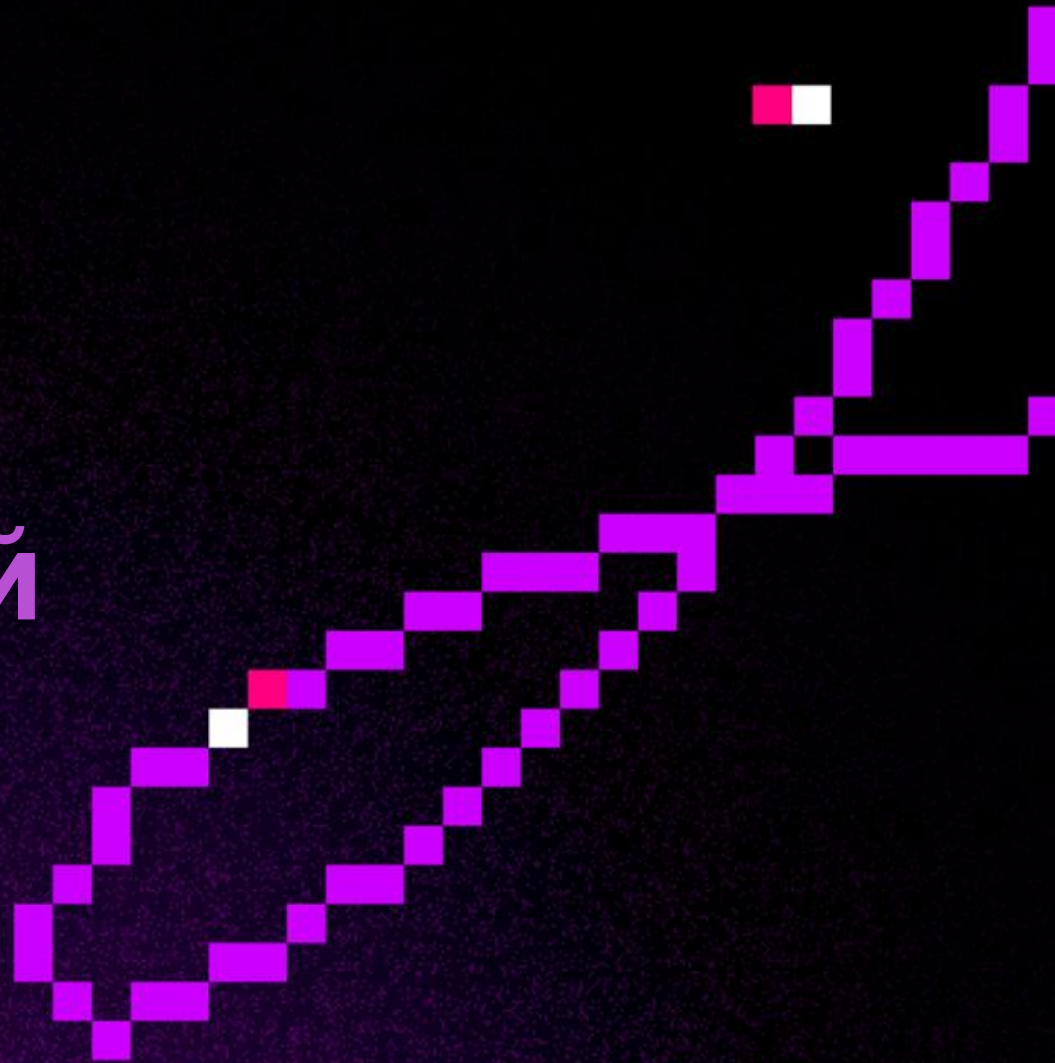


# System.Text.Json в .NET 10: из «и так норм» в производительный конвейер

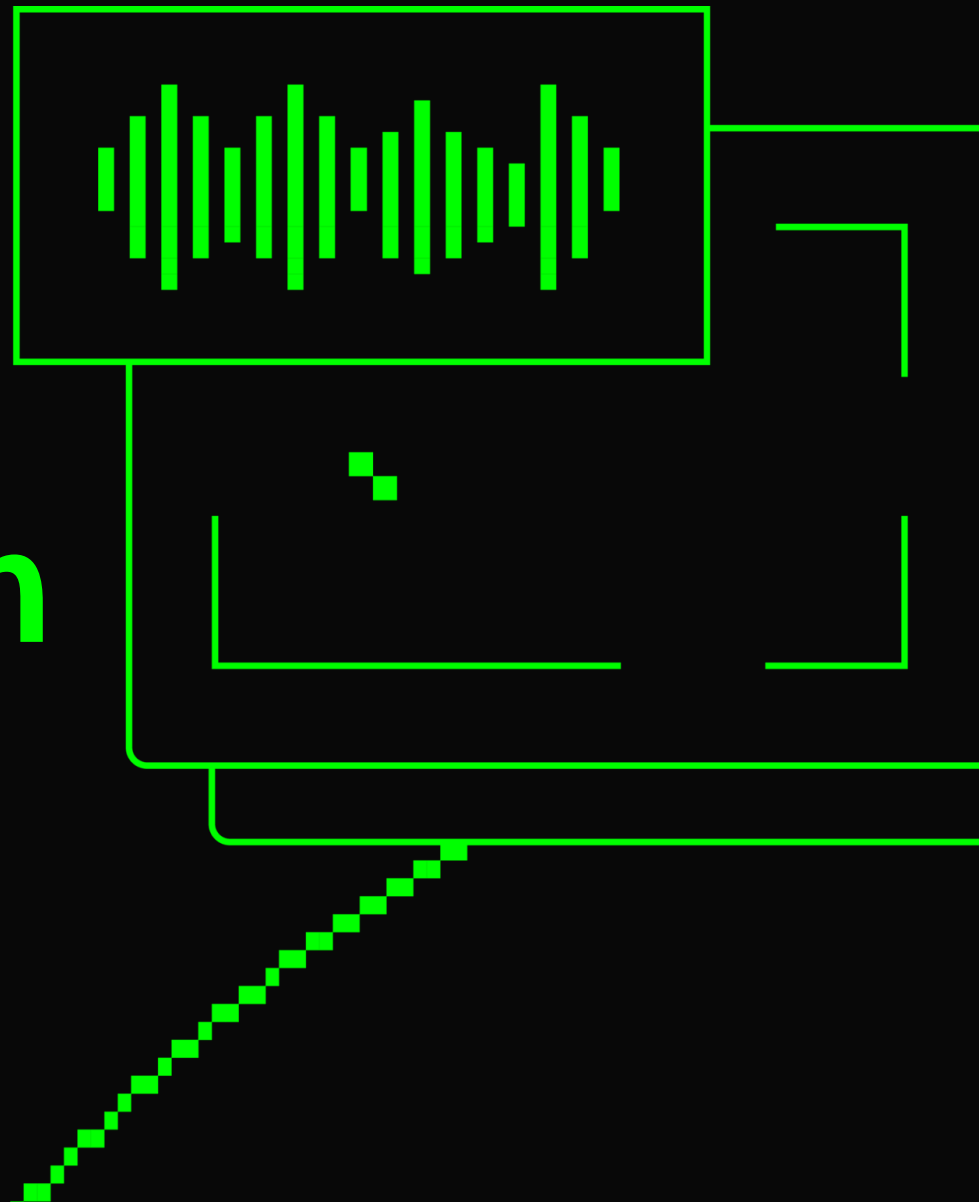
Контур

Мамченко Дмитрий

Инженер-программист



# Нововведения в **System.Text.Json**



# Слои в System.Text.Json

- Публичное API
- Слой сериализации
- Метаданные / контракты
- Конвертеры
- Низкий уровень JSON IO
- Буферы / пулы

# Слои в System.Text.Json

- Публичное API
- Слой сериализации
- Метаданные / контракты
- Конвертеры
- Низкий уровень JSON IO
- Буферы / пулы

# Мотивация добавить AllowDuplicateProperties

```
// БЕЗ проверки на дубликаты – присутствует УЯЗВИМОСТЬ
var loginRequest = JsonSerializer.Deserialize<LoginRequest>(json);

// Вредоносная нагрузка:
// {"username":"hacker","password":"pass","isAdmin":true,"isAdmin":false}
// Будет использовано последнее значение, но валидатор проверил только первое значение
// Результат: злоумышленник обходит защиту IsAdmin.

public class LoginRequest
{
    public string Username { get; set; }
    public string Password { get; set; }
    public bool IsAdmin { get; set; } = false; // Никогда не должно приходить true от клиента
}
```

# Мотивация добавить AllowDuplicateProperties

```
// БЕЗ проверки на дубликаты – присутствует УЯЗВИМОСТЬ
var loginRequest = JsonSerializer.Deserialize<LoginRequest>(json);

// Вредоносная нагрузка:
