Cognizant Softvision

# Quick Code Wednesday

# JSON

## What is JSON?

## WHAT IS JSON?

**JavaScript Object Notation (JSON)** is a lightweight data-interchange format based on the syntax of JavaScript objects.

It is a **text-based** language-independent format for representing structured object data for easy transmission or saving.

## Why JSON?

## WHY JSON?

Compared to its predecessor in server-client communication, **XML**, is much smaller, translating into faster data transfers, and better experiences.

**Human-friendly** easy to read and write with any text editor, and simultaneously **machine-friendly** easy to parse, build and with an excellent compression rate.

**JSON** has expressive syntax for representing: objects, numbers, booleans and arrays.

## How?

Cognizant Softvision

## JSON Syntax Rules

- Uses key/value pairs — { "fileType": "JSON" }
- Uses double quotes around KEY and VALUE
- Must use the specified **data types...**
- File type is ".json"  and MIME type is "Application/json"

## Data Types

**String**: unicode characters in double quotes (“ ”)

**Array**: Ordered list of **Object** or more values in ([ ])

**Object**: Unordered collection of key/value pairs in ({ })

**Number**: no difference between integer and floats

**Boolean**: *true* or *false*

**Empty value**: *null*

Cognizant Softvision

## Pitfalls...

Date and time

Comments

Functions, Undefined, Nan

Binaries

Cyclical or recurring graphs

Schemas

Cognizant Softvision

# JavaScript

## JSON.parse

**`JSON.parse()`** method parses a JSON string, constructing the *JavaScript value* or *object* described by the string.

```
JSON.parse('{}');                    // {}
JSON.parse('true');                  // true
JSON.parse('"foo"');                 // "foo"
JSON.parse('[1, 5, "false"]');       // [1, 5, "false"]
JSON.parse('null');                  // null
```

https://tc39.es/ecma262/#sec-json-object

## JSON.parse() does not allow trailing commas

```
JSON.parse('[1, 2, 3, 4, ]');
JSON.parse('{"foo" : 1, }');
```

## JSON.parse() does not allow single quotes

```
JSON.parse("{'foo': 1}");
```

Cognizant Softvision

## JSON.stringify

**`JSON.stringify()`** method converts a *JavaScript object or value* to a JSON string.

If *value* has a **toJSON()** method, it's responsible to define what data will be serialized.

*undefined*, *functions*, *infinity, NaN*, and *symbols* are not valid JSON values

=> changed to **null** or **{}**.

*Date* implement the **toJSON()** function by returning a **string**, the same as date.toISOString().

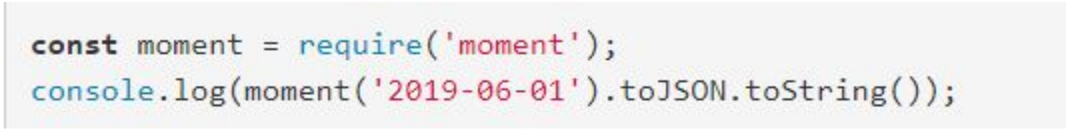
https://tc39.es/ecma262/#sec-json.stringify

## toJSON()

Many **Node.js** libraries and frameworks use `toJSON()` to ensure `JSON.stringify()` can serialize complex objects into something meaningful.

The `toJSON()` function is useful for making sure **ES6** classes get serialized correctly.

For example, Moment.js objects have a nice simple `toJSON()` function.



```
const moment = require('moment');
console.log(moment('2019-06-01').toJSON.toString());
```

Cognizant Softvision

# .Net core
# C#

## System.Text.Json

**`System.Text.Json`** provides the functionality for serializing and deserializing JSON. **`System.Text.Json.Serialization`** namespace contains attributes and APIs for advanced scenarios and customization specific to serialization and deserialization.

