JSON

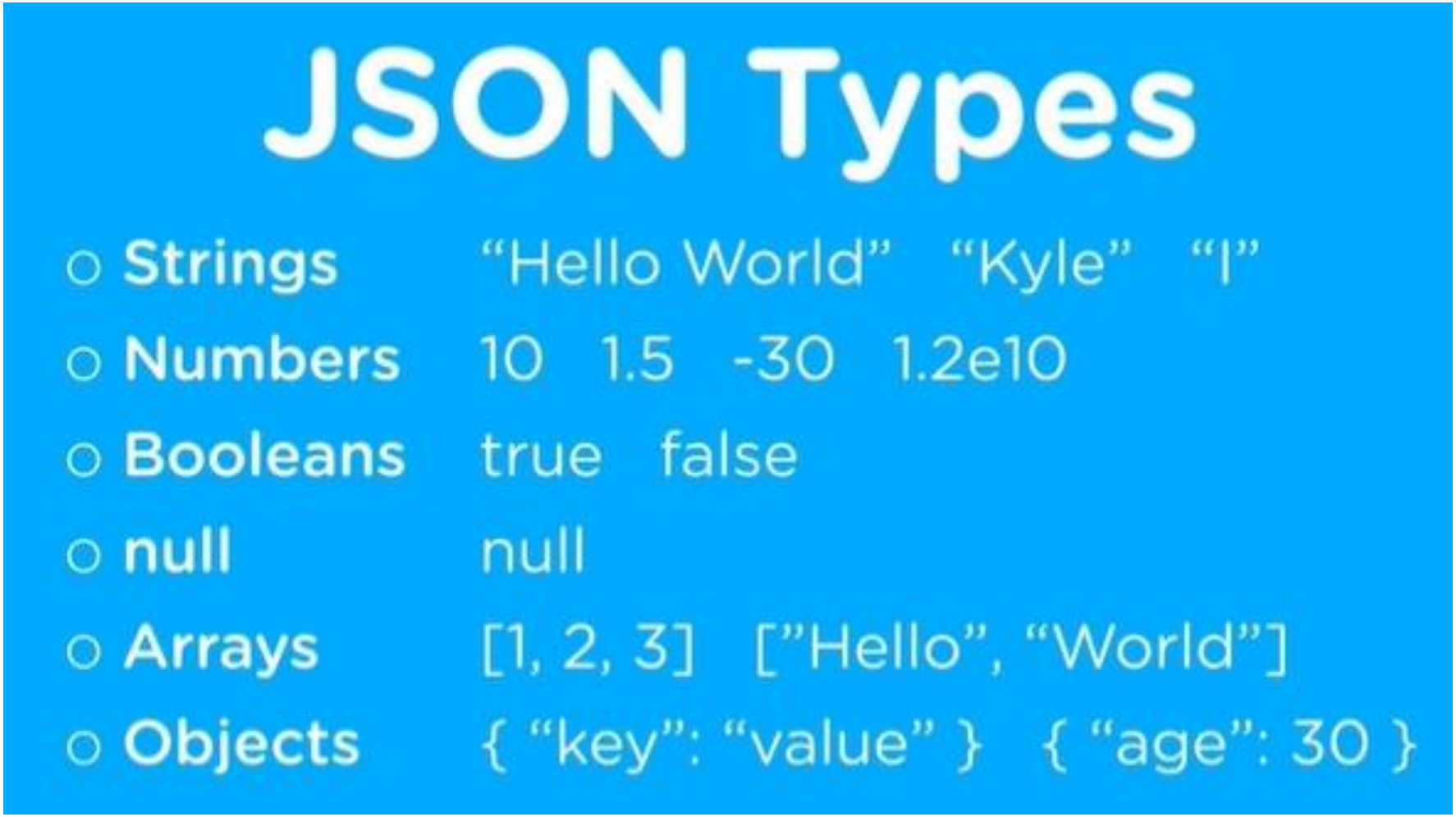
JavaScript Object Notation

Grundlagen

Überblick

Quelle: https://www.youtube.com/watch?v=iiADhChRriM

JSON Datentypen



JSON Dokumente

JSON Types

- **Strings** "Hello World" "Kyle" "I"
- **Numbers** 10 1.5 -30 1.2e10
- **Booleans** true false
- **null** null
- **Arrays** [1, 2, 3] ["Hello", "World"]
- **Objects** { "key": "value" } { "age": 30 }

user.json

```
123
```

user.json

```
{
  "name": "Kyle",
  "favoriteNumber": 3,
  "isProgrammer": true,
  "hobbies": ["Weight Lifting", "Bowling"],
}
```