// {"username":"hacker","password":"pass","isAdmin":true,"isAdmin":false}
// Будет использовано последнее значение, но валидатор проверил только первое значение
// Результат: злоумышленник обходит защиту IsAdmin.

public class LoginRequest
{
    public string Username { get; set; }
    public string Password { get; set; }
    public bool IsAdmin { get; set; } = false; // Никогда не должно приходить true от клиента
}
```



CVE-2022-25757

# Мотивация добавить AllowDuplicateProperties

```
// С ВКЛЮЧЕННОЙ проверкой дубликатов – БЕЗОПАСНО  
var options = new JsonSerializerOptions  
{  
    AllowDuplicateProperties = false  
};  
  
var loginRequest = JsonSerializer.Deserialize<LoginRequest>(json, options);  
// Будет выброшено исключение JsonException: Duplicate property 'isAdmin' detected
```

# JsonSerializerOptions.Strict

```
var strict = new JsonSerializerOptions
{
    // 1) Неизвестные поля -> throw JsonException
    UnmappedMemberHandling = JsonUnmappedMemberHandling.Disallow,
    // 2) Повторяющиеся свойства -> throw JsonException
    AllowDuplicateProperties = false,
    // 3) Case-sensitive matching имен свойств
    PropertyNameCaseInsensitive = false,
    // 4) Учитываем Nullable-аннотации
    RespectNullableAnnotations = true,
    // 5) Требуем обязательные параметры ctor
    RespectRequiredConstructorParameters = true,
};
```



