован для изменения, метод возвращает кэшированный экземпляр для метаданных.

Применяется к

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.MakeReadOnly

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

MakeReadOnly()	Помечает текущий экземпляр как доступный только для чтения, чтобы предотвратить дальнейшие изменения пользователей.
MakeReadOnly(Boolean)	Помечает текущий экземпляр как доступный только для чтения, предотвращая дальнейшие изменения пользователей.

MakeReadOnly()

Помечает текущий экземпляр как доступный только для чтения, чтобы предотвратить дальнейшие изменения пользователей.

C#

```
public void MakeReadOnly();
```

Исключения

[InvalidOperationException](#)

Экземпляр не задает [TypeInfoResolver](#) параметр .

Комментарии

Этот метод является идемпотентным.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

MakeReadOnly(Boolean)

Помечает текущий экземпляр как доступный только для чтения, предотвращая дальнейшие изменения пользователей.

C#

```
public void MakeReadOnly (bool populateMissingResolver);
```

Параметры

populateMissingResolver Boolean

Заполняет ненастроенные [TypeInfoResolver](#) свойства значением по умолчанию на основе отражения.

Исключения

[InvalidOperationException](#)

Экземпляр не задает [TypeInfoResolver](#) параметр. Возникает, когда `populateMissingResolver` имеет значение `false`.

-ИЛИ-

Переключатель [IsReflectionEnabledByDefault](#) функций отключен.

Комментарии

Если `populateMissingResolver` для задано значение `true`, экземпляр настраивается в соответствии с семантикой [JsonSerializer](#) методов, [JsonSerializerOptions](#) принимающих параметры.

Этот метод является идемпотентным.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonSerializerOptions.TryGetType Info(Type, JsonTypeInfo) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается получить метаданные [JsonTypeInfo](#) контракта, разрешенные текущим [JsonSerializerOptions](#) экземпляром.

C#

```
public bool TryGetTypeInfo (Type type, out  
System.Text.Json.Serialization.Metadata.JsonTypeInfo? typeInfo);
```

Параметры

type [Type](#)

Тип для разрешения метаданных контракта.

typeInfo [JsonTypeInfo](#)

При возврате этого метода содержит разрешенные метаданные контракта или [null](#) значение , если не удалось разрешить контракт.

Возвращаемое значение

[Boolean](#)

[true](#) Значение , если контракт для [type](#) был найден или [false](#) иным образом.

Исключения

[ArgumentNullException](#)

[type](#) имеет значение [null](#).

[ArgumentException](#)

[type](#) не является допустимым для сериализации.

Комментарии

Возвращаемые метаданные могут быть понижены `JsonTypeInfo<T>` в и использоваться с соответствующими `JsonSerializer` перегрузками.

`JsonSerializerOptions` Если экземпляр заблокирован для изменения, метод возвращает кэшированный экземпляр для метаданных.

Применяется к

Продукт	Версии
.NET	8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonTokenType Enum

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Defines the various JSON tokens that make up a JSON text.

C#

```
public enum JsonTokenType
```

Inheritance Object → [ValueType](#) → [Enum](#) → JsonTokenType

Fields

Comment	6	The token type is a comment string.
EndArray	4	The token type is the end of a JSON array.
EndObject	2	The token type is the end of a JSON object.
False	10	The token type is the JSON literal false .
None	0	There is no value (as distinct from Null). This is the default token type if no data has been read by the Utf8JsonReader .
Null	11	The token type is the JSON literal null .
Number	8	The token type is a JSON number.
PropertyName	5	The token type is a JSON property name.
StartArray	3	The token type is the start of a JSON array.
StartObject	1	The token type is the start of a JSON object.
String	7	The token type is a JSON string.
True	9	The token type is the JSON literal true .

Remarks

For more information, see [How to write custom serializers and deserializers with System.Text.Json](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonValueKind Enum

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Specifies the data type of a JSON value.

C#

```
public enum JsonValueKind
```

Inheritance Object → [ValueType](#) → [Enum](#) → JsonValueKind

Fields

Array	2	A JSON array.
False	6	The JSON value false .
Null	7	The JSON value null .
Number	4	A JSON number.
Object	1	A JSON object.
String	3	A JSON string.
True	5	The JSON value true .
Undefined	0	There is no value (as distinct from Null).

Remarks

For more information, see [How to write custom serializers and deserializers with System.Text.Json](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonWriterOptions Структура

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Разрешает пользователю определить пользовательское поведение при чтении JSON с помощью [Utf8JsonWriter](#).

C#

```
public struct JsonWriterOptions
```

Наследование [Object](#) → [ValueType](#) → [JsonWriterOptions](#)

Комментарии

По умолчанию JSON записывается без отступов или дополнительных пробелов. Кроме того, создает исключение, [Utf8JsonWriter](#) если пользователь пытается записать структурно недопустимый JSON.

Дополнительные сведения см. в статье [Создание пользовательских сериализаторов и десериализаторов с помощью System.Text.Json](#).

Свойства

Encoder	Возвращает или устанавливает кодировщик, используемый при экранировании строк. Укажите значение <code>null</code> для использования кодировщика по умолчанию.
Indented	Возвращает или задает значение, указывающее, следует ли Utf8JsonWriter форматировать выходные данные JSON, включая добавление отступов для вложенных токенов JSON, добавление новых строк и добавление пробела между именами и значениями свойств.
MaxDepth	Возвращает или задает максимальную глубину, разрешенную при записи JSON, при этом значение по умолчанию (то есть 0), указывающее максимальную глубину 1000.

SkipValidation

Возвращает или задает значение, указывающее, следует ли [Utf8JsonWriter](#) пропустить структурную проверку и разрешить пользователю записать недопустимую JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonWriterOptions.Encoder Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets the encoder to use when escaping strings, or `null` to use the default encoder.

C#

```
public System.Text_ENCODINGS.Web.JavaScriptEncoder? Encoder { get; set; }
```

Property Value

[JavaScriptEncoder](#)

The JavaScript character encoder used to override the escaping behavior.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

JsonWriterOptions.Indented Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает значение, указывающее, следует ли [Utf8JsonWriter](#) форматировать выходные данные JSON, включая добавление отступов для вложенных токенов JSON, добавление новых строк и добавление пробела между именами и значениями свойств.

C#

```
public bool Indented { get; set; }
```

Значение свойства

[Boolean](#)

`true` Значение , если выходные данные JSON форматированы; `false` Значение , если JSON записывается без дополнительных пробелов. Значение по умолчанию — `false`.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonWriterOptions.MaxDepth Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает или задает максимальную глубину, разрешенную при записи JSON, при этом значение по умолчанию (то есть 0), указывающее максимальную глубину 1000.

C#

```
public int MaxDepth { get; set; }
```

Значение свойства

[Int32](#)

Исключения

[ArgumentOutOfRangeException](#)

Возникает, если для максимальной глубины задано отрицательное значение.

Комментарии

При чтении за эту глубину <возникнет исключение
cref="T:System.Text.Json.JsonException".>

Применяется к

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

JsonWriterOptions.SkipValidation Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets or sets a value that indicates whether the [Utf8JsonWriter](#) should skip structural validation and allow the user to write invalid JSON.

C#

```
public bool SkipValidation { get; set; }
```

Property Value

[Boolean](#)

`true` if structural validation is skipped and invalid JSON is allowed; `false` if an [InvalidOperationException](#) is thrown on any attempt to write invalid JSON.

Remarks

If the JSON being written is known to be correct, then skipping validation (by setting this property to `true`) could improve performance. An example of invalid JSON where the writer will throw (when `SkipValidation` is set to `false`) is when you write a value within a JSON object without a property name.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader Struct

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Provides a high-performance API for forward-only, read-only access to UTF-8 encoded JSON text.

C#

```
public ref struct Utf8JsonReader
```

Inheritance [Object](#) → [ValueType](#) → [Utf8JsonReader](#)

Remarks

`Utf8JsonReader` processes the text sequentially with no caching and by default adheres strictly to the [JSON RFC](#).

When `Utf8JsonReader` encounters invalid JSON, it throws a [JsonException](#) with basic error information like line number and byte position on the line.

Since this type is a ref struct, it doesn't directly support async. However, it does provide support for reentrancy to read incomplete data and to continue reading once more data is presented.

To be able to set max depth while reading or to allow skipping comments, create an instance of [JsonReaderOptions](#) and pass it to the reader.

For more information, see [Use Utf8JsonReader](#).

Constructors

[Utf8JsonReader\(ReadOnlySequence<Byte>, Boolean, JsonReaderState\)](#)

Initializes a new instance of the [Utf8JsonReader](#) structure that processes a read-only sequence of UTF-8 encoded text and indicates whether the input contains all the text to process.

<code>Utf8JsonReader(ReadOnlySequence<Byte>, JsonReaderOptions)</code>	Initializes a new instance of the Utf8JsonReader structure that processes a read-only sequence of UTF-8 encoded text using the specified options.
<code>Utf8JsonReader(ReadOnlySpan<Byte>, Boolean, JsonReaderState)</code>	Initializes a new instance of the Utf8JsonReader structure that processes a read-only span of UTF-8 encoded text and indicates whether the input contains all the text to process.
<code>Utf8JsonReader(ReadOnlySpan<Byte>, JsonReaderOptions)</code>	Initializes a new instance of the Utf8JsonReader structure that processes a read-only span of UTF-8 encoded text using the specified options.

Properties

<code>BytesConsumed</code>	Gets the total number of bytes consumed so far by this instance of the Utf8JsonReader .
<code>CurrentDepth</code>	Gets the depth of the current token.
<code>CurrentState</code>	Gets the current Utf8JsonReader state to pass to a Utf8JsonReader constructor with more data.
<code>HasValueSequence</code>	Gets a value that indicates which <code>Value</code> property to use to get the token value.
<code>IsFinalBlock</code>	Gets a value that indicates whether all the JSON data was provided or there is more data to come.
<code>Position</code>	Gets the current SequencePosition within the provided UTF-8 encoded input <code>ReadOnlySequence<byte></code> or a default SequencePosition if the Utf8JsonReader struct was constructed with a <code>ReadOnlySpan<byte></code> .
<code>TokenStartIndex</code>	Gets the index that the last processed JSON token starts at (within the given UTF-8 encoded input text), skipping any white space.
<code>TokenType</code>	Gets the type of the last processed JSON token in the UTF-8 encoded JSON text.
<code>ValueIsEscaped</code>	Gets a value that indicates whether the current ValueSpan or ValueSequence properties contain escape sequences per RFC 8259 section 7, and therefore require unescaping before being consumed.
<code>ValueSequence</code>	Gets the raw value of the last processed token as a <code>ReadOnlySequence<byte></code> slice of the input payload, only if the token is contained within multiple segments.

ValueSpan	Gets the raw value of the last processed token as a <code>ReadOnlySpan<byte></code> slice of the input payload, if the token fits in a single segment or if the reader was constructed with a JSON payload contained in a <code>ReadOnlySpan<byte></code> .
---------------------------	---

Methods

CopyString(Span<Byte>)	Copies the current JSON token value from the source, unescaped, as UTF-8 bytes to a buffer.
CopyString(Span<Char>)	Copies the current JSON token value from the source, unescaped, as UTF-16 characters to a buffer.
GetBoolean()	Reads the next JSON token value from the source as a Boolean .
GetByte()	Parses the current JSON token value from the source as a Byte .
GetBytesFromBase64()	Parses the current JSON token value from the source and decodes the Base64 encoded JSON string as a byte array.
GetComment()	Parses the current JSON token value from the source as a comment and transcodes it as a String .
GetDateTime()	Reads the next JSON token value from the source and parses it to a DateTime .
GetDateTimeOffset()	Reads the next JSON token value from the source and parses it to a DateTimeOffset .
GetDecimal()	Reads the next JSON token value from the source and parses it to a Decimal .
GetDouble()	Reads the next JSON token value from the source and parses it to a Double .
GetGuid()	Reads the next JSON token value from the source and parses it to a Guid .
GetInt16()	Parses the current JSON token value from the source as a Int16 .
GetInt32()	Reads the next JSON token value from the source and parses it to an Int32 .
GetInt64()	Reads the next JSON token value from the source and parses it to an Int64 .
GetSByte()	Parses the current JSON token value from the source as an SByte .

GetSingle()	Reads the next JSON token value from the source and parses it to a Single .
GetString()	Reads the next JSON token value from the source unescaped and transcodes it as a string.
GetUInt16()	Parses the current JSON token value from the source as a UInt16 .
GetUInt32()	Reads the next JSON token value from the source and parses it to a UInt32 .
GetUInt64()	Reads the next JSON token value from the source and parses it to a UInt64 .
Read()	Reads the next JSON token from the input source.
Skip()	Skips the children of the current JSON token.
TryGetByte(Byte)	Tries to parse the current JSON token value from the source as a Byte and returns a value that indicates whether the operation succeeded.
TryGetBytesFromBase64(Byte[])	Tries to parse the current JSON token value from the source and decodes the Base64 encoded JSON string as a byte array and returns a value that indicates whether the operation succeeded.
TryGetDateTime(DateTime)	Tries to parse the current JSON token value from the source as a DateTime and returns a value that indicates whether the operation succeeded.
TryGetDateTimeOffset(DateTimeOffset)	Tries to parse the current JSON token value from the source as a DateTimeOffset and returns a value that indicates whether the operation succeeded.
TryGetDecimal(Decimal)	Tries to parse the current JSON token value from the source as a Decimal and returns a value that indicates whether the operation succeeded.
TryGetDouble(Double)	Tries to parse the current JSON token value from the source as a Double and returns a value that indicates whether the operation succeeded.
TryGetGuid(Guid)	Tries to parse the current JSON token value from the source as a Guid and returns a value that indicates whether the operation succeeded.
TryGetInt16(Int16)	Tries to parse the current JSON token value from the source as an Int16 and returns a value that indicates whether the operation succeeded.
TryGetInt32(Int32)	Tries to parse the current JSON token value from the source as an Int32 and returns a value that indicates whether the operation

	succeeded.
TryGetInt64(Int64)	Tries to parse the current JSON token value from the source as an Int64 and returns a value that indicates whether the operation succeeded.
TryGetSByte(SByte)	Tries to parse the current JSON token value from the source as an SByte and returns a value that indicates whether the operation succeeded.
TryGetSingle(Single)	Tries to parse the current JSON token value from the source as a Single and returns a value that indicates whether the operation succeeded.
TryGetUInt16(UInt16)	Tries to parse the current JSON token value from the source as a UInt16 and returns a value that indicates whether the operation succeeded.
TryGetUInt32(UInt32)	Tries to parse the current JSON token value from the source as a UInt32 and returns a value that indicates whether the operation succeeded.
TryGetUInt64(UInt64)	Tries to parse the current JSON token value from the source as a UInt64 and returns a value that indicates whether the operation succeeded.
TrySkip()	Tries to skip the children of the current JSON token.
ValueTextEquals(ReadOnlySpan<Byte>)	Compares the UTF-8 encoded text in a read-only byte span to the unescaped JSON token value in the source and returns a value that indicates whether they match.
ValueTextEquals(ReadOnlySpan<Char>)	Compares the text in a read-only character span to the unescaped JSON token value in the source and returns a value that indicates whether they match.
ValueTextEquals(String)	Compares the string text to the unescaped JSON token value in the source and returns a value that indicates whether they match.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader Constructors

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

Utf8JsonReader(ReadOnlySequence<Byte>, JsonReaderOptions)	Initializes a new instance of the Utf8JsonReader structure that processes a read-only sequence of UTF-8 encoded text using the specified options.
Utf8JsonReader(ReadOnlySpan<Byte>, JsonReaderOptions)	Initializes a new instance of the Utf8JsonReader structure that processes a read-only span of UTF-8 encoded text using the specified options.
Utf8JsonReader(ReadOnlySequence<Byte>, Boolean, JsonReaderState)	Initializes a new instance of the Utf8JsonReader structure that processes a read-only sequence of UTF-8 encoded text and indicates whether the input contains all the text to process.
Utf8JsonReader(ReadOnlySpan<Byte>, Boolean, JsonReaderState)	Initializes a new instance of the Utf8JsonReader structure that processes a read-only span of UTF-8 encoded text and indicates whether the input contains all the text to process.