Built-in as part of the **.NET Core 3.0**

Also you can install the *System.Text.Json NuGet package* that supports:

- .NET Standard 2.0 and later versions
- .NET Framework 4.7.2 and later versions
- .NET Core 2.0, 2.1, and 2.2

https://www.nuget.org/packages/System.Text.Json
https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-overview

## SERIALIZATION BEHAVIOR

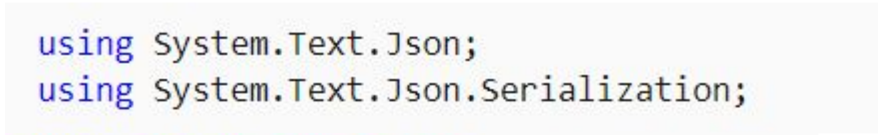
All public properties are serialized. You can decorate with `[JsonIgnore]`
Ignore read only properties `IgnoreReadOnlyProperties = true`
Ignore null values `IgnoreNullValues = true`

Non-ASCII characters, HTML-sensitive characters must be escaped (RFC 8259).

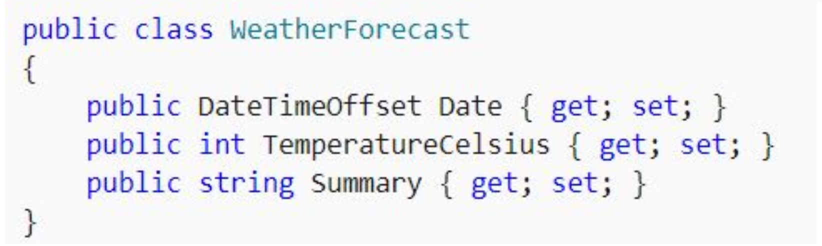
Minified, but pretty-print with `WriteIndented = true` option.

The casing of names matches the .Net names, maybe `PropertyNamingPolicy = JsonNamingPolicy.CamelCase` option.

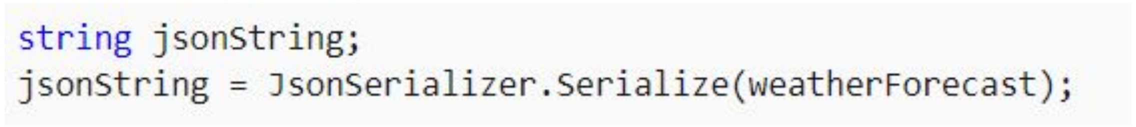
## C# - JsonSerializer.Serialize



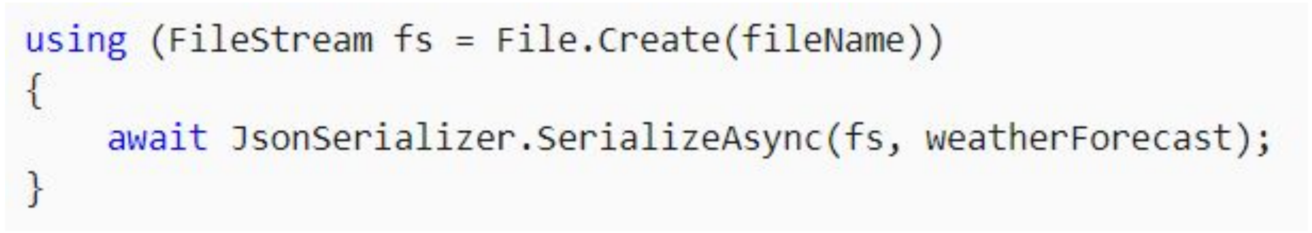
```
using System.Text.Json;
using System.Text.Json.Serialization;
```



```
public class WeatherForecast
{
    public DateTimeOffset Date { get; set; }
    public int TemperatureCelsius { get; set; }
    public string Summary { get; set; }
}
```



```
string jsonString;
jsonString = JsonSerializer.Serialize(weatherForecast);
```



```
using (FileStream fs = File.Create(fileName))
{
    await JsonSerializer.SerializeAsync(fs, weatherForecast);
}
```

## DESERIALIZATION BEHAVIOR

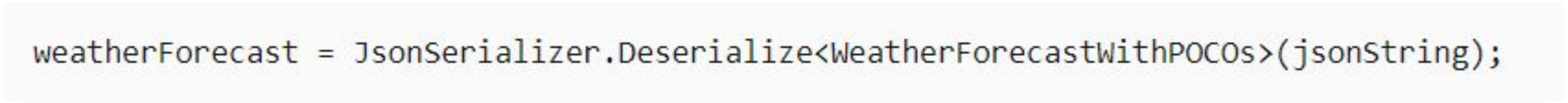
Property name matching is case-sensitive, or `PropertyNameCaseInsensitive = true,` and `[JsonPropertyName("NewName")]`

Any value for a read-only property is ignored and no exception is thrown.