# Новая перегрузка JsonElement.Parse

```
[Benchmark]
public JsonElement WithClone()
{
    using JsonDocument d = JsonDocument.Parse(JsonString);
    return d.RootElement.Clone();
}
```

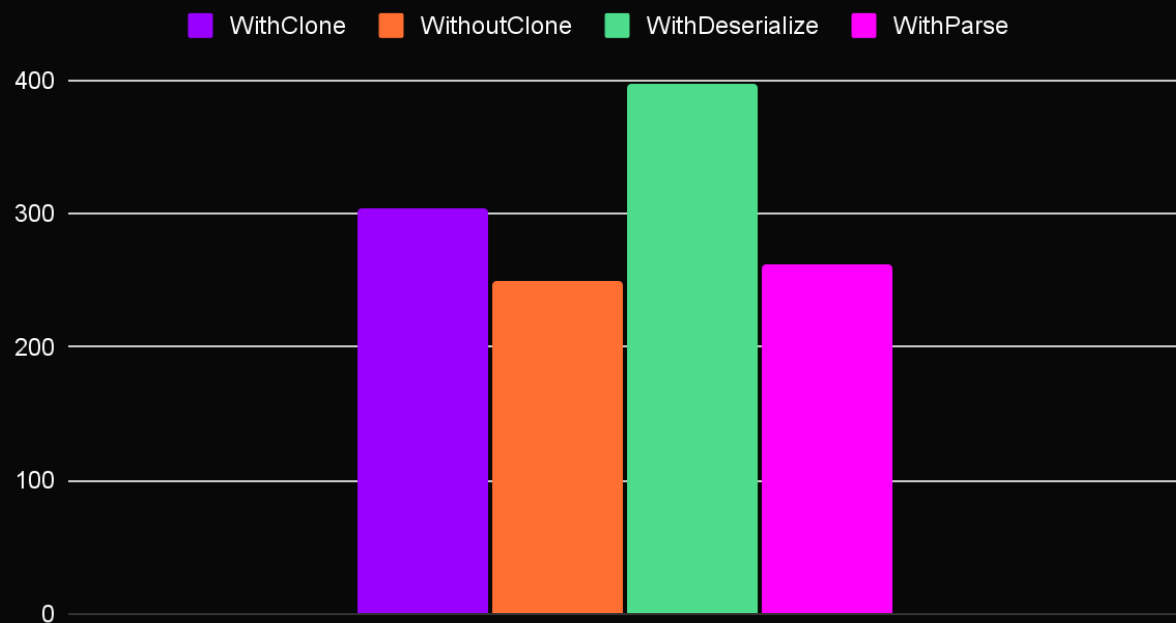
```
[Benchmark]
public JsonElement WithoutClone() =>
    JsonDocument.Parse(JsonString).RootElement;
```

```
[Benchmark]
public JsonElement WithDeserialize() =>
    JsonSerializer.Deserialize<JsonElement>(JsonString);
```

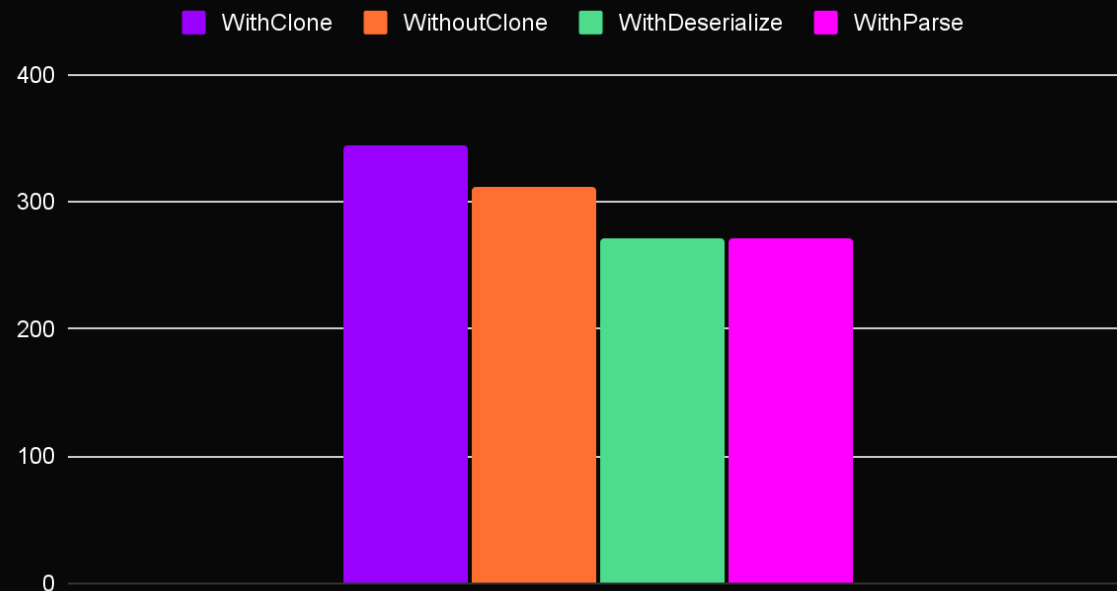
```
[Benchmark]
public JsonElement WithParse() =>
    JsonElement.Parse(JsonString); // NEW in .NET 10
```