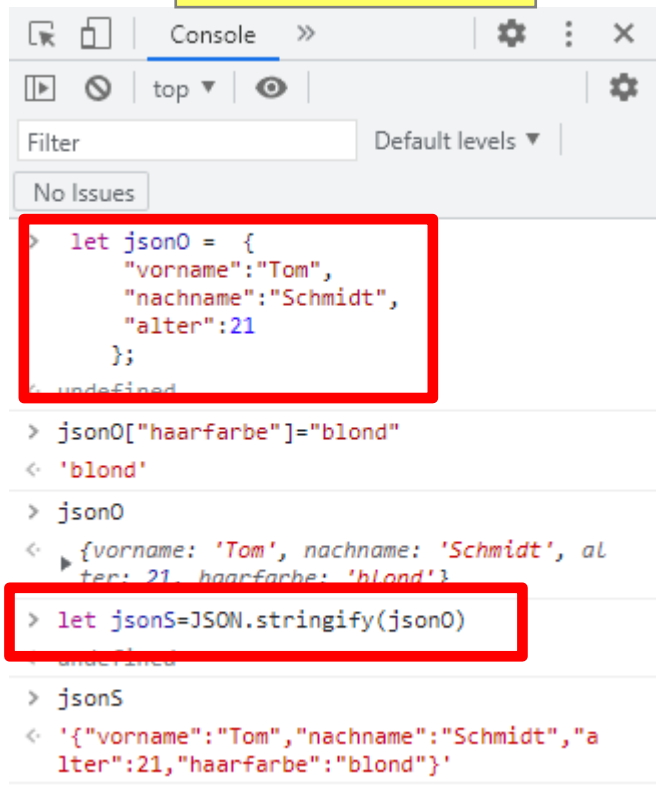
user.json

```
{
  "name": "Kyle",
  "favoriteNumber": 3,
  "isProgrammer": true,
  "hobbies": ["Weight Lifting", "Bowling"],
  "friends": [{
    "name": "Joey",
    "favoriteNumber": 100,
    "isProgrammer": false,
    "friends": [...]
  }]
}
```