Utf8JsonReader(ReadOnlySequence<Byte>, JsonReaderOptions)

Initializes a new instance of the [Utf8JsonReader](#) structure that processes a read-only sequence of UTF-8 encoded text using the specified options.

C#

```
public Utf8JsonReader (System.Buffers.ReadOnlySequence<byte> jsonData,
System.Text.Json.JsonReaderOptions options = default);
```

Parameters

jsonData [ReadOnlySequence<Byte>](#)

The UTF-8 encoded JSON text to process.

options [JsonReaderOptions](#)

Options that define customized behavior of the [Utf8JsonReader](#) that differs from the JSON RFC (for example, how to handle comments or maximum depth allowed when reading). By default, the [Utf8JsonReader](#) follows the JSON RFC strictly; comments within the JSON are invalid, and the maximum depth is 64.

Remarks

Since this type is a ref struct, it is a stack-only type, and all the limitations of ref structs apply to it.

This constructor assumes that the entire JSON payload is contained in `jsonData`; it is equivalent to `Utf8JsonReader.IsFinalBlock = true`.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Utf8JsonReader(ReadOnlySpan<Byte>, JsonReaderOptions)

Initializes a new instance of the [Utf8JsonReader](#) structure that processes a read-only span of UTF-8 encoded text using the specified options.

C#

```
public Utf8JsonReader (ReadOnlySpan<byte> jsonData,  
System.Text.Json.JsonReaderOptions options = default);
```

Parameters

jsonData [ReadOnlySpan<Byte>](#)

The UTF-8 encoded JSON text to process.

options [JsonReaderOptions](#)

Options that define customized behavior of the [Utf8JsonReader](#) that differs from the JSON RFC (for example, how to handle comments or maximum depth allowed when reading). By default, the [Utf8JsonReader](#) follows the JSON RFC strictly; comments within the JSON are invalid, and the maximum depth is 64.

Remarks

Since this type is a ref struct, it is a stack-only type, and all the limitations of ref structs apply to it.

This constructor assumes that the entire JSON payload is contained in `jsonData`; it is equivalent to [Utf8JsonReader.IsFinalBlock = true](#).

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Utf8JsonReader(ReadOnlySequence<Byte>, Boolean, JsonReaderState)

Initializes a new instance of the [Utf8JsonReader](#) structure that processes a read-only sequence of UTF-8 encoded text and indicates whether the input contains all the text to process.

C#

```
public Utf8JsonReader (System.Buffers.ReadOnlySequence<byte> jsonData,
bool isFinalBlock, System.Text.Json.JsonReaderState state);
```

Parameters

jsonData [ReadOnlySequence<Byte>](#)

The UTF-8 encoded JSON text to process.

isFinalBlock [Boolean](#)

`true` to indicate that the input sequence contains the entire data to process; `false` to indicate that the input span contains partial data with more data to follow.

state [JsonReaderState](#)

The reader state. If this is the first call to the constructor, pass the default state; otherwise, pass the value of the [CurrentState](#) property from the previous instance of the [Utf8JsonReader](#).

Remarks

Since this type is a ref struct, it is a stack-only type, and all the limitations of ref structs apply to it. This is the reason why the constructor accepts a [JsonReaderState](#).

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Utf8JsonReader(ReadOnlySpan<Byte>, Boolean, JsonReaderState)

Initializes a new instance of the [Utf8JsonReader](#) structure that processes a read-only span of UTF-8 encoded text and indicates whether the input contains all the text to process.

C#

```
public Utf8JsonReader (ReadOnlySpan<byte> jsonData, bool isFinalBlock,  
System.Text.Json.JsonReaderState state);
```

Parameters

jsonData [ReadOnlySpan<Byte>](#)

The UTF-8 encoded JSON text to process.

isFinalBlock [Boolean](#)

`true` to indicate that the input sequence contains the entire data to process; `false` to indicate that the input span contains partial data with more data to follow.

state [JsonReaderState](#)

The reader state. If this is the first call to the constructor, pass the default state; otherwise, pass the value of the [CurrentState](#) property from the previous instance of the [Utf8JsonReader](#).

Remarks

Since this type is a ref struct, it is a stack-only type, and all the limitations of ref structs apply to it. This is the reason why the constructor accepts a [JsonReaderState](#).

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.BytesConsumed Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the total number of bytes consumed so far by this instance of the [Utf8JsonReader](#).

C#

```
public long BytesConsumed { get; }
```

Property Value

[Int64](#)

The total number of bytes consumed so far.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.CurrentDepth Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает глубину текущего токена.

C#

```
public int CurrentDepth { get; }
```

Значение свойства

[Int32](#)

Глубина текущего маркера.

Комментарии

Свойство `CurrentDepth` отслеживает рекурсивную глубину вложенных объектов или массивов в тексте JSON, обработанном до сих пор.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.CurrentState Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the current [Utf8JsonReader](#) state to pass to a [Utf8JsonReader](#) constructor with more data.

C#

```
public System.Text.Json.JsonReaderState CurrentState { get; }
```

Property Value

[JsonReaderState](#)

The current reader state.

Remarks

Unlike the [Utf8JsonReader](#), which is a ref struct, the state can survive across `async / await` boundaries. This type is required to provide support for reading in more data asynchronously before continuing with a new instance of the [Utf8JsonReader](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.HasValueSequence Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets a value that indicates which `value` property to use to get the token value.

C#

```
public bool HasValueSequence { get; }
```

Property Value

[Boolean](#)

`true` if [ValueSequence](#) should be used to get the token value; `false` if [ValueSpan](#) should be used instead.

Remarks

If `HasValueSequence` is `false`, [ValueSequence](#) is empty. Therefore, read the token value using the [ValueSpan](#) property.

For input data within a `ReadOnlySpan<byte>`, this always returns `false`. For input data within a `ReadOnlySequence<byte>`, this only returns `true` if the token value straddles more than a single segment and hence can't be represented as a span.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.IsFinalBlock Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение, указывающее, были ли предоставлены все данные JSON или есть дополнительные данные.

C#

```
public bool IsFinalBlock { get; }
```

Значение свойства

[Boolean](#)

`true` Значение , если средство чтения было создано с входным диапазоном или последовательностью, содержащей все данные JSON для обработки; `false`

Значение , если средство чтения было создано с диапазоном входных данных или последовательностью, которая может содержать частичные данные JSON с дополнительными данными для выполнения.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезны?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.Position Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает текущий [SequencePosition](#) в рамках предоставленных входных данных `ReadOnlySequence<byte>` в кодировке UTF-8 или значение по умолчанию [SequencePosition](#), если структура [Utf8JsonReader](#) была создана с использованием `ReadOnlySpan<byte>`.

C#

```
public SequencePosition Position { get; }
```

Значение свойства

[SequencePosition](#)

Текущий [SequencePosition](#) байт> в предоставленной входной кодировке UTF-8 `ReadOnlySequence<или значение по умолчаниюSequencePosition`, если [Utf8JsonReader](#) структура была создана с использованием байта> `ReadOnlySpan<`.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TokenStartIndex

Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the index that the last processed JSON token starts at (within the given UTF-8 encoded input text), skipping any white space.

C#

```
public long TokenstartIndex { get; }
```

Property Value

[Int64](#)

The starting index of the last processed JSON token within the given UTF-8 encoded input text.

Remarks

For JSON strings (including property names), this value points to before the start quote.

For comments, this value points to before the first comment delimiter (that is, '/'). This is only applicable when the reader is constructed using the [JsonCommentHandling.Allow](#) option.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TokenType Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the type of the last processed JSON token in the UTF-8 encoded JSON text.

C#

```
public System.Text.Json.JsonTokenType TokenType { get; }
```

Property Value

[JsonTokenType](#)

The type of the last processed JSON token.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.ValueIsEscaped

Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает значение, указывающее, содержит ли текущие [ValueSpan](#) или [ValueSequence](#) свойства escape-последовательности в соответствии с разделом 7 RFC 8259 и, следовательно, требуют отмены экранирования перед использованием.

C#

```
public bool ValueIsEscaped { get; }
```

Значение свойства

[Boolean](#)

Применяется к

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.ValueSequence Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the raw value of the last processed token as a `ReadOnlySequence<byte>` slice of the input payload, only if the token is contained within multiple segments.

C#

```
public System.Buffers.ReadOnlySequence<byte> ValueSequence { get; }
```

Property Value

[ReadOnlySequence<Byte>](#)

A byte read-only sequence.

Remarks

If the JSON is provided within a `ReadOnlySequence<byte>` and the slice that represents the token value fits in a single segment, then [ValueSpan](#) contains the sliced value since it can be represented as a span. Otherwise, `ValueSequence` contains the token value.

If [HasValueSequence](#) is `false`, `ValueSequence` is empty. Therefore, only access `ValueSequence` if [HasValueSequence](#) is `true`. Otherwise, the token value must be accessed from [ValueSpan](#).

There is no guarantee that this property will always contain well-formed data. If the input JSON passed in to the `Utf8JsonReader` contains invalid UTF-8 bytes within JSON string tokens, this property will return back those invalid UTF-8 bytes as is. Therefore, if the input is untrusted or not previously validated, call [GetString\(\)](#) to get the JSON string token.

Since this property returns the raw bytes, avoid using it for text comparison. Instead call [ValueTextEquals](#), which unescapes the text if necessary.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.ValueSpan Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the raw value of the last processed token as a `ReadOnlySpan<byte>` slice of the input payload, if the token fits in a single segment or if the reader was constructed with a JSON payload contained in a `ReadOnlySpan<byte>`.

C#

```
public ReadOnlySpan<byte> ValueSpan { get; }
```

Property Value

[ReadOnlySpan<Byte>](#)

A read-only span of bytes.

Remarks

If the JSON is provided within a `ReadOnlySequence<byte>` and the slice that represents the token value fits in a single segment, then `ValueSpan` contains the sliced value since it can be represented as a span. Otherwise, `ValueSequence` contains the token value.

If `HasValueSequence` is `true`, `ValueSpan` is empty. Therefore, only access `ValueSpan` if `HasValueSequence` is `false`. Otherwise, the token value must be accessed from `ValueSequence`.

There is no guarantee that this property will always contain well-formed data. If the input JSON passed in to the `Utf8JsonReader` contains invalid UTF-8 bytes within JSON string tokens, this property will return back those invalid UTF-8 bytes as is. Therefore, if the input is untrusted or not previously validated, call `GetString()` to get the JSON string token.

Since this property returns the raw bytes, avoid using it for text comparison. Instead call `ValueTextEquals` which unescapes the text if necessary.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.CopyString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

CopyString(Span<Byte>)	Копирует текущее значение токена JSON из исходного файла (неэкранированного) в виде UTF-8 байтов в буфер.
CopyString(Span<Char>)	Копирует текущее значение токена JSON из исходного , неэкранированного, в виде символов UTF-16 в буфер.

CopyString(Span<Byte>)

Копирует текущее значение токена JSON из исходного файла (неэкранированного) в виде UTF-8 байтов в буфер.

C#

```
public readonly int CopyString (Span<byte> utf8Destination);
```

Параметры

utf8Destination [Span<Byte>](#)

Буфер для записи неэкранированных байтов UTF-8.

Возвращаемое значение

[Int32](#)

Число байтов, записанных в `utf8Destination`.

Исключения

[InvalidOperationException](#)

Токен JSON не является строкой, то есть не [String](#) является или [PropertyName](#).

-или-

Строка JSON содержит недопустимые байты UTF-8 либо недопустимые суррогаты UTF-16.

[ArgumentException](#)

Целевой буфер слишком мал, чтобы вместить неэкранированное значение.

Комментарии

В отличие от [GetString\(\)](#), этот метод не поддерживает [Null](#).

Этот метод вызывает исключение , [ArgumentException](#) если целевой буфер слишком мал для хранения неэкранированного значения. Вы можете определить буфер соответствующего размера, проконсультировавшись с длиной [ValueSpan](#) или [ValueSequence](#), так как неэкранированный результат всегда меньше или равен длине закодированных строк.

См. также раздел

- [TokenType](#)

Применяется к

▼ .NET 8 и .NET 7

Продукт	Версии
.NET	7, 8

CopyString(Span<Char>)

Копирует текущее значение токена JSON из исходного , неэкранированного, в виде символов UTF-16 в буфер.

C#

```
public readonly int CopyString (Span<char> destination);
```

Параметры

destination [Span<Char>](#)

Буфер для записи перекодированных символов UTF-16.

Возвращаемое значение

[Int32](#)

Число символов, записанных в `destination`.

Исключения

[InvalidOperationException](#)

Токен JSON не является строкой, то есть не `String` является или `PropertyName`.

-или-

Строка JSON содержит недопустимые байты UTF-8 либо недопустимые суррогаты UTF-16.

[ArgumentException](#)

Целевой буфер слишком мал, чтобы вместить неэкранированное значение.

Комментарии

В отличие от `GetString()`, этот метод не поддерживает `Null`.

Этот метод вызывает исключение , [ArgumentException](#) если целевой буфер слишком мал для хранения неэкранированного значения. Вы можете определить буфер соответствующего размера, проконсультировавшись с длиной `ValueSpan` или `ValueSequence`, так как неэкранированный результат всегда меньше или равен длине закодированных строк.

См. также раздел

- [TokenType](#)

Применяется к

▼ .NET 8 и .NET 7

Продукт	Версии
.NET	7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetBoolean Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Reads the next JSON token value from the source as a [Boolean](#).