# Новая перегрузка JsonElement.Parse

Mean (ms)



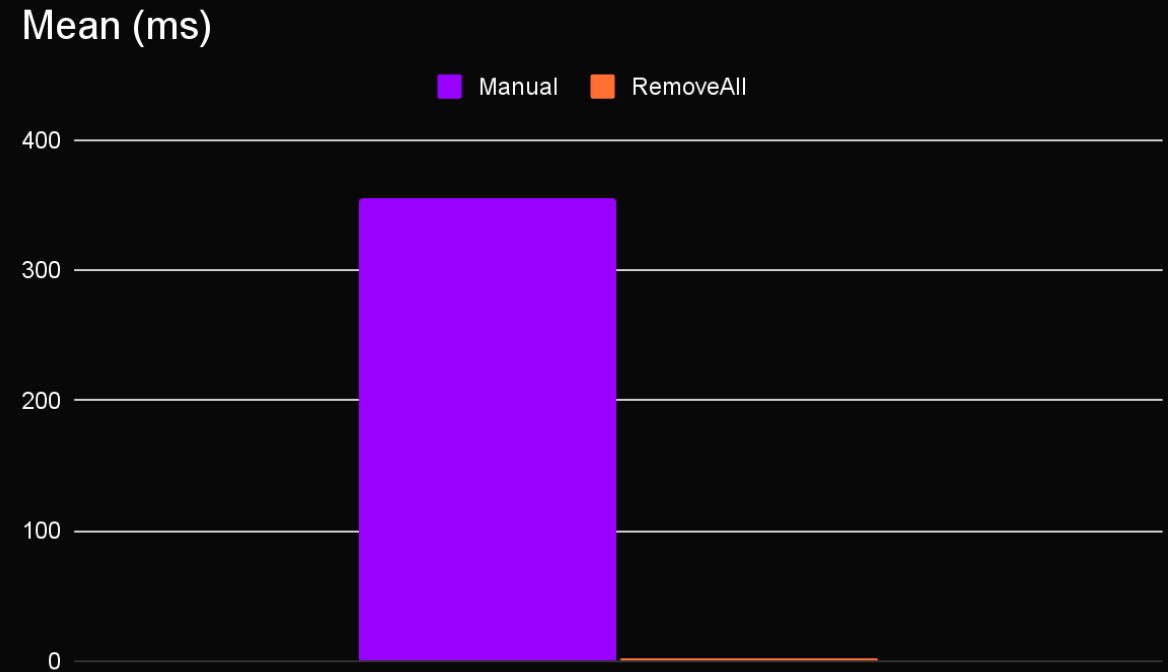
Allocated (B)



# Новые методы JsonArray: RemoveAll RemoveRange

```
[Benchmark]
public void Manual()
{
    int i = 0;
    while (i < _arr.Count)
    {
        if (_arr[i].GetValue<int>() % 2 == 0)
            _arr.RemoveAt(i);
        else
            i++;
    }
}
```

```
[Benchmark]
public void RemoveAll()
    => _arr.RemoveAll(static n => n.GetValue<int>() % 2 == 0);
```



# Новые методы JsonObject: TryAdd TryGetPropertyValue

```
public bool TryAdd(string propertyName, JsonNode? value, out int index)
{
    if (propertyName is null)
    {
        ThrowHelper.ThrowArgumentNullException(nameof(propertyName));
    }
    #if NET9_0
        bool success = Dictionary.TryAdd(propertyName, value);
        index = success ? Dictionary.Count - 1 : Dictionary.IndexOf(propertyName);
    #else
        bool success = Dictionary.TryAdd(propertyName, value, out index);
    #endif
    if (success)
    {
        value?.AssignParent(this);
    }
    return success;
}
```

# Новые методы JsonObject: TryAdd TryGetPropertyValue

```
public bool TryGetPropertyValue(string propertyName, out JsonNode? jsonNode, out int index)
{
    ArgumentNullException.ThrowIfNull(propertyName);

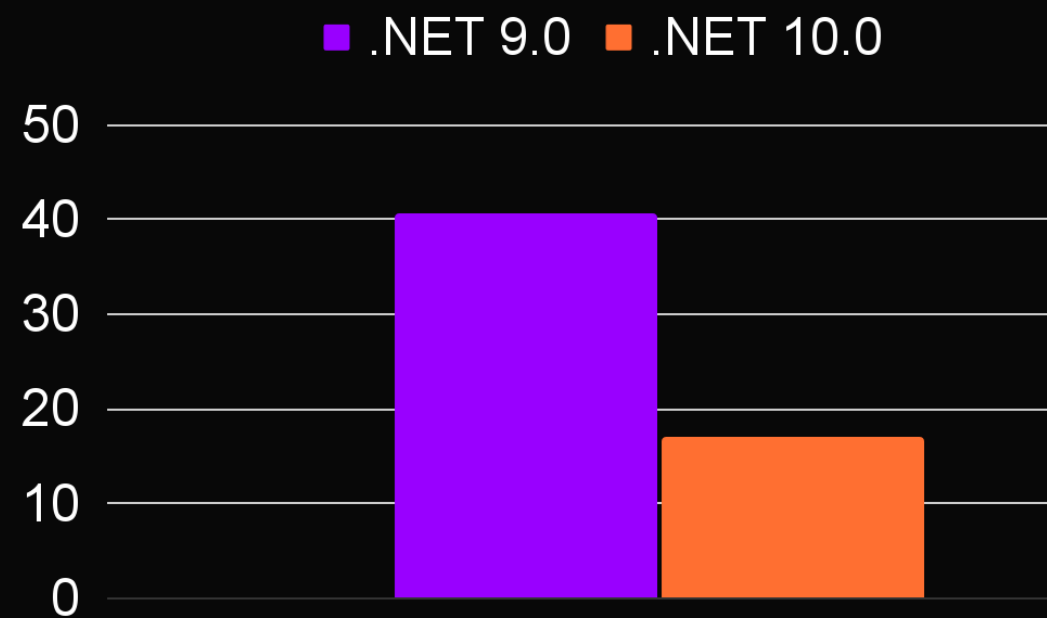
    return Dictionary.TryGetValue(propertyName, out jsonNode);
#if NET9_0
    index = Dictionary.IndexOf(propertyName);
    if (index < 0)
    {
        jsonNode = null;
        return false;
    }

    jsonNode = Dictionary.GetAt(index).Value;
    return true;
#else
    return Dictionary.TryGetValue(propertyName, out jsonNode, out index);
#endif
}
```

