Telerik Academy

# Processing JSON in .NET

Databases

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

# Table of Contents

- The JSON data format
  - Rules and features
  - Usage
- JSON.NET Overview
  - Installation and usage
  - LINQ-to-JSON
  - JSON to XML and XML to JSON

Telerik Academy

# The JSON Data Format

## What is JSON?

Telerik Academy

# The JSON Data Format

- JSON (JavaScript Object Notation) is a lightweight data format
  - Human and machine-readable
  - Based on the way of creating objects in JS
  - Platform independent – can be used with any programming language

# JSON Rules

- The JSON data format follows the rules of object creation in JavaScript
  - Strings, numbers and Booleans are valid JSON

```
"this is string and is valid JSON"
```


  - Arrays are valid JSON

```
[5, 'string', true]
```

  - Objects are valid JSON

```
{
  "firstname": "Doncho",
  "lastname": "Minkov",
  "occupation": "Technical trainer"
}
```

# JSON Rules

- The JSON data format follows the rules of object creation in JavaScript
  - Strings, numbers and Booleans are valid JSON

```
"this is string and is valid JSON"
```


  - Arrays are valid JSON

```
[5, 'string', true]
```

  - Objects are valid JSON

```
{
  "firstname": "Doncho",
  "lastname": "Minkov",
  "occupation": "Technical trainer"
}
```

The double quotes for the keys are mandatory

Telerik Academy

# Processing JSON in .NET

## How to parse JSON in .NET?

Telerik Academy

# Built-in JSON Serializers

- .NET has built-in JSON serializer:
  - The `JavaScriptSerializer` class, contained in `System.Web.Extensions` assembly
- `JavaScriptSerializer` is able to parse C# object to JSON string and vice versa:

```
var place = new Place(…);
var serializer = new JavaScriptSerializer();

var jsonPlace =
          serializer.Serialize(place);
var objPlace =
          serializer.Deserialize<place>(jsonPlace);
```

Telerik Academy

# JavaScript Serializer

## Live Demo

Telerik Academy

# JavaScriptSerializer Features

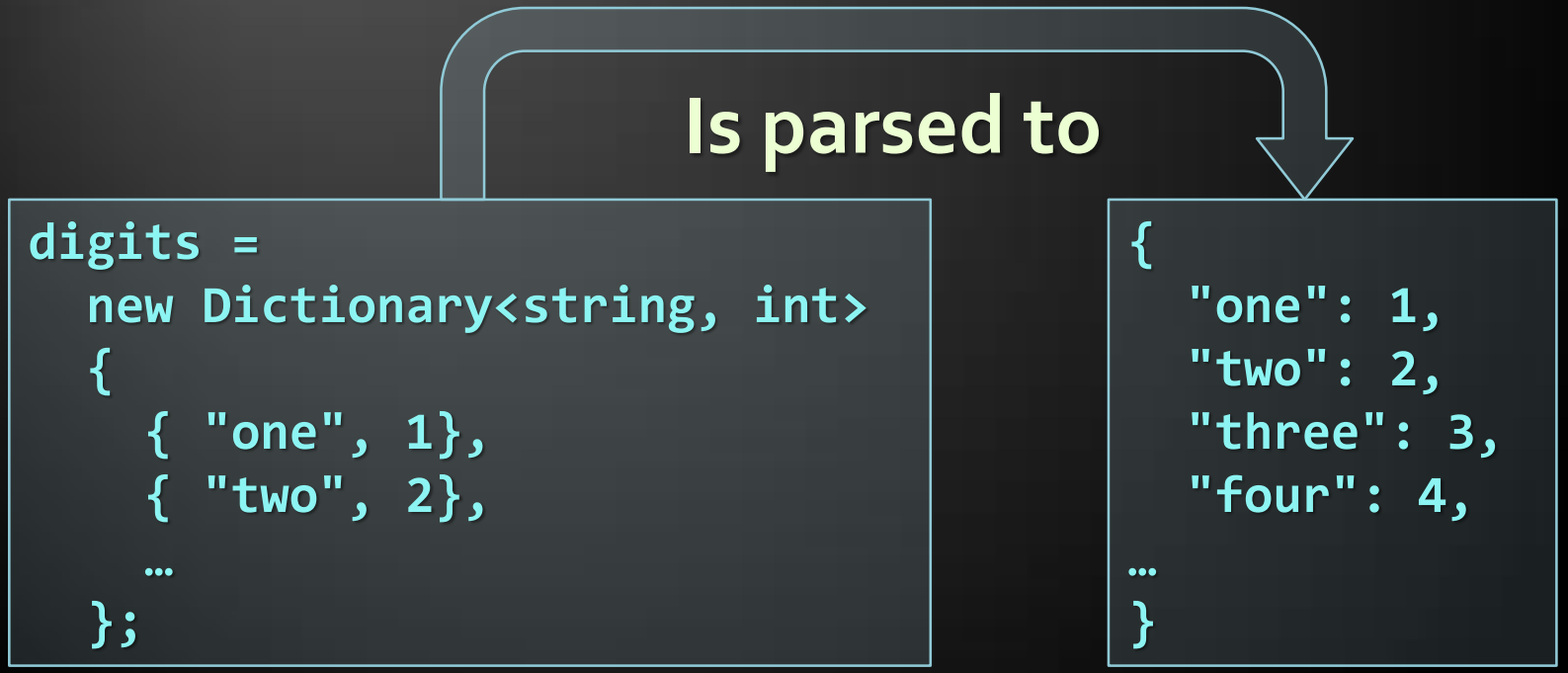
- JavaScript serializer has nice features:
  - Serializing objects to JSON and vice versa
  - Correct parsing of dictionaries:

```
digits =
  new Dictionary<string, int>
  {
    { "one", 1},
    { "two", 2},
    …
  };
```