C#

```
public bool GetBoolean();
```

Returns

[Boolean](#)

`true` if the [TokenType](#) is [True](#); `false` if the [TokenType](#) is [False](#).

Exceptions

[InvalidOperationException](#)

The value of the JSON token isn't a Boolean value (that is, [True](#) or [False](#)).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

Utf8JsonReader.GetByte Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует текущее значение токена JSON из источника в виде [Byte](#).

C#

```
public byte GetByte();
```

Возвращаемое значение

[Byte](#)

Значение токена в кодировке UTF-8.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Неверный числовой формат значения токена JSON (например, он содержит дробное значение или записывается в экспоненциальном представлении).

-или-

Значение токена JSON представляет число меньше [Byte.MinValue](#) или больше [Byte.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetBytesFromBase64

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует текущее значение токена JSON из источника и декодирует строку JSON в кодировке Base64 в виде массива байтов.

C#

```
public byte[] GetBytesFromBase64();
```

Возвращаемое значение

[Byte\[\]](#)

Массив байтов, представляющий текущее значение токена JSON.

Исключения

[InvalidOperationException](#)

Тип токена JSON не является [String](#).

[FormatException](#)

Значение не закодировано в тексте Base64 и поэтому не может быть декодировано в байты.

-или-

Значение содержит недопустимый символ или более двух символов заполнения.

-или-

Значение является неполным. То есть длина строки JSON не кратна 4.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetComment Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Parses the current JSON token value from the source as a comment and transcodes it as a [String](#).

C#

```
public string GetComment();
```

Returns

[String](#)

The comment that represents the current JSON token value.

Exceptions

[InvalidOperationException](#)

The JSON token is not a comment.

Remarks

This is only applicable when the reader is constructed using the [Allow](#) option. Otherwise, the [TokenType](#) will never be [Comment](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- TokenType
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.GetDateTime Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [DateTime](#).

C#

```
public DateTime GetDateTime();
```

Возвращаемое значение

[DateTime](#)

Значение даты и времени, если все значение токена в кодировке UTF-8 можно успешно проанализировать.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [String](#).

[FormatException](#)

Значение токена JSON не может быть прочитано как [DateTime](#).

-ИЛИ-

Все значения токенов в кодировке UTF-8 невозможно проанализировать до значения [DateTime](#).

-ИЛИ-

Значение токена JSON имеет неподдерживаемый формат.

Комментарии

Этот метод создает [DateTime](#) только представление строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetDateTimeOffset Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Reads the next JSON token value from the source and parses it to a [DateTimeOffset](#).

C#

```
public DateTimeOffset GetDateTimeOffset();
```

Returns

[DateTimeOffset](#)

The date and time offset, if the entire UTF-8 encoded token value can be successfully parsed.

Exceptions

[InvalidOperationException](#)

The value of the JSON token isn't a [String](#).

[FormatException](#)

The JSON token value cannot be read as a [DateTimeOffset](#).

-or-

The entire UTF-8 encoded token value cannot be parsed to a [DateTimeOffset](#) value.

-or-

The JSON token value is of an unsupported format.

Remarks

This method only creates a `DateTimeOffset` representation of JSON strings that conform to the ISO 8601-1 extended format (see [DateTime and DateTimeOffset support in System.Text.Json](#)).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.GetDecimal Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Decimal](#).

C#

```
public decimal GetDecimal();
```

Возвращаемое значение

[Decimal](#)

Значение токена в кодировке UTF-8, преобразуемое в [Decimal](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON представляет собой число меньше [Decimal.MinValue](#) или больше [Decimal.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetDouble Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Double](#).

C#

```
public double GetDouble();
```

Возвращаемое значение

[Double](#)

Значение токена в кодировке UTF-8, преобразуемое в [Double](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON представляет собой число, [меньшее Double.MinValue](#) или [больше Double.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetGuid Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Guid](#).

C#

```
public Guid GetGuid();
```

Возвращаемое значение

[Guid](#)

Значение GUID, если все значение токена в кодировке UTF-8 можно успешно проанализировать.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [String](#).

[FormatException](#)

Значение токена JSON имеет неподдерживаемый формат для GUID.

-ИЛИ-

Все значения токенов в кодировке UTF-8 невозможно проанализировать до значения [Guid](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt16 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Анализирует текущее значение токена JSON из источника в виде [Int16](#).

C#

```
public short GetInt16();
```

Возвращаемое значение

[Int16](#)

Значение токена в кодировке UTF-8, преобразуемое в [Int16](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Неверный числовой формат значения токена JSON (например, он содержит дробное значение или записывается в экспоненциальном представлении).

-или-

Значение токена JSON представляет собой число меньше, чем [Int16.MinValue](#) или больше [Int16.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt32 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Int32](#).

C#

```
public int GetInt32 ();
```

Возвращаемое значение

[Int32](#)

Значение токена в кодировке UTF-8, преобразуемое в [Int32](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON имеет неправильный числовой формат. Например, оно содержит десятичную часть или записано в экспоненциальном представлении.

-или-

Значение токена JSON представляет число меньше [Int32.MinValue](#) или больше [Int32.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt64 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Int64](#).

C#

```
public long GetInt64 ();
```

Возвращаемое значение

[Int64](#)

Значение токена в кодировке UTF-8, преобразуемое в [Int64](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON имеет неправильный числовой формат. Например, оно содержит десятичную часть или записано в экспоненциальном представлении.

-или-

Значение токена JSON представляет число меньше , чем [Int64.MinValue](#) или больше [Int64.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetSByte Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Анализирует текущее значение токена JSON из источника в виде [SByte](#).

C#

```
[System.CLSCompliant(false)]
public sbyte GetSByte();
```

Возвращаемое значение

[SByte](#)

Значение токена в кодировке UTF-8, преобразуемое в [SByte](#).

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Неверный числовой формат значения токена JSON (например, он содержит дробное значение или записывается в экспоненциальном представлении).

-или-

Значение токена JSON представляет собой число меньше [SByte.MinValue](#) или больше [SByte.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetSingle Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из источника и преобразует его в [Single](#).

C#

```
public float GetSingle();
```

Возвращаемое значение

[Single](#)

Значение токена в кодировке UTF-8, преобразуемое в [Single](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON представляет собой число меньше, чем [Single.MinValue](#) или больше [Single.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Считывает следующее значение токена JSON из неэкранированного источника и перекодирует его как строку.

C#

```
public string? GetString();
```

Возвращаемое значение

[String](#)

Значение токена, преобразуемое в строку, или `null`, если [TokenType](#) имеет значение [Null](#).

Исключения

[InvalidOperationException](#)

Значение токена JSON не является строкой (то есть [String](#), [PropertyName](#) или [Null](#)).

-или-

Строка JSON содержит недопустимые байты UTF-8 либо недопустимые суррогаты UTF-16.

Комментарии

Возвращает `null` если [TokenType](#) имеет значение [JsonTokenType.Null](#).

Если вы используете .NET 7 или более позднюю версию и производительность является проблемой, рассмотрите [Utf8JsonReader.CopyString](#) возможность использования метода `.CopyString` позволяет избежать выделения новой строки при каждом вызове метода.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt16 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Анализирует текущее значение токена JSON из источника в виде [UInt16](#).

C#

```
[System.CLSCompliant(false)]
public ushort GetUInt16();
```

Возвращаемое значение

[UInt16](#)

Значение токена в кодировке UTF-8, преобразуемое в [UInt16](#).

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Неверный числовой формат значения токена JSON (например, он содержит дробное значение или записывается в экспоненциальном представлении).

-или-

Значение токена JSON представляет число меньше [UInt16.MinValue](#) или больше [UInt16.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt32 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Считывает следующее значение токена JSON из источника и преобразует его в [UInt32](#).

C#

```
[System.CLSCompliant(false)]
public uint GetUInt32();
```

Возвращаемое значение

[UInt32](#)

Значение токена в кодировке UTF-8, преобразуемое в [UInt32](#).

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON имеет неправильный числовой формат. Например, оно содержит десятичную часть или записано в экспоненциальном представлении.

-ИЛИ-

Значение токена JSON представляет собой число меньше [UInt32.MinValue](#) или больше [UInt32.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.GetInt64 Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Считывает следующее значение токена JSON из источника и преобразует его в [UInt64](#).

C#

```
[System.CLSCompliant(false)]
public ulong GetUInt64();
```

Возвращаемое значение

[UInt64](#)

Значение токена в кодировке UTF-8, преобразуемое в [UInt64](#).

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

[FormatException](#)

Значение токена JSON имеет неправильный числовой формат. Например, оно содержит десятичную часть или записано в экспоненциальном представлении.

-ИЛИ-

Значение токена JSON представляет число меньше [UInt64.MinValue](#) или больше [UInt64.MaxValue](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.Read Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Reads the next JSON token from the input source.

C#

```
public bool Read();
```

Returns

[Boolean](#)

`true` if the token was read successfully; otherwise, `false`.

Exceptions

[JsonException](#)

An invalid JSON token according to the JSON RFC is encountered.

-or-

The current depth exceeds the recursive limit set by the maximum depth.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

Utf8JsonReader.Skip Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Skips the children of the current JSON token.

C#

```
public void Skip();
```

Exceptions

[InvalidOperationException](#)

The reader was given partial data with more data to follow (that is, `IsFinalBlock` is `false`).

[JsonException](#)

An invalid JSON token was encountered while skipping, according to the JSON RFC.

-or-

The current depth exceeds the recursive limit set by the maximum depth.

Remarks

When `TokenType` is `JsonTokenType.PropertyName`, the reader first moves to the property value.

When `TokenType` (originally, or after advancing) is `StartObject` or `JsonTokenType.StartArray`, the reader advances to the matching `JsonTokenType.EndObject` or `JsonTokenType.EndArray`.

For all other token types, the reader does not move. After the next call to `Read()`, the reader will be at the next value (when in an array), the next property name (when in an object), or the end array/object token.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetByte(Byte) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Tries to parse the current JSON token value from the source as a [Byte](#) and returns a value that indicates whether the operation succeeded.

C#

```
public bool TryGetByte (out byte value);
```

Parameters

value [Byte](#)

When this method returns, contains the byte equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [Byte](#) value; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetBytesFromBase64(Byte[]) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Tries to parse the current JSON token value from the source and decodes the Base64 encoded JSON string as a byte array and returns a value that indicates whether the operation succeeded.

C#

```
public bool TryGetBytesFromBase64 (out byte[]? value);
```

Parameters

value [Byte\[\]](#)

When this method returns, contains the decoded binary representation of the Base64 text.

Returns

[Boolean](#)

`true` if the entire token value is encoded as valid Base64 text and can be successfully decoded to bytes; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

The JSON token is not a [String](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetDateTime Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [DateTime](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetDateTime (out DateTime value);
```

Параметры

value [DateTime](#)

При возврате этим методом содержит значение даты и времени, эквивалентное текущей строке JSON, если преобразование выполнено успешно или [MinValue](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [DateTime](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [String](#).

Комментарии

Этот метод создает [DateTime](#) только представление строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetDateTimeOffset(DateTimeOffset) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [DateTimeOffset](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetDateTimeOffset (out DateTimeOffset value);
```

Параметры

value [DateTimeOffset](#)

При возврате этим методом содержит значение даты и времени, эквивалентное текущей строке JSON, если преобразование выполнено успешно или [MinValue](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [DateTimeOffset](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [String](#).

Комментарии

Этот метод создает [DateTimeOffset](#) только представление строк JSON, соответствующих расширенному формату ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetDecimal(Decimal) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Tries to parse the current JSON token value from the source as a [Decimal](#) and returns a value that indicates whether the operation succeeded.

C#

```
public bool TryGetDecimal (out decimal value);
```

Parameters

value [Decimal](#)

When this method returns, contains the decimal equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [Decimal](#) value; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetDouble(Double)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [Double](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetDouble (out double value);
```

Параметры

value [Double](#)

При возврате этим методом содержит значение двойной точности с плавающей запятой, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [Double](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetGuid(Guid)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [Guid](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetGuid (out Guid value);
```

Параметры

value [Guid](#)

При возврате этим методом содержит ИДЕНТИФИКАТОР GUID, эквивалентный текущей строке JSON, если преобразование выполнено успешно или [Empty](#) если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [Guid](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [String](#).

Комментарии

Этот метод анализирует [Guid](#) только значения с дефисами и без окружающих украшений.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetInt16(Int16) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Tries to parse the current JSON token value from the source as an [Int16](#) and returns a value that indicates whether the operation succeeded.

C#

```
public bool TryGetInt16 (out short value);
```

Parameters

value [Int16](#)

When this method returns, contains the 16-bit integer value equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [Int16](#) value; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetInt32(Int32) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Tries to parse the current JSON token value from the source as an [Int32](#) and returns a value that indicates whether the operation succeeded.

C#

```
public bool TryGetInt32 (out int value);
```

Parameters

value [Int32](#)

When this method returns, contains the 32-bit integer value equivalent to the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to an [Int32](#) value; otherwise, `false`.

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetInt64(Int64)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [Int64](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetInt64 (out long value);
```

Параметры

value [Int64](#)

При возврате этого метода содержит 64-разрядное целое число, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось неудачно.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [Int64](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetSByte(SByte)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Важно!

Этот API несовместим с CLS.

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [SByte](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
[System.CLSCompliant(false)]
public bool TryGetSByte (out sbyte value);
```

Параметры

value [SByte](#)

При возврате этим методом содержит эквивалент со знаком байт текущего числа JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [SByte](#); в противном случае — значение `false`.

Атрибуты [CLSCompliantAttribute](#)

Исключения

InvalidOperationException

Значение токена JSON не является [Number](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetSingle(Single)

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается выполнить синтаксический анализ текущего значения токена JSON из источника как [Single](#) и возвращает значение, указывающее, завершилась ли операция.

C#

```
public bool TryGetSingle (out float value);
```

Параметры

value [Single](#)

При возврате этим методом содержит значение с плавающей запятой с одной точностью, эквивалентное текущему номеру JSON, если преобразование выполнено успешно, или 0, если преобразование завершилось сбоем.

Возвращаемое значение

[Boolean](#)

Значение `true`, если все значение токена в кодировке UTF-8 можно успешно преобразовать в значение [Single](#); в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Значение токена JSON не является [Number](#).

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

См. также раздел

- [TokenType](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.TryGetUInt16(UInt16) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Tries to parse the current JSON token value from the source as a [UInt16](#) and returns a value that indicates whether the operation succeeded.

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt16 (out ushort value);
```

Parameters

value [UInt16](#)

When this method returns, contains the unsigned 16-bit integer value equivalent of the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [UInt16](#) value; otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetUInt32(UInt32) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Tries to parse the current JSON token value from the source as a [UInt32](#) and returns a value that indicates whether the operation succeeded.

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt32 (out uint value);
```

Parameters

value [UInt32](#)

When this method returns, contains unsigned 32-bit integer value equivalent to the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [UInt32](#) value; otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TryGetUInt64(UInt64) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Important

This API is not CLS-compliant.