# Новые методы JsonObject: TryAdd TryGetPropertyValue

```
private JsonObject _obj = new();  
  
[Benchmark]  
public void Set() => _obj["key"] = "value";
```

Mean (ms)



~50%

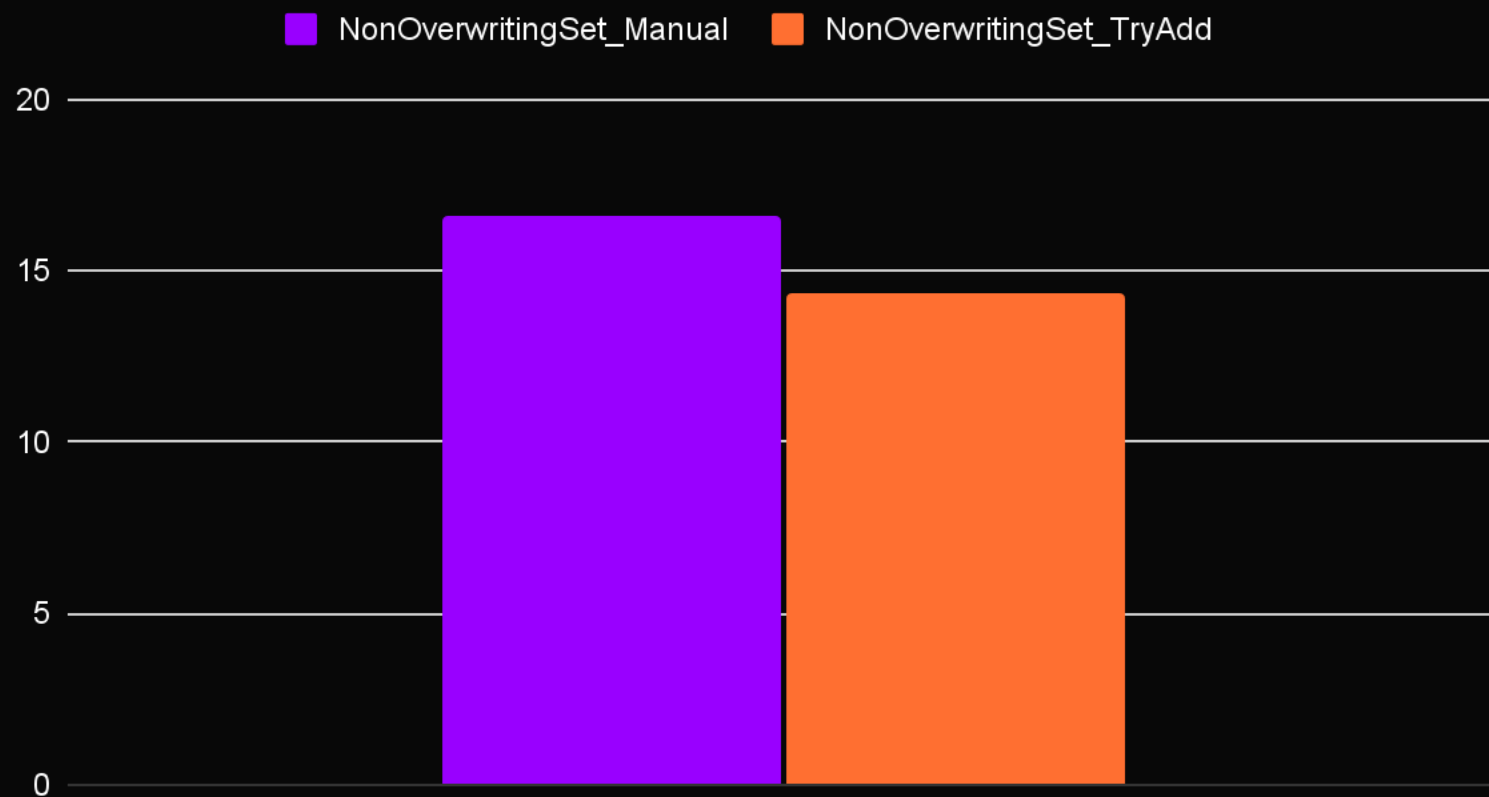
# Новые методы JsonObject: TryAdd TryGetPropertyValue

```
[Benchmark(Baseline = true)]
public void NonOverwritingSet_Manual()
{
    _obj.Remove("key");
    if (!_obj.ContainsKey("key"))
    {
        _obj.Add("key", _value);
    }
}

[Benchmark]
public void NonOverwritingSet_TryAdd()
{
    _obj.Remove("key");
    _obj.TryAdd("key", _value);
}
```