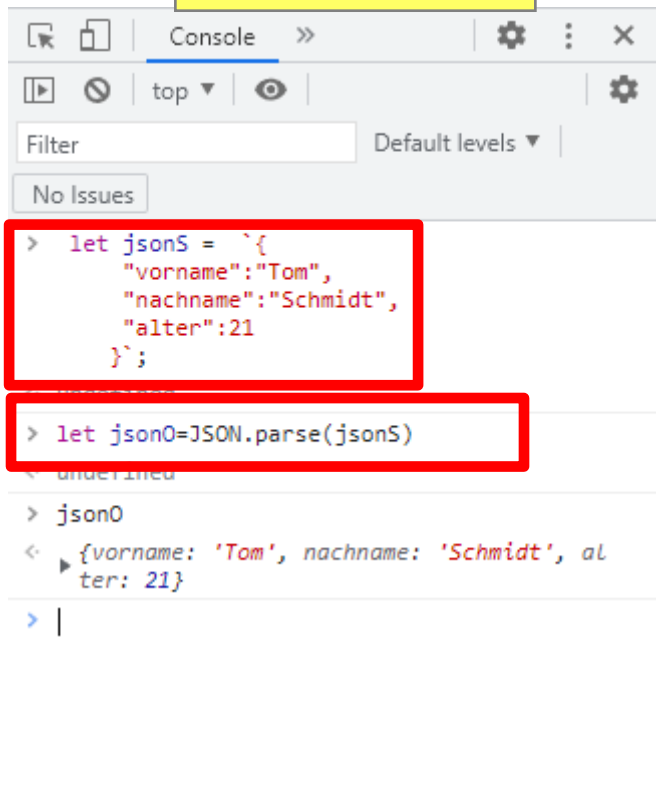
JSON in JavaScript

JSON in JavaScript

Objekt

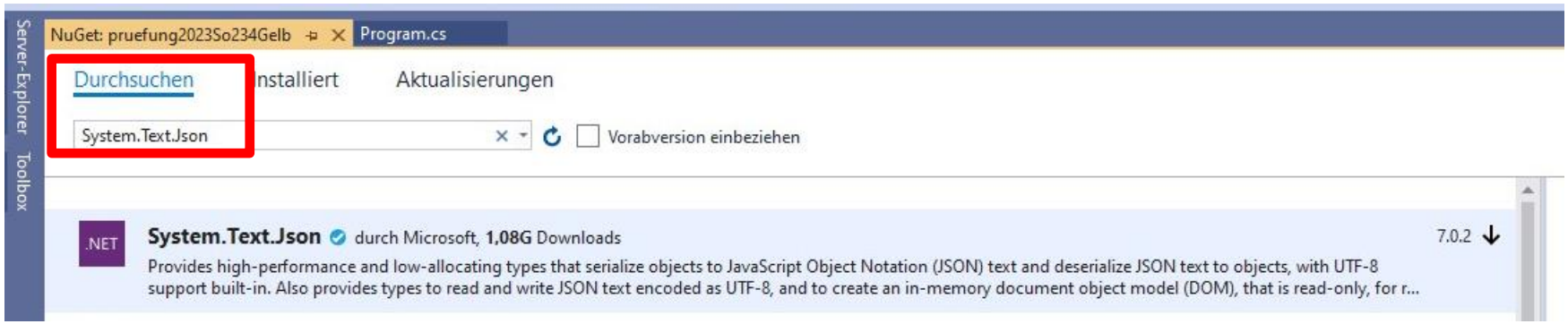


String



JSON in C#

Package System.Text.Json installieren (VS 2019)



JSON Document erstellen / parsen

```
string data = @" [ {""vorname"": ""Tom"", ""Nachname"": ""Schmidt"", ""alter"":21},
                   {""vorname"": ""Tina"", ""Nachname"": ""Schlau"", ""alter"":18}
                 ]";

JsonDocument doc = JsonDocument.Parse(data);
JsonElement root = doc.RootElement;

var u1 = root[0];
var u2 = root[1];
Console.WriteLine(u1);
Console.WriteLine(u2);

Console.WriteLine(u1.GetProperty("vorname"));
Console.WriteLine(u1.GetProperty("alter"));

Console.WriteLine(u2.GetProperty("vorname"));
Console.WriteLine(u2.GetProperty("alter"));
```

JSON durch Arrays iterieren

```
var users = root.EnumerateArray();

while (users.MoveNext()) {
    var user = users.Current;
    System.Console.WriteLine(user);

    var props = user.EnumerateObject();

    while (props.MoveNext()) {
        var prop = props.Current;
        Console.WriteLine($"{prop.Name}: {prop.Value}");
    }
}
```

C# Objekte in JSON serialisieren

```
class User {
    2 Verweise
    public string vorname { get; set; }
    2 Verweise
    public string nachname { get; set; }
    2 Verweise
    public MyDate geburtstag { get; set; }
    1 Verweis
    public User(string vorname, string nachname, MyDate geburtstag)
        this.vorname = vorname;
        this.nachname = nachname;
        this.geburtstag = geburtstag;
    }
    0 Verweise
    public override string ToString() {
        return $"{vorname},{ nachname},{ geburtstag}";
    }
}
```

```
class MyDate {
    2 Verweise
    public int year { get; set; }
    2 Verweise
    public int month { get; set; }
    2 Verweise
    public int day { get; set; }
    1 Verweis
    public MyDate(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
    0 Verweise
    public override string ToString() {
        return $"{year}/{month}/{day}";
    }
}
```

```
User user1 = new User("Tom", "Schmidt",new MyDate(2002, 09, 15));
Console.WriteLine(user1);

var options = new JsonSerializerOptions { WriteIndented = true };
string jsonString = JsonSerializer.Serialize(user1,options);
Console.WriteLine(jsonString);

string fileName = @"../../user.json";
File.WriteAllText(fileName, jsonString);
```