Tries to parse the current JSON token value from the source as a [UInt64](#) and returns a value that indicates whether the operation succeeded.

C#

```
[System.CLSCompliant(false)]
public bool TryGetUInt64 (out ulong value);
```

Parameters

value [UInt64](#)

When this method returns, contains unsigned 64-bit integer value equivalent to the current JSON number if the conversion succeeded, or 0 if the conversion failed.

Returns

[Boolean](#)

`true` if the entire UTF-8 encoded token value can be successfully parsed to a [UInt64](#) value; otherwise, `false`.

Attributes [CLSCompliantAttribute](#)

Exceptions

[InvalidOperationException](#)

The JSON token value isn't a [Number](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

See also

- [TokenType](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonReader.TrySkip Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Пытается пропустить дочерние узлы текущего токена JSON.

C#

```
public bool TrySkip();
```

Возвращаемое значение

[Boolean](#)

`true` при наличии достаточного количества данных для успешного пропуска дочерних элементов; в противном случае — `false`.

Исключения

[JsonException](#)

При пропуске обнаружен токен JSON, не являющийся допустимым согласно RFC по JSON.

-или-

Текущая глубина превышает рекурсивный предел, заданный максимальной глубиной.

Комментарии

Если у средства чтения недостаточно данных, чтобы полностью пропустить дочерние элементы текущего маркера, оно будет сброшено в состояние, в которое оно находилось до вызова метода.

Если `TokenType` имеет значение [JsonTokenType.PropertyName](#), средство чтения сначала переходит к значению свойства .

Если `TokenType` (первоначально или после продвижения) имеет значение `JsonTokenType.StartObject` или `JsonTokenType.StartArray`, средство чтения переходит к сопоставлению `JsonTokenType.EndObject` или `JsonTokenType.EndArray`.

Для всех остальных типов маркеров средство чтения не перемещается. После следующего вызова `Read()` средство чтения будет иметь следующее значение (в массиве), имя следующего свойства (в объекте) или конечный маркер массива или объекта.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonReader.ValueTextEquals Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

ValueTextEquals(ReadOnlySpan<Byte>)	Сравнивает текст в кодировке UTF-8 в диапазоне байтов только для чтения с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.
ValueTextEquals(ReadOnlySpan<Char>)	Сравнивает текст в диапазоне символов только для чтения с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.
ValueTextEquals(String)	Сравнивает текст строки с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.

ValueTextEquals(ReadOnlySpan<Byte>)

Сравнивает текст в кодировке UTF-8 в диапазоне байтов только для чтения с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.

C#

```
public readonly bool ValueTextEquals (ReadOnlySpan<byte> utf8Text);
```

Параметры

utf8Text [ReadOnlySpan<Byte>](#)

Текст в кодировке UTF-8, с которым производится сравнение.

Возвращаемое значение

Boolean

Значение `true`, если значение токена JSON в источнике совпадает с текстом поиска в кодировке UTF-8, в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Токен JSON не является строкой JSON (т. е. не [String](#) или [PropertyName](#)).

Комментарии

Текст подстановки должен быть допустимым текстом UTF-8. В противном случае этот метод может возвращать значение `true`, если источник имеет токен строки, содержащий недопустимый текст UTF-8, который соответствует.

Сравнение значения токена JSON в источнике и текста подстановки выполняется путем предварительного отсения значения JSON в источнике, если это необходимо. Текст подстановки сопоставляется как есть без каких-либо изменений.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

ValueTextEquals(ReadOnlySpan<Char>)

Сравнивает текст в диапазоне символов только для чтения с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.

C#

```
public readonly bool ValueTextEquals (ReadOnlySpan<char> text);
```

Параметры

text `ReadOnlySpan<Char>`

Текст, с которым выполняется сравнение.

Возвращаемое значение

`Boolean`

Значение `true`, если значение токена JSON в источнике совпадает с текстом поиска, в противном случае — значение `false`.

Исключения

`InvalidOperationException`

Токен JSON не является строкой JSON (т. е. не `String` или `PropertyName`).

Комментарии

Если текст подстановки является недопустимым или неполным текстом UTF-16 (т. е. непарными суррогатами), метод возвращает значение `false`, так как в полезных данных JSON нельзя использовать недопустимый UTF-16.

Сравнение значения токена JSON в источнике и текста подстановки выполняется путем предварительного отсояния значения JSON в источнике, если это необходимо. Текст подстановки сопоставляется как есть без каких-либо изменений.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

ValueTextEquals(String)

Сравнивает текст строки с неэкранированным значением токена JSON в источнике и возвращает значение, указывающее, совпадают ли они.

C#

```
public readonly bool ValueTextEquals (string? text);
```

Параметры

text [String](#)

Текст, с которым выполняется сравнение.

Возвращаемое значение

[Boolean](#)

Значение `true`, если значение токена JSON в источнике совпадает с текстом поиска, в противном случае — значение `false`.

Исключения

[InvalidOperationException](#)

Токен JSON не является строкой JSON (т. е. не [String](#) или [PropertyName](#)).

Комментарии

Если текст подстановки является недопустимым или неполным текстом UTF-16 (т. е. непарными суррогатами), метод возвращает значение `false`, так как в полезных данных JSON нельзя использовать недопустимый UTF-16.

Сравнение значения токена JSON в источнике и текста подстановки выполняется путем предварительного отсояния значения JSON в источнике, если это необходимо. Текст подстановки сопоставляется как есть без каких-либо изменений.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter Класс

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Предоставляет высокопроизводительный API для односторонней некэшированной записи текста JSON в кодировке UTF-8.

C#

```
public sealed class Utf8JsonWriter : IAsyncDisposable, IDisposable
```

Наследование [Object](#) → Utf8JsonWriter

Реализации [IAsyncDisposable](#), [IDisposable](#)

Комментарии

`Utf8JsonWriter` записывает текст последовательно без кэширования и по умолчанию соответствует [JSON RFC](#), за исключением написания комментариев.

Метод, который пытается записать недопустимый JSON при включенной проверке [InvalidOperationException](#), вызывает исключение с сообщением об ошибке, зависящим от контекста.

Чтобы иметь возможность отформатировать выходные данные с отступами и пробелами, пропустить проверку или настроить поведение экранирования, создайте экземпляр [JsonWriterOptions](#) и передайте его в модуль записи.

Дополнительные сведения см. в статье [Создание пользовательских сериализаторов и десериализаторов с помощью System.Text.Json](#).

Конструкторы

`Utf8JsonWriter(IBufferWriter<Byte>, JsonWriter)`

Инициализирует новый экземпляр класса [Utf8JsonWriter](#), используя указанный [IBufferWriter<T>](#) для записи в него

Options)	выходных данных и параметров настройки.
Utf8JsonWriter(Stream, JsonWriterOptions)	Инициализирует новый экземпляр класса Utf8JsonWriter , используя указанный поток для записи в него выходных данных и параметров настройки.

Свойства

BytesCommitted	Возвращает общее число байтов, зафиксированных в выходных данных к настоящему времени текущим экземпляром.
BytesPending	Возвращает число байтов, записанных к настоящему времени объектом Utf8JsonWriter и еще не зафиксированных в выходных данных.
CurrentDepth	Возвращает глубину текущего токена.
Options	Возвращает пользовательский режим записи JSON с использованием этого экземпляра, который определяет, необходимо ли форматировать выходные данные при записи, следует ли пропускать структурную проверку JSON и какие символы экранировановать.

Методы

Dispose()	Фиксирует весь оставшийся текст JSON, который еще не был записан, и освобождает все ресурсы, используемые текущим экземпляром.
DisposeAsync()	Асинхронно фиксирует весь оставшийся текст JSON, который еще не был записан, и освобождает все ресурсы, используемые текущим экземпляром.
Equals(Object)	Определяет, равен ли указанный объект текущему объекту. (Унаследовано от Object)
Flush()	Фиксирует текст JSON, записанный на данный момент, что делает его доступным для места назначения вывода.
FlushAsync(CancellationToken)	Асинхронно фиксирует текст JSON, записанный на данный момент, что делает его доступным для места назначения вывода.
GetHashCode()	Служит хэш-функцией по умолчанию. (Унаследовано от Object)

GetType()	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object)
MemberwiseClone()	Создает неполную копию текущего объекта Object . (Унаследовано от Object)
Reset()	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно.
Reset(IBufferWriter<Byte>)	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром IBufferWriter<T> .
Reset(Stream)	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром Stream .
ToString()	Возвращает строку, представляющую текущий объект. (Унаследовано от Object)
WriteBase64String(JsonEncodedText, ReadOnlySpan<Byte>)	Записывает заранее закодированное имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(ReadOnlySpan<Char>, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(String, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64StringValue(ReadOnlySpan<Byte>)	Записывает необработанные байты (в виде строки JSON в кодировке Base64) в качестве элемента массива JSON.
WriteBoolean(JsonEncodedText, Boolean)	Записывает заранее закодированное имя свойства и значение Boolean (в виде литерала JSON <code>true</code> или <code>false</code>) в составе пары "имя-значение" объекта JSON.
WriteBoolean(ReadOnlySpan<Byte>, Boolean)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Boolean (в виде литерала JSON <code>true</code> или <code>false</code>) в составе пары "имя-значение" объекта JSON.
WriteBoolean(ReadOnlySpan<Char>, Boolean)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Boolean (в виде

	литерала JSON <code>true</code> или <code>false</code>) в составе пары "имя-значение" объекта JSON.
WriteBoolean(String, Boolean)	Записывает имя свойства, указываемое как строка, и значение <code>Boolean</code> (в виде литерала JSON <code>true</code> или <code>false</code>) в составе пары "имя-значение" объекта JSON.
WriteBooleanValue(Boolean)	Записывает значение <code>Boolean</code> (в виде литерала JSON <code>true</code> или <code>false</code>) в качестве элемента массива JSON.
WriteCommentValue(ReadOnlySpan<Byte>)	Записывает текстовое значение UTF-8 в виде комментария JSON.
WriteCommentValue(ReadOnlySpan<Char>)	Записывает текстовое значение UTF-16 в виде комментария JSON.
WriteCommentValue(String)	Записывает строковое текстовое значение в виде комментария JSON.
WriteEndArray()	Записывает конец массива JSON.
WriteEndObject()	Записывает конец объекта JSON.
WriteNull(JsonEncodedText)	Записывает заранее закодированное имя свойства и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(ReadOnlySpan<Byte>)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(ReadOnlySpan<Char>)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(String)	Записывает имя свойства, указываемое как строка, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNullValue()	Записывает литерал JSON <code>null</code> в качестве элемента массива JSON.
WriteNumber(JsonEncodedText, Decimal)	Записывает заранее закодированное имя свойства и значение <code>Decimal</code> (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Double)	Записывает заранее закодированное имя свойства и значение <code>Double</code> (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Int32)	Записывает заранее закодированное имя свойства и значение <code>Int32</code> (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

WriteNumber(JsonEncodedText, Int64)	Записывает заранее закодированное имя свойства и значение Int64 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Single)	Записывает заранее закодированное имя свойства и значение Single (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, UInt32)	Записывает заранее закодированное имя свойства и значение UInt32 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, UInt64)	Записывает заранее закодированное имя свойства и значение UInt64 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Decimal)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Decimal (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Double)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Double (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Int32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Int32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Int64)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Int64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Single)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Single (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, UInt32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение UInt32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, UInt64)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение UInt64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, Decimal)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Decimal (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, Double)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Double (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

WriteNumber(ReadOnlySpan<Char>, Int32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Int32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, Int64)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Int64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, Single)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Single (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, UInt32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение UInt32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, UInt64)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение UInt64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Decimal)	Записывает имя свойства, указываемое как строка, и значение Decimal (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Double)	Записывает имя свойства, указываемое как строка, и значение Double (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Int32)	Записывает имя свойства, указываемое как строка, и значение Int32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Int64)	Записывает имя свойства, указываемое как строка, и значение Int64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Single)	Записывает имя свойства, указываемое как строка, и значение Single (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, UInt32)	Записывает имя свойства, указываемое как строка, и значение UInt32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, UInt64)	Записывает имя свойства, указываемое как строка, и значение UInt64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumberValue(Decimal)	Записывает значение Decimal (в виде числа JSON) как элемент массива JSON.

WriteNumberValue(Double)	Записывает значение Double (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Int32)	Записывает значение Int32 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Int64)	Записывает значение Int64 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Single)	Записывает значение Single (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(UInt32)	Записывает значение UInt32 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(UInt64)	Записывает значение UInt64 (в виде числа JSON) как элемент массива JSON.
WritePropertyName(JsonEncodedText)	Записывает заранее закодированное имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(ReadOnlySpan<Byte>)	Записывает имя свойства в кодировке UTF-8 (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(ReadOnlySpan<Char>)	Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(String)	Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WriteRawValue(ReadOnlySequence<Byte>, Boolean)	Записывает входные данные в виде содержимого JSON. Ожидается, что входное содержимое является одним полным значением JSON.
WriteRawValue(ReadOnlySpan<Byte>, Boolean)	Записывает входные данные в виде содержимого JSON. Ожидается, что входное содержимое является одним полным значением JSON.
WriteRawValue(ReadOnlySpan<Char>, Boolean)	Записывает входные данные в виде содержимого JSON. Ожидается, что входное содержимое является одним полным значением JSON.
WriteRawValue(String, Boolean)	Записывает входные данные в виде содержимого JSON. Ожидается, что входное содержимое является одним полным значением JSON.
WriteStartArray()	Записывает начало массива JSON.
WriteStartArray(JsonEncodedText)	Записывает начало массива JSON с заранее закодированным именем свойства в качестве ключа.