# Новые методы JsonObject: TryAdd TryGetPropertyValue

Mean (ms)



~14%



# Добавление PipeReader

[API Proposal]: Add PipeReader and PipeWriter overloads to JsonSerializer.SerializeAsync/DeserializeAsync #68586

✓ Closed

#116947



davidfowl opened on Apr 27, 2022 · edited by eiriktsarpalis

Edits ▾

Member

⋮

Background and motivation

# Добавление PipeReader



<https://github.com/gragra33/Utf8JsonAsyncStreamReader>

# Мотивация

```
// СТАРЫЙ ПОДХОД  
var pipe = GetPipeReader();  
var stream = pipe.AsStream(); // Создает промежуточный буфер  
var data = await JsonSerializer.DeserializeAsync<MyData>(stream);
```

# Мотивация

```
// СТАРЫЙ ПОДХОД  
var pipe = GetPipeReader();  
var stream = pipe.AsStream(); // Создает промежуточный буфер  
var data = await JsonSerializer.DeserializeAsync<MyData>(stream);
```

- Нужно создавать Stream
- Скопировать данные из буфера Pipe в буфер Stream
- Снижала производительность

# Мотивация

- ASP.NET Core
- SignalR
- gRPC

# Бенчмарки

[Benchmark]

```
public async ValueTask PipeReader_Old_AsStream()
{
    var reader = PipeReader.Create(new ReadOnlySequence<byte>(_jsonUtf8));
    await using var stream = reader.AsStream(leaveOpen: true);
    _ = await JsonSerializer.DeserializeAsync(stream, PayloadJsonContext.Default.Payload);
    await reader.CompleteAsync();
}
```

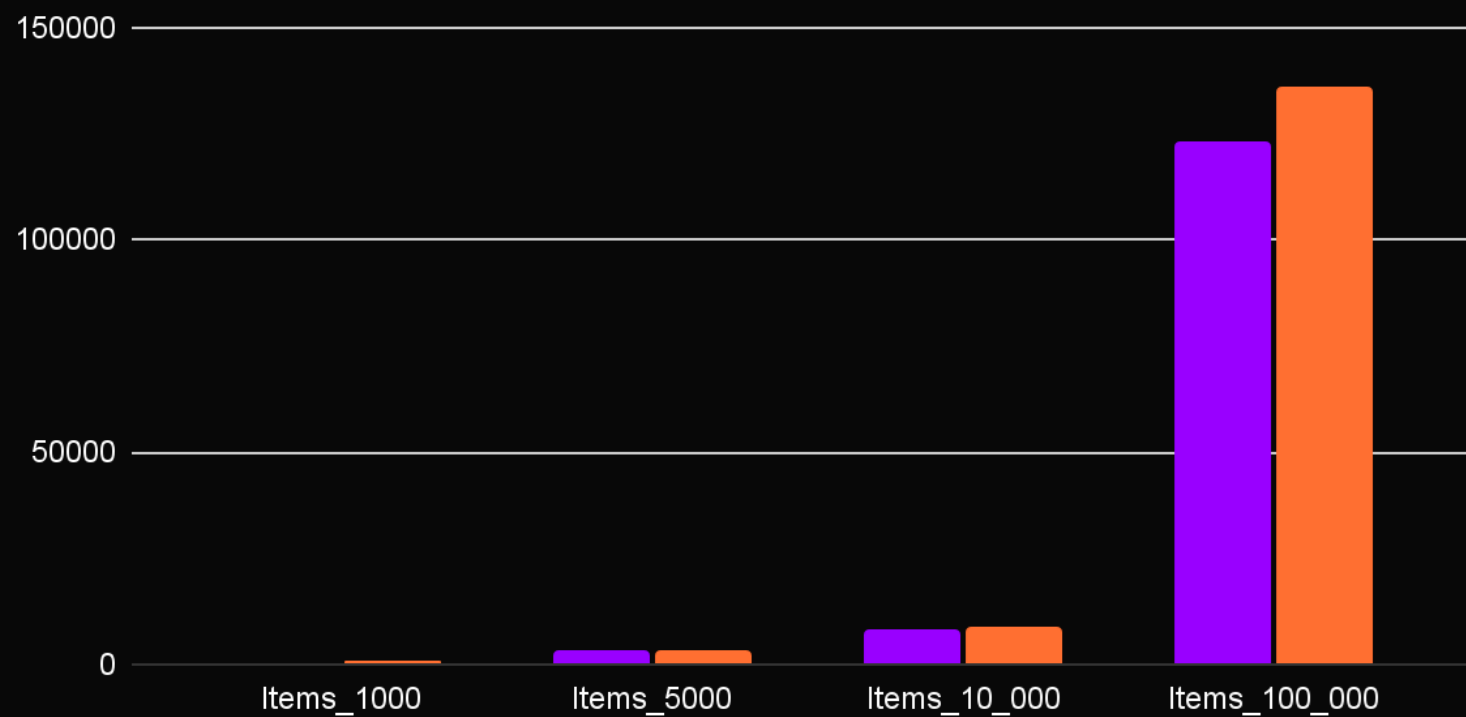
[Benchmark]

```
public async ValueTask PipeReader_NewOverload()
{
    var reader = PipeReader.Create(new ReadOnlySequence<byte>(_jsonUtf8));
    _ = await JsonSerializer.DeserializeAsync(reader, PayloadJsonContext.Default.Payload);
    await reader.CompleteAsync();
}
```