C# Objekte aus JSON deserialisieren

```
class User {
    2 Verweise
    public string vorname { get; set; }
    2 Verweise
    public string nachname { get; set; }
    2 Verweise
    public MyDate geburtstag { get; set; }
    1 Verweis
    public User(string vorname, string nachname, MyDate geburtstag)
        this.vorname = vorname;
        this.nachname = nachname;
        this.geburtstag = geburtstag;
    }
    0 Verweise
    public override string ToString() {
        return $"{vorname},{ nachname},{ geburtstag}";
    }
}
```

```
class MyDate {
    2 Verweise
    public int year { get; set; }
    2 Verweise
    public int month { get; set; }
    2 Verweise
    public int day { get; set; }
    1 Verweis
    public MyDate(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
    0 Verweise
    public override string ToString() {
        return $"{year}/{month}/{day}";
    }
}
```

```
jsonString = File.ReadAllText(fileName);

var userResurrected = JsonSerializer.Deserialize<User>(jsonString);

Console.WriteLine(userResurrected);
```

C# JSON asynchron einlesen, mit Objekten

```
var httpClient = new HttpClient();

var url = "https://raw.githubusercontent.com/dotnet/core/master/release-notes/releases-index.json";
var ts = await httpClient.GetStreamAsync(url);

var resp = await JsonDocument.ParseAsync(ts);

var root1 = resp.RootElement.GetProperty("releases-index");

var elems1 = root1.EnumerateArray();

while (elems1.MoveNext()) {
    var node = elems1.Current;
    Console.WriteLine(node);
}
```

C# JSON HTTPRequestMessage

```
static async void translate(string whatToTranslate)
{
    var client = new HttpClient();
    var request = new HttpRequestMessage
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri("https://google-translate1.p.rapidapi.com/language/translate/v2"),
        Headers =
        {
            //Key eindeutig nach Anmeldung bei RapidAPI
            { "X-RapidAPI-Key", "60d5ad015fmsh8725bdbf7c3ec14p1291f9jsn8bd29cfd7653" },
            { "X-RapidAPI-Host", "google-translate1.p.rapidapi.com" },
        },
        Content = new FormUrlEncodedContent(new Dictionary<string, string>
        {
            { "source", "en" },
            { "target", "de" },
            { "q", $"{whatToTranslate}" },
        }),
    };
    using (var response = await client.SendAsync(request))
    {
        response.EnsureSuccessStatusCode();
        var t = await response.Content.ReadAsStringAsync();
        Console.WriteLine(t);
    }
    Console.ReadKey();
}
```

C# JSON HTTPRequestMessage mit Objekten

```
static async void translate(string whatToTranslate)
{
    var client = new HttpClient();
    var request = new HttpRequestMessage
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri("https://google-translate1.p.rapidapi.com/language/translate/v2"),
        Headers =
        {
            //Key eindeutig nach Anmeldung bei RapidAPI
            { "X-RapidAPI-Key", "...d5ad015fmsh8725bdbf7c3ec14p1291f9jsn8bd29cfd7653" },
            { "X-RapidAPI-Host", "google-translate1.p.rapidapi.com" },
        },
        Content = new FormUrlEncodedContent(new Dictionary<string, string>
        {
            { "source", "en" },
            { "target", "de" },
            { "q", $"{whatToTranslate}" },
        }),
    };
    using (var response = await client.SendAsync(request))
    {
        response.EnsureSuccessStatusCode();
        var t = await response.Content.ReadAsStringAsync();
        var tm = JsonSerializer.Deserialize<TranslationResponse>(t);
        for (int i =0; i< tm.data.translations.Count; i++)
        {
            Console.WriteLine(tm.data.translations[i].translatedText );
        }
        Console.ReadKey();
    }
}
```

C# JSON HTTPRequestMessage mit Objekten Hilfsklassen

```
What would you like to translate?
How are you?

{"data":
  {"translations":[
                   {"translatedText":"Wie geht es dir?"}
                  ]
  }
}
```

```
public class TranslationResponse
{
    2 Verweise
    public TranslationData data { get; set; }
}
```

```
public class TranslationData
{
    2 Verweise
    public List<Translation> translations {get;set;}
}
```

```
public class Translation
{
    1 Verweis
    public string translatedText { get; set; }
}
```