Telerik Academy

# JavaScriptSerializer Features

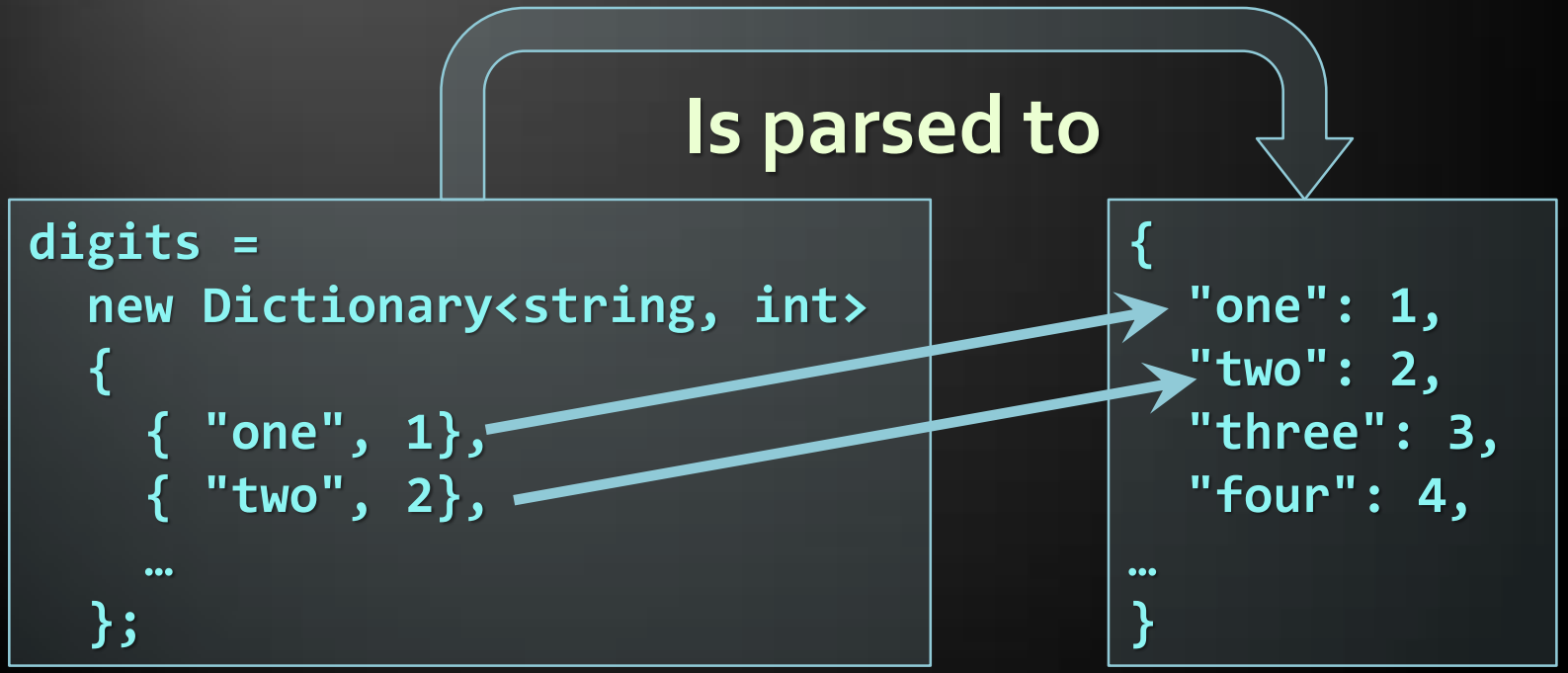
- JavaScript serializer has nice features:
  - Serializing objects to JSON and vice versa
  - Correct parsing of dictionaries:



Telerik Academy

# JavaScriptSerializer Features

- JavaScript serializer has nice features:
  - Serializing objects to JSON and vice versa
  - Correct parsing of dictionaries:



Telerik Academy

# JavaScript Serializer Features

Live Demo

Telerik Academy

# JSON.NET

## Better JSON parsing than with JavaScriptSerializer

# JSON.NET

- JavaScriptSerializer is good
  - But JSON.NET is better
- JSON.NET:
  - Has better performance
  - Supports LINQ-to-JSON
  - Has an out-of-the-box support for parsing between JSON and XML

Telerik Academy

# Installing JSON.NET

- To install JSON.NET, run in the Package Manager Console:

```
$ Install-Package Newtonsoft.Json
```

- Has two primary methods:
  - Serialize an object:

```
var serializedPlace =
   JsonConvert.SerializeObject(place);
```

  - Deserialize an object:

```
var deserializedPlace =
   JsonConvert.DeserializeObject<Place>(serializedPlace);
```

Telerik Academy

# Serializing and Deserializing Objects

Live Demo

Telerik Academy

# JSON.NET Features

- JSON.NET can be configured to:
  - Indent the output JSON string
  - To convert JSON to anonymous types
  - To control the casing and properties to parse
  - To skip errors
- JSON.NET also supports:
  - LINQ-to-JSON
  - Direct parse between XML and JSON

Telerik Academy

# Configuring JSON.NET

- To indent the output string use `Formattingg.Indented`:

```
JsonConvert.SerializeObject(place, Formatting.Indented);
```

- Deserializing to anonymous types:

```
var json = @"{ 'firstName': 'Doncho',"
               'lastName': 'Minkov',
               'occupation': 'Technical Trainer' }";
var template = new
{
  FirstName = "",
  Lastname = "",
  Occupation = ""
};
var person =
   JsonConvert.DeserializeAnonymousType(json, template);
```

Telerik Academy

# Configuring JSON.NET

- To indent the output string use `Formattingg.Indented`:

```
JsonConvert.SerializeObject(place, Formatting.Indented);
```

- Deserializing to anonymous types:

```
var json = @"{ 'firstName': 'Doncho',"
               'lastName': 'Minkov',
               'occupation': 'Technical Trainer' }";
var template = new
{
  FirstName = "",
  Lastname = "",
  Occupation = ""
};
var person =
   JsonConvert.DeserializeAnonymousType(json, template);
```

Should provide a template

Telerik Academy

# Configuring JSON.NET

## Live Demo

Telerik Academy

# JSON.NET Parsing of Objects

- By default JSON.NET takes each Property/Field from the public interface of a class and parses it
  - This can be controlled using attributes:

```
public class User
{
  [JsonProperty("user")]
  public string Username{get;set;}

  [JsonIgnore]
  public string Password{get;set;}
}
```

Telerik Academy

# JSON.NET Parsing of Objects

- By default JSON.NET takes each Property/Field from the public interface of a class and parses it
  - This can be controlled using attributes:

```
public class User
{
  [JsonProperty("user")]
  public string Username{get;set;}

  [JsonIgnore]
  public string Password{get;set;}
}
```

Tells the converter to parse "Username" to "user"

Telerik Academy

# JSON.NET Parsing of Objects

- By default JSON.NET takes each Property/Field from the public interface of a class and parses it as is
  - This can be controlled using attributes:

```
public class User
{
  [JsonProperty("user")]
  public string Username{get;set;}

  [JsonIgnore]
  public string Password{get;set;}
}
```

Tells the converter to parse "Username" to "user"

Tells the converter to skip the property "Password"

Telerik Academy

# JSON.NET Parsing of Objects

Live Demo

# LINQ-to-JSON

- JSON.NET has a support for LINQ-to-JSON

```
var jsonObj = JObject.Parse(json);
Console.WriteLine("Places in {0}:", jsonObj["name"]);

jsonObj["places"].Select(
    pl =>
        string.Format("{0}) {1} ({2})",
                      index++,
                      pl["name"],
                      string.Join(", ",
                                  pl["categories"].Select(
                                    cat => cat["name"]))))
                  .Print();
```

Telerik Academy

# LINQ-to-JSON

## Live Demo

Telerik Academy

# XML to JSON and JSON to XML

Telerik Academy

# XML to JSON and JSON to XML

- Conversions from JSON to XML are done using two methods:
  - XML to JSON

```
string jsonFromXml = JsonConvert.SerializeXNode(doc);
```

  - JSON to XML

```
XDocument xmlFromJson = JsonConvert.DeserializeXNode(json);
```

Telerik Academy

# XML to JSON and JSON to XML

Live Demo

Telerik Academy

# Processing JSON in .NET


http://academy.telerik.com