WriteStartArray(ReadOnlySpan<Byte>)	Записывает начало массива JSON с именем свойства, указанным как доступный только для чтения диапазон байтов, в качестве ключа.
WriteStartArray(ReadOnlySpan<Char>)	Записывает начало массива JSON с именем свойства, указанным как доступный только для чтения диапазон символов, в качестве ключа.
WriteStartArray(String)	Записывает начало массива JSON с именем свойства, указанным как строка, в качестве ключа.
WriteStartObject()	Записывает начало объекта JSON.
WriteStartObject(JsonEncodedText)	Записывает начало объекта JSON с заранее закодированным именем свойства в качестве ключа.
WriteStartObject(ReadOnlySpan<Byte>)	Записывает начало объекта JSON с именем свойства, указанным как доступный только для чтения диапазон байтов, в качестве ключа.
WriteStartObject(ReadOnlySpan<Char>)	Записывает начало объекта JSON с именем свойства, указанным в качестве диапазона символов только для чтения в качестве ключа.
WriteStartObject(String)	Записывает начало объекта JSON с именем свойства, указанным как строка, в качестве ключа.
WriteString(JsonEncodedText, DateTime)	Записывает заранее закодированное имя свойства и значение DateTime (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, DateTimeOffset)	Записывает заранее закодированное имя свойства и значение DateTimeOffset (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, Guid)	Записывает заранее закодированное имя свойства и значение Guid (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, JsonEncodedText)	Записывает заранее закодированные имя свойства и значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, ReadOnlySpan<Byte>)	Записывает заранее закодированное имя свойства и текстовое значение в UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, ReadOnlySpan<Char>)	Записывает заранее закодированное имя свойства и текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

WriteString(JsonEncodedText, String)	Записывает заранее закодированное имя свойства и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, DateTime)	Записывает имя свойства UTF-8 и значение DateTime (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, DateTimeOffset)	Записывает имя свойства UTF-8 и значение DateTimeOffset (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, Guid)	Записывает имя свойства UTF-8 и значение Guid (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, JsonEncodedText)	Записывает имя свойства в кодировке UTF-8 и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)	Записывает имя свойства UTF-8 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Char>)	Записывает имя свойства UTF-8 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Byte>, String)	Записывает имя свойства UTF-8 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Char>, DateTime)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение DateTime (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Char>, DateTimeOffset)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение DateTimeOffset (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Char>, Guid)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение Guid (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Char>, JsonEncodedText)	Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(ReadOnlySpan<Char>, ReadOnlySpan<Byte>)	Записывает имя свойства UTF-16 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

<code>WriteString(ReadOnlySpan<Char>, ReadOnlySpan<Char>)</code>	Записывает имя свойства UTF-16 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, String)</code>	Записывает имя свойства UTF-16 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, DateTime)</code>	Записывает имя свойства, указываемое как строка, и значение <code>DateTime</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, DateTimeOffset)</code>	Записывает имя свойства, указываемое как строка, и значение <code>DateTimeOffset</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, Guid)</code>	Записывает имя свойства, указываемое как строка, и значение <code>Guid</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, JsonEncodedText)</code>	Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, ReadOnlySpan<Byte>)</code>	Записывает имя свойства, указываемое как строка, и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, ReadOnlySpan<Char>)</code>	Записывает имя свойства, указываемое как строка, и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, String)</code>	Записывает имя свойства, указываемое как строка, и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteStringValue(DateTime)</code>	Записывает значение <code>DateTime</code> (в виде строки JSON) в качестве элемента массива JSON.
<code>WriteStringValue(DateTimeOffset)</code>	Записывает значение <code>DateTimeOffset</code> (в виде строки JSON) в качестве элемента массива JSON.
<code>WriteStringValue(Guid)</code>	Записывает значение <code>Guid</code> (в виде строки JSON) в качестве элемента массива JSON.
<code>WriteStringValue(JsonEncodedText)</code>	Записывает заранее закодированное значение (в виде строки JSON) в качестве элемента массива JSON.
<code>WriteStringValue(ReadOnlySpan<Byte>)</code>	Записывает текстовое значение UTF-8 (в виде строки JSON) в качестве элемента массива JSON.

<code>WriteStringValue(ReadOnlySpan<Char>)</code>	Записывает текстовое значение UTF-16 (в виде строки JSON) в качестве элемента массива JSON.
<code>WriteStringValue(String)</code>	Записывает строковое текстовое значение (в виде строки JSON) в качестве элемента массива JSON.

Методы расширения

<code>ConfigureAwait(IAsyncDisposable, Boolean)</code>	Настраивает способ выполнения ожиданий для задач, возвращаемых из асинхронного высвобождаемого объекта.
--	---

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да  Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter Constructors

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

Utf8JsonWriter(IBufferWriter<Byte>, JsonWriterOptions)	Initializes a new instance of the Utf8JsonWriter class using the specified IBufferWriter<T> to write the output to and customization options.
Utf8JsonWriter(Stream, JsonWriterOptions)	Initializes a new instance of the Utf8JsonWriter class using the specified stream to write the output to and customization options.

Utf8JsonWriter(IBufferWriter<Byte>, JsonWriterOptions)

Initializes a new instance of the [Utf8JsonWriter](#) class using the specified [IBufferWriter<T>](#) to write the output to and customization options.

C#

```
public Utf8JsonWriter (System.Buffers.IBufferWriter<byte> bufferWriter,  
System.Text.Json.JsonWriterOptions options = default);
```

Parameters

bufferWriter [IBufferWriter<Byte>](#)

The destination for writing JSON text.

options [JsonWriterOptions](#)

Defines the customized behavior of the [Utf8JsonWriter](#). By default, it writes minimized JSON (with no extra white space) and validates that the JSON being written is structurally valid according to the JSON RFC.

Exceptions

ArgumentNullException

`bufferWriter` is `null`.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Utf8JsonWriter(Stream, JsonWriterOptions)

Initializes a new instance of the [Utf8JsonWriter](#) class using the specified stream to write the output to and customization options.

C#

```
public Utf8JsonWriter (System.IO.Stream utf8Json,  
System.Text.Json.JsonWriterOptions options = default);
```

Parameters

utf8Json Stream

The destination for writing JSON text.

options JsonWriterOptions

Defines the customized behavior of the [Utf8JsonWriter](#). By default, it writes minimized JSON (with no extra white space) and validates that the JSON being written is structurally valid according to the JSON RFC.

Exceptions

ArgumentNullException

`utf8Json` is `null`.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.BytesCommitted Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the total number of bytes committed to the output by the current instance so far.

C#

```
public long BytesCommitted { get; }
```

Property Value

[Int64](#)

The total number of bytes committed to the output by the [Utf8JsonWriter](#) so far.

Remarks

In the case of an [IBufferWriter<T>](#), this property indicates how much the IBufferWriter has advanced.

In the case of a [Stream](#), this property indicates how much data has been written to the stream.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

Utf8JsonWriter.BytesPending Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает число байтов, записанных к настоящему времени объектом [Utf8JsonWriter](#) и еще не зафиксированных в выходных данных.

C#

```
public int BytesPending { get; }
```

Значение свойства

[Int32](#)

Число байтов, записанных до сих пор с помощью [Utf8JsonWriter](#), которые еще не были записаны в выходные данные и не зафиксированы.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.CurrentDepth Свойство

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Возвращает глубину текущего токена.

C#

```
public int CurrentDepth { get; }
```

Значение свойства

[Int32](#)

Глубина текущего маркера.

Комментарии

Свойство `CurrentDepth` отслеживает рекурсивную глубину вложенных объектов или массивов в тексте JSON, написанном до сих пор.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.Options Property

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Gets the custom behavior when writing JSON using this instance, which indicates whether to format the output while writing, whether to skip structural JSON validation, and which characters to escape.

C#

```
public System.Text.Json.JsonWriterOptions Options { get; }
```

Property Value

[JsonWriterOptions](#)

The custom behavior of this instance of the writer for formatting, validating, and escaping.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.Dispose Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Фиксирует весь оставшийся текст JSON, который еще не был записан, и освобождает все ресурсы, используемые текущим экземпляром.

C#

```
public void Dispose();
```

Реализации

[Dispose\(\)](#)

Комментарии

В случае IBufferWriter, это продвигается в основе [IBufferWriter<T>](#) на основе того, что было написано до сих пор.

В случае Stream данные записываются в поток и сбрасывается.

Экземпляр [Utf8JsonWriter](#) нельзя использовать повторно после удаления.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.DisposeAsync Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Asynchronously commits any leftover JSON text that has not yet been flushed and releases all resources used by the current instance.

C#

```
public System.Threading.Tasks.ValueTask DisposeAsync();
```

Returns

[ValueTask](#)

A task representing the asynchronous dispose operation.

Implements

[DisposeAsync\(\)](#)

Remarks

In the case of [IBufferWriter](#), this advances the underlying [IBufferWriter<T>](#) based on what has been written so far.

In the case of [Stream](#), this writes the data to the stream and flushes it.

The [Utf8JsonWriter](#) instance cannot be reused after disposing.

This method stores in the task it returns all non-usage exceptions that the method's synchronous counterpart can throw. If an exception is stored into the returned task, that exception will be thrown when the task is awaited. Usage exceptions, such as [ArgumentException](#), are still thrown synchronously. For the stored exceptions, see the exceptions thrown by [Dispose\(\)](#).

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.Flush Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Commits the JSON text written so far, which makes it visible to the output destination.

C#

```
public void Flush();
```

Exceptions

[ObjectDisposedException](#)

This instance has been disposed.

Remarks

In the case of [IBufferWriter](#), this advances the underlying [IBufferWriter<T>](#) based on what has been written so far.

In the case of [Stream](#), this writes the data to the stream and flushes it.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

Utf8JsonWriter.FlushAsync(CancellationToken) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Асинхронно фиксирует текст JSON, записанный на данный момент, что делает его доступным для места назначения вывода.

C#

```
public System.Threading.Tasks.Task FlushAsync  
(System.Threading.CancellationToken cancellationToken = default);
```

Параметры

cancellationToken [CancellationToken](#)

Токен для отслеживания запросов отмены. Значение по умолчанию — [None](#).

Возвращаемое значение

[Task](#)

Задача, представляющая асинхронную операцию освобождения.

Исключения

[ObjectDisposedException](#)

Этот экземпляр удален.

[OperationCanceledException](#)

Маркер отмены был отменен. Это исключение сохраняется в возвращаемой задаче.

Комментарии

В случае `IBufferWriter`, это продвигается в основе `IBufferWriter<T>` на основе того, что было написано до сих пор.

В случае `Stream` при этом данные записываются в поток и асинхронно сбрасывают их, отслеживая запросы отмены.

Этот метод сохраняет в задаче все исключения, не относящиеся к использованию, которые может создавать синхронный аналог метода. Если исключение сохраняется в возвращаемой задаче, это исключение будет создано при ожидании задачи. Исключения использования, такие как `ArgumentException`, по-прежнему создаются синхронно. Хранимые исключения см. в разделе исключения, создаваемые `Flush()`.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.Reset Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

Reset(Stream)	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром Stream .
Reset()	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно.
Reset(IBufferWriter<Byte>)	Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром IBufferWriter<T> .

Reset(Stream)

Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром [Stream](#).

C#

```
public void Reset (System.IO.Stream utf8Json);
```

Параметры

utf8Json [Stream](#)

Назначение для записи текста JSON.

Исключения

[ArgumentNullException](#)

`utf8Json` имеет значение `null`.

ObjectDisposedException

Этот экземпляр удален.

Комментарии

Будет [Utf8JsonWriter](#) по-прежнему использовать исходные параметры записи, но теперь выполняет запись в `utf8Json` в качестве нового назначения.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Reset()

Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно.

C#

```
public void Reset();
```

Исключения

ObjectDisposedException

Этот экземпляр удален.

Комментарии

Будет [Utf8JsonWriter](#) по-прежнему использовать исходные параметры записи и исходные выходные данные (или [IBufferWriter<T>Stream](#)) в качестве назначения.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Reset(IBufferWriter<Byte>)

Восстанавливает внутреннее состояние этого экземпляра, чтобы его можно было использовать повторно с новым экземпляром [IBufferWriter<T>](#).

C#

```
public void Reset (System.Buffers.IBufferWriter<byte> bufferWriter);
```

Параметры

bufferWriter [IBufferWriter<Byte>](#)

Назначение для записи текста JSON.

Исключения

[ArgumentNullException](#)

`bufferWriter` имеет значение `null`.

[ObjectDisposedException](#)

Этот экземпляр удален.

Комментарии

Будет [Utf8JsonWriter](#) по-прежнему использовать исходные параметры записи, но теперь выполняет запись в `bufferWriter` в качестве нового назначения.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteBase64String

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteBase64String(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(ReadOnlySpan<Char>, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(String, ReadOnlySpan<Byte>)	Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.
WriteBase64String(JsonEncodedText, ReadOnlySpan<Byte>)	Записывает заранее закодированное имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.

WriteBase64String(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)

Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteBase64String (ReadOnlySpan<byte> utf8PropertyName,  
    ReadOnlySpan<byte> bytes);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя записываемого свойства в кодировке UTF-8.

bytes [ReadOnlySpan<Byte>](#)

Двоичные данные для записи в виде текста в кодировке Base64.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Максимальный допустимый размер двоичных данных для записи в формате Base64 составляет 125 000 000 байт (или приблизительно 125 МБ). Превышение этого ограничения приводит к возникновению [ArgumentException](#) ошибки.

Имя свойства экранируется, а байты кодируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBase64String(ReadOnlySpan<Char>, ReadOnlySpan<Byte>)

Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteBase64String (ReadOnlySpan<char> propertyName,
```

```
ReadOnlySpan<byte> bytes);
```

Параметры

propertyName `ReadOnlySpan<Char>`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

bytes `ReadOnlySpan<Byte>`

Двоичные данные для записи в виде текста в кодировке Base64.

Исключения

`ArgumentException`

Имя или значение указанного свойства слишком велико.

`InvalidOperationException`

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Максимальный допустимый размер двоичных данных для записи в формате Base64 составляет 125 000 000 байт (или приблизительно 125 МБ). Превышение этого ограничения приводит к возникновению `ArgumentException` ошибки.

Имя свойства экранируется, а байты кодируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBase64String(String, ReadOnlySpan<Byte>)

Записывает имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteBase64String (string propertyName, ReadOnlySpan<byte>  
bytes);
```

Параметры

propertyName `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

bytes `ReadOnlySpan<Byte>`

Двоичные данные для записи в виде текста в кодировке Base64.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Максимальный допустимый размер двоичных данных для записи в формате Base64 составляет 125 000 000 байт (или приблизительно 125 МБ). Превышение этого ограничения приводит к возникновению [ArgumentException](#) ошибки.