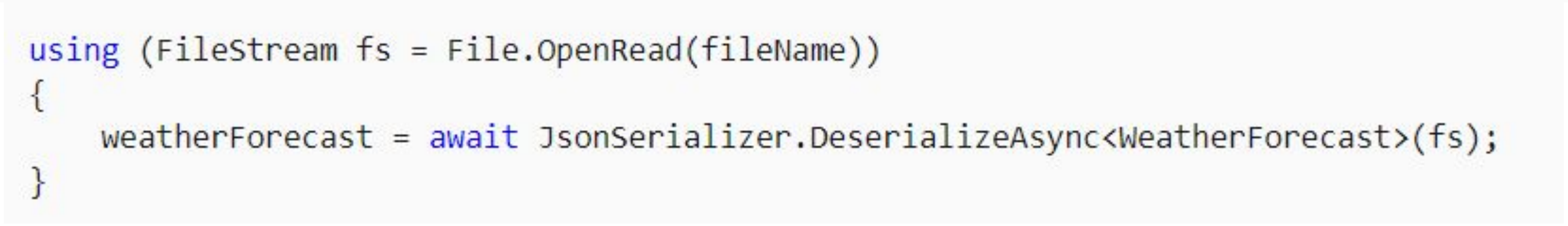
Comments and trailing commas in the JSON throw exceptions (RFC 8259)
You can do it with this options: `ReadCommentHandling = JsonCommentHandling.Skip,` `AllowTrailingCommas = true,`

The maximum depth allowed when reading JSON is 64 levels.

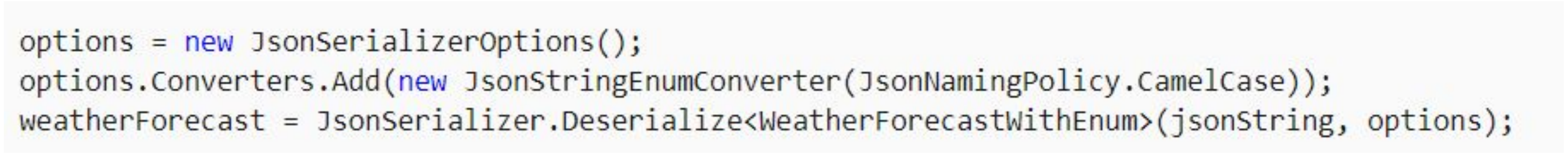
## C# - JsonSerializer.Deserialize



```
weatherForecast = JsonSerializer.Deserialize<WeatherForecastWithPOCOs>(jsonString);
```



```
using (FileStream fs = File.OpenRead(fileName))
{
    weatherForecast = await JsonSerializer.DeserializeAsync<WeatherForecast>(fs);
}
```



```
options = new JsonSerializerOptions();
options.Converters.Add(new JsonStringEnumConverter(JsonNamingPolicy.CamelCase));
weatherForecast = JsonSerializer.Deserialize<WeatherForecastWithEnum>(jsonString, options);
```

https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-how-to

## C# - DateTimeOffset

The DateTimeOffset defines the difference between the current instance's date and time and Coordinated Universal Time (UTC).

JSON serialize using both DateTime and DateTimeOffset.

https://docs.microsoft.com/en-us/dotnet/api/system.datetimeoffset?view=netcore-3.1
https://docs.microsoft.com/en-us/dotnet/standard/datetime/converting-between-datetime-and-offset

Cognizant Softvision

# REFERENCE LINKS

## LINKS

**Json.org (Douglas Crockford one page specification)**
https://www.json.org/

**RFC 8259 (2017-12)**
https://tools.ietf.org/html/rfc8259

**ECMA-404 (a pure coincidence number!)**
**(starts in 2013-10, last edition 2017-12)**
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

## LINKS

**JSON Web Signature (JWS)**
https://tools.ietf.org/html/rfc7515

**JSON Web Tokens (JWT)**
https://tools.ietf.org/html/rfc7519

**JWT.io**
**WARNING: You are exposing access tokens to the world when using online tools to analyze JWT tokens.**
https://jwt.io/

Cognizant Softvision

## THANK YOU

## MUCHAS GRACIAS