# Бенчмарки

Mean (ms)

PipeReader\_NewOverload PipeReader\_Old\_AsStream



~9%

# Новые методы Utf8JsonWriter

- WriteStringValueSegment
- WriteBase64StringSegment



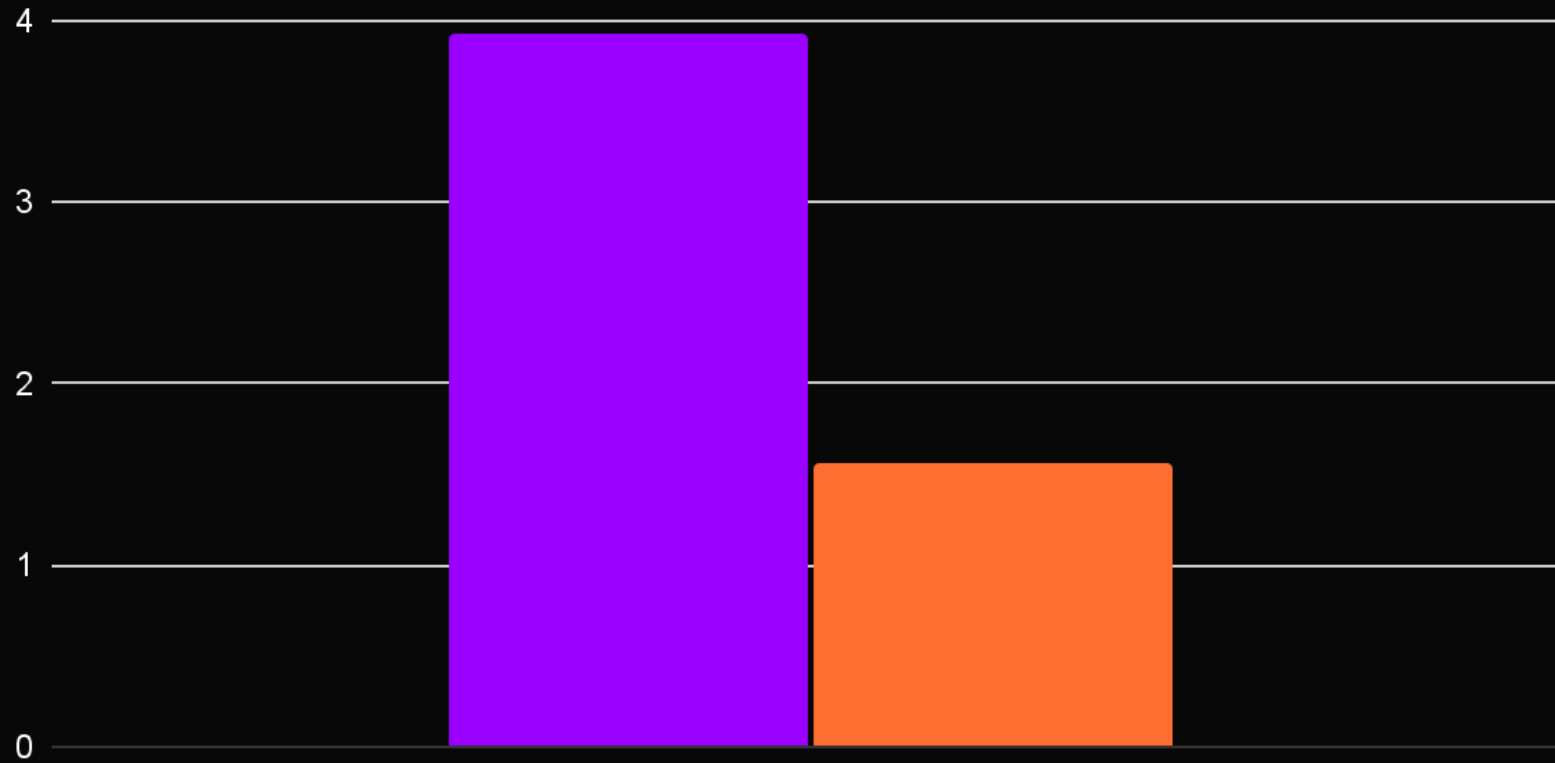
```
{
    byte[] buffer = ArrayPool<byte>.Shared.Rent(0x1000);
    while ((read = await _source.ReadAsync(buffer.AsMemory(totalBytes))) > 0)
    {
        // заполняем буфер, увеличивая размер буфера в 2 раза
    }

    _writer.WriteStartObject();
    _writer.WriteBase64String("data", buffer.AsSpan(0, totalBytes));
}
```

```
byte[] buffer = ArrayPool<byte>.Shared.Rent(0x1000);
while ((read = await _source.ReadAsync(buffer)) > 0)
{
    _writer.WriteBase64StringSegment(buffer.AsSpan(0, read), isFinalSegment: false);
}
_writer.WriteBase64StringSegment(default, isFinalSegment: true);
```