Имя свойства экранируется, а байты кодируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBase64String(JsonEncodedText, ReadOnlySpan<Byte>)

Записывает заранее закодированное имя свойства и необработанные байты (в виде строки JSON в кодировке Base64) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteBase64String (System.Text.Json.JsonEncodedText  
propertyName, ReadOnlySpan<byte> bytes);
```

Параметры

propertyName [JsonEncodedText](#)

Имя записываемого свойства в кодировке JSON.

bytes [ReadOnlySpan<Byte>](#)

Двоичные данные для записи в виде текста в кодировке Base64.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Максимальный допустимый размер двоичных данных для записи в формате Base64 составляет 125 000 000 байт (или приблизительно 125 МБ). Превышение этого ограничения приводит к возникновению [ArgumentException](#) ошибки.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Байты кодируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteBase64String Value(ReadOnlySpan<Byte>) Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Записывает необработанные байты (в виде строки JSON в кодировке Base64) в качестве элемента массива JSON.

C#

```
public void WriteBase64StringValue (ReadOnlySpan<byte> bytes);
```

Параметры

bytes [ReadOnlySpan<Byte>](#)

Двоичные данные, которые нужно записать в виде строкового элемента JSON массива JSON в кодировке Base64.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Максимальный допустимый размер двоичных данных для записи в формате Base64 составляет 125 000 000 байт (или приблизительно 125 МБ). Превышение этого ограничения приводит к возникновению [ArgumentException](#) ошибки.

Байты кодируются перед записью.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteBoolean Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

WriteBoolean(String, Boolean)	Writes a property name specified as a string and a Boolean value (as a JSON literal <code>true</code> or <code>false</code>) as part of a name/value pair of a JSON object.
WriteBoolean(ReadOnlySpan<Byte>, Boolean)	Writes a property name specified as a read-only span of bytes and a Boolean value (as a JSON literal <code>true</code> or <code>false</code>) as part of a name/value pair of a JSON object.
WriteBoolean(ReadOnlySpan<Char>, Boolean)	Writes a property name specified as a read-only character span and a Boolean value (as a JSON literal <code>true</code> or <code>false</code>) as part of a name/value pair of a JSON object.
WriteBoolean(JsonEncodedText, Boolean)	Writes the pre-encoded property name and Boolean value (as a JSON literal <code>true</code> or <code>false</code>) as part of a name/value pair of a JSON object.

WriteBoolean(String, Boolean)

Writes a property name specified as a string and a [Boolean](#) value (as a JSON literal `true` or `false`) as part of a name/value pair of a JSON object.

C#

```
public void WriteBoolean (string propertyName, bool value);
```

Parameters

propertyName [String](#)

The UTF-16 encoded property name of the JSON object to be transcoded and written as UTF-8.

value Boolean

The value to be written as a JSON literal `true` or `false` as part of the name/value pair.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

Validation is enabled, and the operation would result in writing invalid JSON.

[ArgumentNullException](#)

The `propertyName` parameter is `null`.

Remarks

The property name is escaped before writing.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBoolean(ReadOnlySpan<Byte>, Boolean)

Writes a property name specified as a read-only span of bytes and a [Boolean](#) value (as a JSON literal `true` or `false`) as part of a name/value pair of a JSON object.

C#

```
public void WriteBoolean (ReadOnlySpan<byte> utf8PropertyName, bool value);
```

Parameters

utf8PropertyName [ReadOnlySpan<Byte>](#)

The UTF-8 encoded property name of the JSON object to be written.

value Boolean

The value to be written as a JSON literal **true** or **false** as part of the name/value pair.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

Validation is enabled, and the operation would result in writing invalid JSON.

Remarks

The property name is escaped before writing.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBoolean(ReadOnlySpan<Char>, Boolean)

Writes a property name specified as a read-only character span and a [Boolean](#) value (as a JSON literal **true** or **false**) as part of a name/value pair of a JSON object.

C#

```
public void WriteBoolean (ReadOnlySpan<char> propertyName, bool value);
```

Parameters

propertyName [ReadOnlySpan<Char>](#)

The UTF-16 encoded property name of the JSON object to be transcoded and written as UTF-8.

value Boolean

The value to be written as a JSON literal **true** or **false** as part of the name/value pair.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

Validation is enabled, and the operation would result in writing invalid JSON.

Remarks

The property name is escaped before writing.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteBoolean(JsonEncodedText, Boolean)

Writes the pre-encoded property name and [Boolean](#) value (as a JSON literal `true` or `false`) as part of a name/value pair of a JSON object.

C#

```
public void WriteBoolean (System.Text.Json.JsonEncodedText propertyName,  
bool value);
```

Parameters

propertyName [JsonEncodedText](#)

The JSON encoded property name of the JSON object to be transcoded and written as UTF-8.

value [Boolean](#)

The value to be written as a JSON literal `true` or `false` as part of the name/value pair.

Exceptions

InvalidOperationException

Validation is enabled, and this method would result in writing invalid JSON.

Remarks

The property name should already be escaped when the instance of [JsonEncodedText](#) was created.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.WriteBooleanValue(Boolean) Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Writes a [Boolean](#) value (as a JSON literal **true** or **false**) as an element of a JSON array.

C#

```
public void WriteBooleanValue (bool value);
```

Parameters

value [Boolean](#)

The value to be written as a JSON literal **true** or **false** as an element of a JSON array.

Exceptions

[InvalidOperationException](#)

Validation is enabled, and the operation would result in writing invalid JSON.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.WriteCommentValue

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteCommentValue(ReadOnlySpan<Byte>)	Записывает текстовое значение UTF-8 в виде комментария JSON.
WriteCommentValue(ReadOnlySpan<Char>)	Записывает текстовое значение UTF-16 в виде комментария JSON.
WriteCommentValue(String)	Записывает строковое текстовое значение в виде комментария JSON.

WriteCommentValue(ReadOnlySpan<Byte>)

Записывает текстовое значение UTF-8 в виде комментария JSON.

C#

```
public void WriteCommentValue (ReadOnlySpan<byte> utf8Value);
```

Параметры

utf8Value [ReadOnlySpan<Byte>](#)

Значение в кодировке UTF-8, записываемое в виде комментария JSON, заключенного в `.../`.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

-ИЛИ-

`utf8Value` содержит разделитель комментария `(*/)`.

Комментарии

Значение комментария не экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteCommentValue(ReadOnlySpan<Char>)

Записывает текстовое значение UTF-16 в виде комментария JSON.

C#

```
public void WriteCommentValue (ReadOnlySpan<char> value);
```

Параметры

`value` `ReadOnlySpan<Char>`

Значение в кодировке UTF-16, записываемое в виде перекодированного в UTF-8 комментария JSON, заключенного в `/.../`.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

-ИЛИ-

`value` содержит разделитель комментария `(*/)`.

Комментарии

Значение комментария не экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteCommentValue(String)

Записывает строковое текстовое значение в виде комментария JSON.

C#

```
public void WriteCommentValue (string value);
```

Параметры

value `String`

Значение в кодировке UTF-16, записываемое в виде перекодированного в UTF-8 комментария JSON, заключенного в `/.../`.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

-ИЛИ-

`value` содержит разделитель комментария `(* /)`.

[ArgumentNullException](#)

Параметр `value` имеет значение `null`.

Комментарии

Значение комментария не экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteEndArray Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Записывает конец массива JSON.

C#

```
public void WriteEndArray();
```

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteEndObject Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Writes the end of a JSON object.

C#

```
public void WriteEndObject();
```

Exceptions

[InvalidOperationException](#)

Validation is enabled, and the operation would result in writing invalid JSON.

Applies to

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.WriteLine Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteNull(ReadOnlySpan<Byte>)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(ReadOnlySpan<Char>)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(String)	Записывает имя свойства, указываемое как строка, и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.
WriteNull(JsonEncodedText)	Записывает заранее закодированное имя свойства и литерал JSON <code>null</code> в составе пары "имя-значение" объекта JSON.

WriteNull(ReadOnlySpan<Byte>)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и литерал JSON `null` в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNull (ReadOnlySpan<byte> utf8PropertyName);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNull(ReadOnlySpan<Char>)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и литерал JSON **null** в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNull (ReadOnlySpan<char> propertyName);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNull(String)

Записывает имя свойства, указываемое как строка, и литерал JSON `null` в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNull (string propertyName);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

ArgumentNullException

Параметр `propertyName` имеет значение `null`.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNull(JsonEncodedText)

Записывает заранее закодированное имя свойства и литерал JSON `null` в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNull (System.Text.Json.JsonEncodedText propertyName);
```

Параметры

`propertyName` `JsonEncodedText`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

Исключения

InvalidOperationException

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра `JsonEncodedText`.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteLine Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Записывает литерал JSON **null** в качестве элемента массива JSON.

C#

```
public void WriteNullValue();
```

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Применяется к

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteNumber Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteNumber(String, Int32)	Записывает имя свойства, указываемое как строка, и значение Int32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Int64)	Записывает имя свойства, указываемое как строка, и значение Int64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Single)	Записывает имя свойства, указываемое как строка, и значение Single (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, UInt32)	Записывает имя свойства, указываемое как строка, и значение UInt32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, UInt64)	Записывает имя свойства, указываемое как строка, и значение UInt64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Single)	Записывает заранее закодированное имя свойства и значение Single (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Double)	Записывает заранее закодированное имя свойства и значение Double (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Int32)	Записывает заранее закодированное имя свойства и значение Int32 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Int64)	Записывает заранее закодированное имя свойства и значение Int64 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

WriteNumber(String, Double)	Записывает имя свойства, указываемое как строка, и значение Double (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, UInt32)	Записывает заранее закодированное имя свойства и значение UInt32 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, Decimal)	Записывает заранее закодированное имя свойства и значение Decimal (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(String, Decimal)	Записывает имя свойства, указываемое как строка, и значение Decimal (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(JsonEncodedText, UInt64)	Записывает заранее закодированное имя свойства и значение UInt64 (в виде номера JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Char>, UInt32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение UInt32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Decimal)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Decimal (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Double)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Double (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Int32)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Int32 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Int64)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Int64 (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
WriteNumber(ReadOnlySpan<Byte>, Single)	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение Single (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

<code>WriteNumber(ReadOnlySpan<Byte>, UInt32)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение <code>UInt32</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, UInt64)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>UInt64</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, Decimal)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Decimal</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, Double)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Double</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, Int32)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Int32</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, Int64)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Int64</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Char>, Single)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Single</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteNumber(ReadOnlySpan<Byte>, UInt64)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение <code>UInt64</code> (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

WriteNumber(String, Int32)

Записывает имя свойства, указываемое как строка, и значение `Int32` (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (string propertyName, int value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Int32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает , [Int32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, Int64)

Записывает имя свойства, указываемое как строка, и значение [Int64](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (string propertyName, long value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Int64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает `Int64` используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, Single)

Записывает имя свойства, указываемое как строка, и значение [Single](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (string propertyName, float value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Single](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает с [Single](#) помощью по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версий. Использует "G9" на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, UInt32)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как строка, и значение [UInt32](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (string propertyName, uint value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [UInt32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает `UInt32` используя значение по умолчанию `StandardFormat` (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, UInt64)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как строка, и значение `UInt64` (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (string propertyName, ulong value);
```

Параметры

`propertyName` `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value `UInt64`

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает , `UInt64` используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, Single)

Записывает заранее закодированное имя свойства и значение [Single](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,  
float value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Single](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [Single](#) используя значение по умолчанию [StandardFormat](#) (G) в .NET Core 3.0 или более поздних версиях. Использует G9 на любой другой платформе.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, Double)

Записывает заранее закодированное имя свойства и значение [Double](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,  
double value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Double](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [Double](#) используя значение по умолчанию [StandardFormat](#) (G) в .NET Core 3.0 или более поздних версиях. Использует G17 на любой другой платформе.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, Int32)

Записывает заранее закодированное имя свойства и значение [Int32](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,  
int value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Int32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [Int32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, Int64)

Записывает заранее закодированное имя свойства и значение [Int64](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,  
long value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Int64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [Int64](#) используя значение по умолчанию [StandardFormat](#) (G), например 32767.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, Double)

Записывает имя свойства, указываемое как строка, и значение [Double](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (string propertyName, double value);
```

Параметры

propertyName String

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value Double

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает `Double` используя значение по умолчанию `StandardFormat (G)` в .NET Core 3.0 или более поздних версиях. Использует `G17` на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, UInt32)

Важно!

Этот API несовместим с CLS.

Записывает заранее закодированное имя свойства и значение [UInt32](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,
uint value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [UInt32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [UInt32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, Decimal)

Записывает заранее закодированное имя свойства и значение [Decimal](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,  
decimal value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Decimal](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [Decimal](#) используя значение по умолчанию [StandardFormat](#) (то есть "G").

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(String, Decimal)

Записывает имя свойства, указываемое как строка, и значение [Decimal](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (string propertyName, decimal value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Decimal](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает , [Decimal](#) используя значение по умолчанию [StandardFormat](#) (то есть "G").

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(JsonEncodedText, UInt64)

Важно!

Этот API несовместим с CLS.

Записывает заранее закодированное имя свойства и значение [UInt64](#) (в виде номера JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (System.Text.Json.JsonEncodedText propertyName,
ulong value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [UInt64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Записывает , [UInt64](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#) .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, UInt32)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [UInt32](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (ReadOnlySpan<char> propertyName, uint value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value UInt32

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCCompliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [UInt32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, Decimal)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [Decimal](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, decimal value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [Decimal](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Decimal](#) используя значение по умолчанию [StandardFormat](#) (то есть "G").

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, Double)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [Double](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, double value);
```

Параметры

utf8PropertyName `ReadOnlySpan<Byte>`

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value `Double`

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

`ArgumentException`

Имя указанного свойства слишком длинное.

`InvalidOperationException`

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , `Double` используя значение по умолчанию `StandardFormat (G)` в .NET Core 3.0 или более поздних версиях. Использует G17 на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(`ReadOnlySpan<Byte>`, `Int32`)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [Int32](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, int value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [Int32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Int32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, Int64)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [Int64](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, long value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [Int64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Int64](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, Single)

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [Single](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, float value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [Single](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает с [Single](#) помощью по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версий. Использует "G9" на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, UInt32)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [UInt32](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, uint
value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [UInt32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

InvalidOperationException

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает [UInt32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, UInt64)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [UInt64](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (ReadOnlySpan<char> propertyName, ulong value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value UInt64

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [UInt64](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, Decimal)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Decimal](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<char> propertyName, decimal value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Decimal](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Decimal](#) используя значение по умолчанию [StandardFormat](#) (то есть "G").

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, Double)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Double](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<char> propertyName, double value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Double](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает с [Double](#) помощью по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версий. Использует "G17" на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, Int32)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Int32](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<char> propertyName, int value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Int32](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Int32](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, Int64)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Int64](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<char> propertyName, long value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Int64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , [Int64](#) используя значение по умолчанию [StandardFormat](#) (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Char>, Single)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Single](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteNumber (ReadOnlySpan<char> propertyName, float value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Single](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает с [Single](#) помощью по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версий. Использует "G9" на любой другой платформе.

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumber(ReadOnlySpan<Byte>, UInt64)

Важно!

Этот API несовместим с CLS.

Записывает имя свойства, указываемое как доступный только для чтения диапазон байтов, и значение [UInt64](#) (в виде числа JSON) в составе пары "имя-значение" объекта JSON.

C#

```
[System.CLSCompliant(false)]
public void WriteNumber (ReadOnlySpan<byte> utf8PropertyName, ulong
value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [UInt64](#)

Значение, которое нужно записать в виде номера JSON в составе пары имя-значение.

Атрибуты [CLSCoмpliantAttribute](#)

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

InvalidOperationException

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает `UInt64` используя значение по умолчанию `StandardFormat` (то есть G), например 32767.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteNumberValue

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteNumberValue(UInt64)	Записывает значение UInt64 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(UInt32)	Записывает значение UInt32 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Single)	Записывает значение Single (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Int64)	Записывает значение Int64 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Int32)	Записывает значение Int32 (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Double)	Записывает значение Double (в виде числа JSON) как элемент массива JSON.
WriteNumberValue(Decimal)	Записывает значение Decimal (в виде числа JSON) как элемент массива JSON.

WriteNumberValue(UInt64)

Важно!

Этот API несовместим с CLS.

Записывает значение [UInt64](#) (в виде числа JSON) как элемент массива JSON.

C#

```
[System.CLSCompliant(false)]  
public void WriteNumberValue (ulong value);
```

Параметры

value [UInt64](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение, [UInt64](#) используя значение по умолчанию [StandardFormat](#) (то есть "G"). Например, 32767.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(UInt32)

Важно!

Этот API несовместим с CLS.

Записывает значение [UInt32](#) (в виде числа JSON) как элемент массива JSON.

C#

```
[System.CLSCompliant(false)]  
public void WriteNumberValue (uint value);
```

Параметры

value [UInt32](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Атрибуты [CLSCompliantAttribute](#)

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение, [UInt32](#) используя значение по умолчанию [StandardFormat](#) (то есть "G"). Например, 32767.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(Single)

Записывает значение [Single](#) (в виде числа JSON) как элемент массива JSON.

C#

```
public void WriteNumberValue (float value);
```

Параметры

value [Single](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение с [Single](#) использованием значения по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версиях. Использует G9 на любой другой платформе.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(Int64)

Записывает значение [Int64](#) (в виде числа JSON) как элемент массива JSON.

C#

```
public void WriteNumberValue (long value);
```

Параметры

value [Int64](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение, [Int64](#) используя значение по умолчанию [StandardFormat](#) (то есть "G"). Например, 32767.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(Int32)

Записывает значение [Int32](#) (в виде числа JSON) как элемент массива JSON.

C#

```
public void WriteNumberValue (int value);
```

Параметры

value [Int32](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение, [Int32](#) используя значение по умолчанию [StandardFormat](#) (то есть "G"). Например, 32767.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(Double)

Записывает значение [Double](#) (в виде числа JSON) как элемент массива JSON.

C#

```
public void WriteNumberValue (double value);
```

Параметры

value [Double](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение с [Double](#) использованием значения по умолчанию [StandardFormat](#) (то есть G) в .NET Core 3.0 или более поздних версиях. Использует G17 на любой другой платформе.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteNumberValue(Decimal)

Записывает значение [Decimal](#) (в виде числа JSON) как элемент массива JSON.

C#

```
public void WriteNumberValue (decimal value);
```

Параметры

value [Decimal](#)

Значение, которое нужно записать в виде номера JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает значение, [Decimal](#) используя значение по умолчанию [StandardFormat](#) (то есть "G").

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WritePropertyName

Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WritePropertyName(JsonEncodedText)	Записывает заранее закодированное имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(String)	Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(ReadOnlySpan<Byte>)	Записывает имя свойства в кодировке UTF-8 (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.
WritePropertyName(ReadOnlySpan<Char>)	Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.

WritePropertyName(JsonEncodedText)

Записывает заранее закодированное имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.

C#

```
public void WritePropertyName (System.Text.Json.JsonEncodedText  
propertyName);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WritePropertyName(String)

Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.

C#

```
public void WritePropertyName (string propertyName);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

`propertyName` имеет значение `null`.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WritePropertyName(ReadOnlySpan<Byte>)

Записывает имя свойства в кодировке UTF-8 (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.

C#

```
public void WritePropertyName (ReadOnlySpan<byte> utf8PropertyName);
```

Параметры

`utf8PropertyName` [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WritePropertyName(ReadOnlySpan<Char>)

Записывает имя свойства (в виде строки JSON) в качестве первой части пары "имя-значение" объекта JSON.

C#

```
public void WritePropertyName (ReadOnlySpan<char> propertyName);
```

Параметры

propertyName `ReadOnlySpan<Char>`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteRawValue Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteRawValue(ReadOnlySequence<Byte>, Boolean)	Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.
WriteRawValue(ReadOnlySpan<Char>, Boolean)	Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.
WriteRawValue(String, Boolean)	Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.
WriteRawValue(ReadOnlySpan<Byte>, Boolean)	Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.

WriteRawValue(ReadOnlySequence<Byte>, Boolean)

Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.

C#

```
public void WriteRawValue (System.Buffers.ReadOnlySequence<byte>
utf8Json, bool skipInputValidation = false);
```

Параметры

utf8Json [ReadOnlySequence<Byte>](#)

Необработанное содержимое JSON для записи.

`skipInputValidation` Boolean

`false` для проверки того, являются ли входные данные полезными данными JSON, совместимыми с RFC 8259; `true`, чтобы пропустить проверку.

Исключения

[ArgumentException](#)

Длина входных данных равна нулю или равна [Int32.MaxValue](#).

[JsonException](#)

`skipInputValidation` имеет значение `false`, а входные данные не являются допустимым, полным, одним значением JSON в соответствии с [JSON RFC](#), или входной JSON превышает рекурсивную глубину 64.

Комментарии

При записи недоверенных значений JSON не устанавливайте значение `skipInputValidation true`, так как это может привести к записи недопустимого JSON или к записи в экземпляр модуля записи недопустимыми общими полезными данными.

При использовании этого метода входное содержимое будет записываться в назначение записи "как есть", если только проверка не завершится ошибкой (если она включена).

Значение [SkipValidation](#) экземпляра модуля записи учитывается при использовании этого метода.

Значения [Indented](#) и [Encoder](#) для экземпляра модуля записи не применяются при использовании этого метода.

Применяется к

▼ .NET 8

Продукт	Версии
.NET	8

WriteRawValue(ReadOnlySpan<Char>, Boolean)

Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.

C#

```
public void WriteRawValue (ReadOnlySpan<char> json, bool  
skipInputValidation = false);
```

Параметры

json `ReadOnlySpan<Char>`

Необработанное содержимое JSON для записи.

skipInputValidation `Boolean`

`false` для проверки того, являются ли входные данные полезными данными JSON, совместимыми с RFC 8259; `true` Иначе.

Исключения

`ArgumentException`

Длина входных данных равна нулю или больше 715 827 882 (`Int32.MaxValue / 3`).

`JsonException`

`skipInputValidation` имеет значение `false`, а входные данные не являются допустимым, полным значением JSON в соответствии с [JSON RFC](#), или входной `json` превышает рекурсивную глубину 64.

Комментарии

При записи недоверенных значений JSON не устанавливайте значение `skipInputValidation true`, так как это может привести к записи недопустимого JSON или к записи в экземпляр модуля записи недопустимыми общими полезными данными.

При использовании этого метода входное содержимое будет записываться в назначение записи "как есть", если только проверка не завершится ошибкой (если она включена).

Значение [SkipValidation](#) экземпляра модуля записи учитывается при использовании этого метода.

Значения [Indented](#) и [Encoder](#) для экземпляра модуля записи не применяются при использовании этого метода.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

WriteRawValue(String, Boolean)

Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.

C#

```
public void WriteRawValue (string json, bool skipInputValidation =  
false);
```

Параметры

json [String](#)

Необработанное содержимое JSON для записи.

skipInputValidation [Boolean](#)

`false` для проверки того, являются ли входные данные полезными данными JSON, совместимыми с RFC 8259; `true` Иначе.

Исключения

[ArgumentNullException](#)

`json` имеет значение `null`.

[ArgumentException](#)

Длина входных данных равна нулю или больше $715\ 827\ 882$ ([Int32.MaxValue](#) / 3).

JsonException

`skipInputValidation` имеет значение `false`, а входные данные не являются допустимым, полным значением JSON в соответствии с [JSON RFC](#), или входной json превышает рекурсивную глубину 64.

Комментарии

При записи недоверенных значений JSON не устанавливайте значение `skipInputValidation true`, так как это может привести к записи недопустимого JSON или к записи в экземпляр модуля записи недопустимыми общими полезными данными.

При использовании этого метода входное содержимое будет записываться в назначение записи "как есть", если только проверка не завершится ошибкой (если она включена).

Значение `SkipValidation` экземпляра модуля записи учитывается при использовании этого метода.

Значения `Indented` и `Encoder` для экземпляра модуля записи не применяются при использовании этого метода.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

WriteRawValue(ReadOnlySpan<Byte>, Boolean)

Записывает входные данные в виде содержимого JSON. Предполагается, что входное содержимое является одним полным значением JSON.

C#

```
public void WriteRawValue (ReadOnlySpan<byte> utf8Json, bool skipInputValidation = false);
```

Параметры

`utf8Json` `ReadOnlySpan<Byte>`

Необработанное содержимое JSON для записи.

`skipInputValidation` `Boolean`

`false` для проверки того, являются ли входные данные полезными данными JSON, совместимыми с RFC 8259; `true` Иначе.

Исключения

`ArgumentException`

Длина входных данных равна нулю или равна `Int32.MaxValue`.

`JsonException`

`skipInputValidation` имеет значение `false`, а входные данные не являются допустимым, полным значением JSON в соответствии с [JSON RFC](#), или входной json превышает рекурсивную глубину 64.

Комментарии

При записи недоверенных значений JSON не устанавливайте значение `skipInputValidation true`, так как это может привести к записи недопустимого JSON или к тому, что в модуль записи записываются недопустимые общие полезные данные.

При использовании этого метода входное содержимое будет записываться в назначение записи "как есть", если только проверка не завершится ошибкой (если она включена).

Значение `SkipValidation` экземпляра модуля записи учитывается при использовании этого метода.

Значения `Indented` и `Encoder` для экземпляра модуля записи не применяются при использовании этого метода.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteStartArray Method

Reference

Definition

Namespace: [System.Text.Json](#)

Assembly: System.Text.Json.dll

Overloads

WriteStartArray()	Writes the beginning of a JSON array.
WriteStartArray(ReadOnlySpan<Byte>)	Writes the beginning of a JSON array with a property name specified as a read-only span of bytes as the key.
WriteStartArray(ReadOnlySpan<Char>)	Writes the beginning of a JSON array with a property name specified as a read-only character span as the key.
WriteStartArray(String)	Writes the beginning of a JSON array with a property name specified as a string as the key.
WriteStartArray(JsonEncodedText)	Writes the beginning of a JSON array with a pre-encoded property name as the key.

WriteStartArray()

Writes the beginning of a JSON array.

C#

```
public void WriteStartArray ();
```

Exceptions

[InvalidOperationException](#)

The depth of the JSON exceeds the maximum depth of 1,000.

-or-

Validation is enabled, and this write operation would produce invalid JSON.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartArray(ReadOnlySpan<Byte>)

Writes the beginning of a JSON array with a property name specified as a read-only span of bytes as the key.

C#

```
public void WriteStartArray (ReadOnlySpan<byte> utf8PropertyName);
```

Parameters

utf8PropertyName `ReadOnlySpan<Byte>`

The UTF-8 encoded property name of the JSON array to be written.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

The depth of the JSON exceeds the maximum depth of 1,000.

-or-

Validation is enabled, and this write operation would produce invalid JSON.

Remarks

The property name is escaped before writing.

Applies to

- ▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartArray(ReadOnlySpan<Char>)

Writes the beginning of a JSON array with a property name specified as a read-only character span as the key.

C#

```
public void WriteStartArray (ReadOnlySpan<char> propertyName);
```

Parameters

propertyName `ReadOnlySpan<Char>`

The UTF-16 encoded property name of the JSON array to be transcoded and written as UTF-8.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

The depth of the JSON exceeds the maximum depth of 1,000.

-or-

Validation is enabled, and this write operation would produce invalid JSON.

Remarks

The property name is escaped before writing.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartArray(String)

Writes the beginning of a JSON array with a property name specified as a string as the key.

C#

```
public void WriteStartArray (string propertyName);
```

Parameters

propertyName `String`

The UTF-16 encoded property name of the JSON array to be transcoded and written as UTF-8.

Exceptions

[ArgumentException](#)

The specified property name is too large.

[InvalidOperationException](#)

The depth of the JSON exceeds the maximum depth of 1,000.

-or-

Validation is enabled, and this write operation would produce invalid JSON.

[ArgumentNullException](#)

The `propertyName` parameter is `null`.

Remarks

The property name is escaped before writing.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartArray(JsonEncodedText)

Writes the beginning of a JSON array with a pre-encoded property name as the key.

C#

```
public void WriteStartArray (System.Text.Json.JsonEncodedText  
propertyName);
```

Parameters

propertyName [JsonEncodedText](#)

The JSON encoded property name of the JSON array to be transcoded and written as UTF-8.

Exceptions

[InvalidOperationException](#)

The depth of the JSON has exceeded the maximum depth of 1,000.

-or-

Validation is enabled, and this method would result in writing invalid JSON.

Remarks

The property name should already be escaped when the instance of [JsonEncodedText](#) was created.

Applies to

▼ .NET 8 and other versions

Product	Versions
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

Utf8JsonWriter.WriteStartObject Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteStartObject(JsonEncodedText)	Записывает начало объекта JSON с заранее закодированным именем свойства в качестве ключа.
WriteStartObject(String)	Записывает начало объекта JSON с именем свойства, указанным как строка, в качестве ключа.
WriteStartObject(ReadOnlySpan<Char>)	Записывает начало объекта JSON с именем свойства, указанным в качестве диапазона символов только для чтения в качестве ключа.
WriteStartObject()	Записывает начало объекта JSON.
WriteStartObject(ReadOnlySpan<Byte>)	Записывает начало объекта JSON с именем свойства, указанным как доступный только для чтения диапазон байтов, в качестве ключа.

WriteStartObject(JsonEncodedText)

Записывает начало объекта JSON с заранее закодированным именем свойства в качестве ключа.

C#

```
public void WriteStartObject (System.Text.Json.JsonEncodedText  
propertyName);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

Исключения

[InvalidOperationException](#)

Глубина JSON превысила максимальную глубину, равную 1000.

-или-

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartObject(String)

Записывает начало объекта JSON с именем свойства, указанным как строка, в качестве ключа.

C#

```
public void WriteStartObject (string propertyName);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Глубина JSON превышает максимальную глубину, равную 1000.

-или-

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartObject(ReadOnlySpan<Char>)

Записывает начало объекта JSON с именем свойства, указанным в качестве диапазона символов только для чтения в качестве ключа.

C#

```
public void WriteStartObject (ReadOnlySpan<char> propertyName);
```

Параметры

`propertyName` `ReadOnlySpan<Char>`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Глубина JSON превышает максимальную глубину, равную 1000.

-ИЛИ-

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartObject()

Записывает начало объекта JSON.

C#

```
public void WriteStartObject();
```

Исключения

[InvalidOperationException](#)

Глубина JSON превышает максимальную глубину, равную 1000.

-ИЛИ-

Проверка включена, и операция приведет к записи недопустимого JSON.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStartObject(ReadOnlySpan<Byte>)

Записывает начало объекта JSON с именем свойства, указанным как доступный только для чтения диапазон байтов, в качестве ключа.