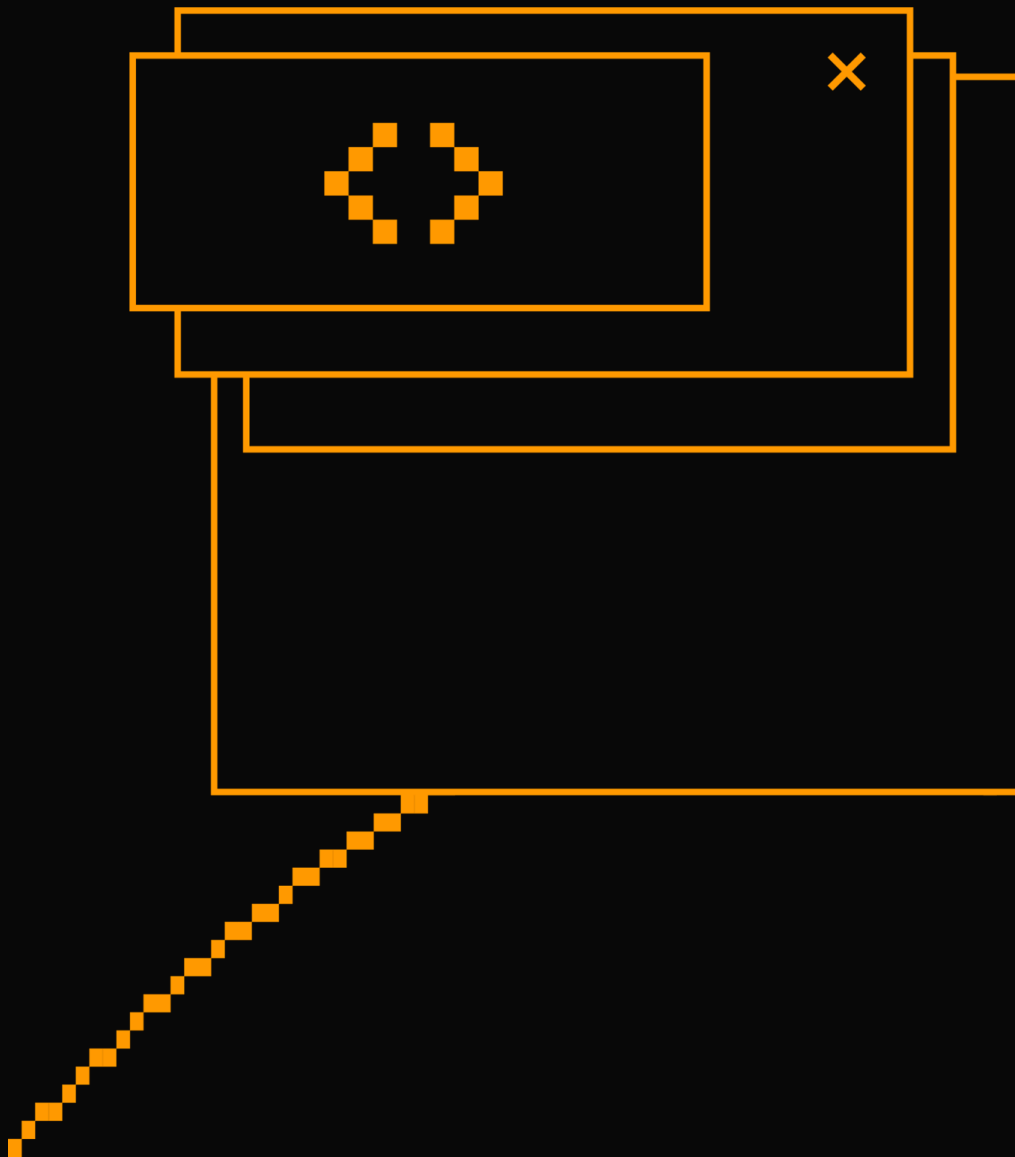
Mean (ms)

Buffered Streaming



~60%

**Вывод**



# Вывод

- System.Text.Json активно развивается
- Производительность растёт от версии к версии
- API становится более дружелюбным
- Разработчики Microsoft активно взаимодействуют с комьюнити

# Ресурсы

- <https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-10/#json>
- <https://jkrussell.dev/blog/system-text-json-vs-newtonsoft-json-benchmark/>
- <https://github.com/dotnet/runtime/issues/25087>
- <https://github.com/dotnet/runtime/issues/122548>
- <https://github.com/gragra33/Utf8JsonAsyncStreamReader>
- <https://github.com/mityaiii/SystemTextJsonBenchmarks>

Подписывайтесь  
на технологии в Контуре



Контур