C#

```
public void WriteStartObject (ReadOnlySpan<byte> utf8PropertyName);
```

Параметры

utf8PropertyName `ReadOnlySpan<Byte>`

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Глубина JSON превышает максимальную глубину, равную 1000.

-или-

Проверка включена, и эта операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteString Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteString(JsonEncodedText, JsonEncodedText)	Записывает заранее закодированные имя свойства и значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, String)	Записывает заранее закодированное имя свойства и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, ReadOnlySpan<Char>)	Записывает заранее закодированное имя свойства и текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, ReadOnlySpan<Byte>)	Записывает заранее закодированное имя свойства и текстовое значение в UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, Guid)	Записывает заранее закодированное имя свойства и значение Guid (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, DateTimeOffset)	Записывает заранее закодированное имя свойства и значение DateTimeOffset (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(JsonEncodedText, DateTime)	Записывает заранее закодированное имя свойства и значение DateTime (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(String, JsonEncodedText)	Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
WriteString(String, String)	Записывает имя свойства, указываемое как строка, и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

<code>WriteString(String, ReadOnlySpan<Char>)</code>	Записывает имя свойства, указываемое как строка, и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, ReadOnlySpan<Byte>)</code>	Записывает имя свойства, указываемое как строка, и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, DateTimeOffset)</code>	Записывает имя свойства, указываемое как строка, и значение <code>DateTimeOffset</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, DateTime)</code>	Записывает имя свойства, указываемое как строка, и значение <code>DateTime</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(String, Guid)</code>	Записывает имя свойства, указываемое как строка, и значение <code>Guid</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, String)</code>	Записывает имя свойства UTF-16 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, DateTime)</code>	Записывает имя свойства UTF-8 и значение <code>DateTime</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, DateTimeOffset)</code>	Записывает имя свойства UTF-8 и значение <code>DateTimeOffset</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, JsonEncodedText)</code>	Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)</code>	Записывает имя свойства UTF-8 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Char>)</code>	Записывает имя свойства UTF-8 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, String)</code>	Записывает имя свойства UTF-8 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, Guid)</code>	Записывает имя свойства UTF-8 и значение <code>Guid</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

<code>WriteString(ReadOnlySpan<Char>, DateTime)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>DateTime</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, DateTimeOffset)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>DateTimeOffset</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, Guid)</code>	Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение <code>Guid</code> (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, ReadOnlySpan<Byte>)</code>	Записывает имя свойства UTF-16 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Char>, ReadOnlySpan<Char>)</code>	Записывает имя свойства UTF-16 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.
<code>WriteString(ReadOnlySpan<Byte>, JsonEncodedText)</code>	Записывает имя свойства в кодировке UTF-8 и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

WriteString(JsonEncodedText, JsonEncodedText)

Записывает заранее закодированные имя свойства и значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
System.Text.Json.JsonEncodedText value);
```

Параметры

propertyName `JsonEncodedText`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value `JsonEncodedText`

Значение в кодировке JSON, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства уже должны быть экранированы при создании экземпляра [JsonEncodedText](#).

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(JsonEncodedText, String)

Записывает заранее закодированное имя свойства и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
string? value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [String](#)

Значение, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Значение экранируется перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNull\(JsonEncodedText\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(JsonEncodedText, ReadOnlySpan<Char>)

Записывает заранее закодированное имя свойства и текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,
ReadOnlySpan<char> value);
```

Параметры

`propertyName` [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [ReadOnlySpan<Char>](#)

Значение, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Значение экранируется перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNull\(JsonEncodedText\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

[WriteString\(JsonEncodedText, ReadOnlySpan<Byte>\)](#)

Записывает заранее закодированное имя свойства и текстовое значение в UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
ReadOnlySpan<byte> utf8Value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

utf8Value [ReadOnlySpan<Byte>](#)

Значение в кодировке UTF-8, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Значение экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(JsonEncodedText, Guid)

Записывает заранее закодированное имя свойства и значение [Guid](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
Guid value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [Guid](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает объект [Guid](#) используя значение по умолчанию [StandardFormat](#) (то есть "D") в формате nnnnnnnn-nnnn-nnnn-nnnnnn-nnnnnnnn.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(JsonEncodedText, DateTimeOffset)

Записывает заранее закодированное имя свойства и значение [DateTimeOffset](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
DateTimeOffset value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [DateTimeOffset](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает, используя расширенный [DateTimeOffset](#) формат ISO 8601-1 (см. [поддержку DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000-07:00.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(JsonEncodedText, DateTime)

Записывает заранее закодированное имя свойства и значение [DateTime](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (System.Text.Json.JsonEncodedText propertyName,  
DateTime value);
```

Параметры

propertyName [JsonEncodedText](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке JSON.

value [DateTime](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает, используя расширенный [DateTime](#) формат ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000.

Имя свойства должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, JsonEncodedText)

Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName,  
System.Text.Json.JsonEncodedText value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

value [JsonEncodedText](#)

Значение в кодировке JSON, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Значение должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, String)

Записывает имя свойства, указываемое как строка, и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, string? value);
```

Параметры

propertyName `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value `String`

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Имя и значение свойства экранируются перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNull\(String\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, ReadOnlySpan<Char>)

Записывает имя свойства, указываемое как строка, и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, ReadOnlySpan<char> value);
```

Параметры

`propertyName` `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

`value` `ReadOnlySpan<Char>`

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, ReadOnlySpan<Byte>)

Записывает имя свойства, указываемое как строка, и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, ReadOnlySpan<byte>  
utf8Value);
```

Параметры

`propertyName` `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

`utf8Value` `ReadOnlySpan<Byte>`

Значение в кодировке UTF-8, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, DateTimeOffset)

Записывает имя свойства, указываемое как строка, и значение [DateTimeOffset](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, DateTimeOffset value);
```

Параметры

`propertyName` [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

`value` [DateTimeOffset](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает, используя расширенный `DateTimeOffset` формат ISO 8601-1 (см. [поддержку DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000-07:00.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, DateTime)

Записывает имя свойства, указываемое как строка, и значение `DateTime` (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, DateTime value);
```

Параметры

`propertyName` `String`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [DateTime](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает, используя расширенный [DateTime](#) формат ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(String, Guid)

Записывает имя свойства, указываемое как строка, и значение [Guid](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (string propertyName, Guid value);
```

Параметры

propertyName [String](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [Guid](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

[ArgumentNullException](#)

Параметр `propertyName` имеет значение `null`.

Комментарии

Записывает объект `Guid` используя значение по умолчанию [StandardFormat](#) (то есть "D") в формате nnnnnnnn-nnnn-nnnn-nnnnnnnn-nnnnnnnn. Имя свойства сканируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, String)

Записывает имя свойства UTF-16 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName, string? value);
```

Параметры

propertyName `ReadOnlySpan<Char>`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value `String`

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNull\(ReadOnlySpan<Char>\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, DateTime)

Записывает имя свойства UTF-8 и значение [DateTime](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName, DateTime  
value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [DateTime](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , используя расширенный [DateTime](#) формат ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, DateTimeOffset)

Записывает имя свойства UTF-8 и значение [DateTimeOffset](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName,  
DateTimeOffset value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [DateTimeOffset](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , используя расширенный [DateTimeOffset](#) формат ISO 8601-1 (см. [поддержку DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000-07:00.

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, JsonEncodedText)

Записывает имя свойства и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName,  
System.Text.Json.JsonEncodedText value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8.

value [JsonEncodedText](#)

Значение в кодировке JSON, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Значение должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Byte>)

Записывает имя свойства UTF-8 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName,  
ReadOnlySpan<byte> utf8Value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

utf8Value [ReadOnlySpan<Byte>](#)

Значение в кодировке UTF-8, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, ReadOnlySpan<Char>)

Записывает имя свойства UTF-8 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName,  
ReadOnlySpan<char> value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [ReadOnlySpan<Char>](#)

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, String)

Записывает имя свойства UTF-8 и строковое текстовое значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName, string?  
value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [String](#)

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNull\(ReadOnlySpan<Byte>\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, Guid)

Записывает имя свойства UTF-8 и значение [Guid](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName, Guid value);
```

Параметры

`utf8PropertyName` [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

`value` [Guid](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает объект [Guid](#) используя значение по умолчанию [StandardFormat](#) (то есть "D") в формате nnnnnnnn-nnnn-nnnn-nnnnnnnn-nnnnnnnn. Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, DateTime)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [DateTime](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName, DateTime value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [DateTime](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

InvalidOperationException

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает, используя расширенный [DateTime](#) формат ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000.

Имя свойства экранируется перед записью.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, DateTimeOffset)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [DateTimeOffset](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName, DateTimeOffset value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [DateTimeOffset](#)

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает , используя расширенный [DateTimeOffset](#) формат ISO 8601-1 (см. [поддержку DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000-07:00.

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, Guid)

Записывает имя свойства, указываемое как доступный только для чтения диапазон символов, и значение [Guid](#) (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName, Guid value);
```

Параметры

propertyName `ReadOnlySpan<Char>`

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value `Guid`

Значение, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Записывает объект `Guid` используя значение по умолчанию `StandardFormat` (то есть "D") в формате nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnn. Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(`ReadOnlySpan<Char>`, `ReadOnlySpan<Byte>`)

Записывает имя свойства UTF-16 и текстовое значение UTF-8 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName,
```

```
ReadOnlySpan<byte> utf8Value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

utf8Value [ReadOnlySpan<Byte>](#)

Значение в кодировке UTF-8, которое нужно записать в виде строки JSON в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Char>, ReadOnlySpan<Char>)

Записывает имя свойства UTF-16 и текстовое значение UTF-16 (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<char> propertyName,  
ReadOnlySpan<char> value);
```

Параметры

propertyName [ReadOnlySpan<Char>](#)

Имя свойства объекта JSON, который нужно перекодировать и записать в формате UTF-8, в кодировке UTF-16.

value [ReadOnlySpan<Char>](#)

Значение в кодировке UTF-16, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары имя-значение.

Исключения

[ArgumentException](#)

Имя или значение указанного свойства слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Имя и значение свойства экранируются перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteString(ReadOnlySpan<Byte>, JsonEncodedText)

Записывает имя свойства в кодировке UTF-8 и заранее закодированное значение (в виде строки JSON) в составе пары "имя-значение" объекта JSON.

C#

```
public void WriteString (ReadOnlySpan<byte> utf8PropertyName,  
System.Text.Json.JsonEncodedText value);
```

Параметры

utf8PropertyName [ReadOnlySpan<Byte>](#)

Имя свойства объекта JSON, который нужно записать, в кодировке UTF-8.

value [JsonEncodedText](#)

Значение в кодировке JSON, которое нужно записать в виде строки JSON, перекодированной в UTF-8, в составе пары "имя-значение".

Исключения

[ArgumentException](#)

Имя указанного свойства слишком длинное.

[InvalidOperationException](#)

Проверка включена, и этот метод приведет к записи недопустимого JSON.

Комментарии

Значение должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Имя свойства экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)

Utf8JsonWriter.WriteStringValue Метод

Ссылка

Определение

Пространство имен: [System.Text.Json](#)

Сборка: System.Text.Json.dll

Перегрузки

WriteStringValue(String)	Записывает строковое текстовое значение (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(DateTime)	Записывает значение DateTime (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(DateTimeOffset)	Записывает значение DateTimeOffset (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(Guid)	Записывает значение Guid (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(ReadOnlySpan<Byte>)	Записывает текстовое значение UTF-8 (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(ReadOnlySpan<Char>)	Записывает текстовое значение UTF-16 (в виде строки JSON) в качестве элемента массива JSON.
WriteStringValue(JsonEncodedText)	Записывает заранее закодированное значение (в виде строки JSON) в качестве элемента массива JSON.

WriteStringValue(String)

Записывает строковое текстовое значение (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (string? value);
```

Параметры

value String

Значение, которое нужно записать в виде перекодированного в UTF-8 строкового элемента JSON массива JSON, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Значение экранируется перед записью.

Если `value` имеет значение `null`, то значение NULL JSON записывается, как если бы [WriteNullValue\(\)](#) был вызван метод .

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(DateTime)

Записывает значение [DateTime](#) (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (DateTime value);
```

Параметры

value DateTime

Значение, которое нужно записать в виде строки JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает с использованием расширенного [DateTime](#) формата ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(DateTimeOffset)

Записывает значение [DateTimeOffset](#) (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (DateTimeOffset value);
```

Параметры

value [DateTimeOffset](#)

Значение, которое нужно записать в виде строки JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает с [DateTimeOffset](#) использованием расширенного формата ISO 8601-1 (см. [раздел Поддержка DateTime и DateTimeOffset в System.Text.Json](#)); например, 2017-06-12T05:30:45.7680000-07:00.

Применяется к

- ▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(Guid)

Записывает значение [Guid](#) (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (Guid value);
```

Параметры

value [Guid](#)

Значение, которое нужно записать в виде строки JSON в качестве элемента массива JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция приведет к записи недопустимого JSON.

Комментарии

Этот метод записывает [Guid](#) значение, используя значение по умолчанию [StandardFormat](#) (то есть "D"), в виде: nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnn.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(ReadOnlySpan<Byte>)

Записывает текстовое значение UTF-8 (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (ReadOnlySpan<byte> utf8Value);
```

Параметры

utf8Value `ReadOnlySpan<Byte>`

Значение, которое нужно записать в виде строкового элемента JSON массива JSON, в кодировке UTF-8.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Значение экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(ReadOnlySpan<Char>)

Записывает текстовое значение UTF-16 (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (ReadOnlySpan<char> value);
```

Параметры

value `ReadOnlySpan<Char>`

Значение, которое нужно записать в виде перекодированного в UTF-8 строкового элемента JSON массива JSON, в кодировке UTF-16.

Исключения

[ArgumentException](#)

Указанное значение слишком велико.

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Значение экранируется перед записью.

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

WriteStringValue(JsonEncodedText)

Записывает заранее закодированное значение (в виде строки JSON) в качестве элемента массива JSON.

C#

```
public void WriteStringValue (System.Text.Json.JsonEncodedText value);
```

Параметры

value [JsonEncodedText](#)

Значение, которое нужно записать в виде перекодированного в UTF-8 строкового элемента JSON массива JSON, в кодировке JSON.

Исключения

[InvalidOperationException](#)

Проверка включена, и операция записи приведет к получению недопустимого JSON.

Комментарии

Значение должно быть экранировано при создании экземпляра [JsonEncodedText](#).

Применяется к

▼ .NET 8 и другие версии

Продукт	Версии
.NET	Core 3.0, Core 3.1, 5, 6, 7, 8

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Отзыв о продукте](#) | [Получить справку в Microsoft Q&A](#)