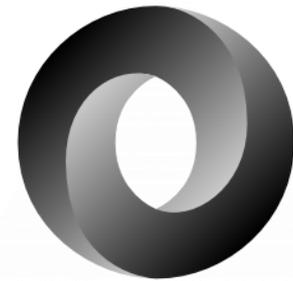


SWE 363: Web Engineering & Development

Module 8-2

**JavaScript Object
Notation**



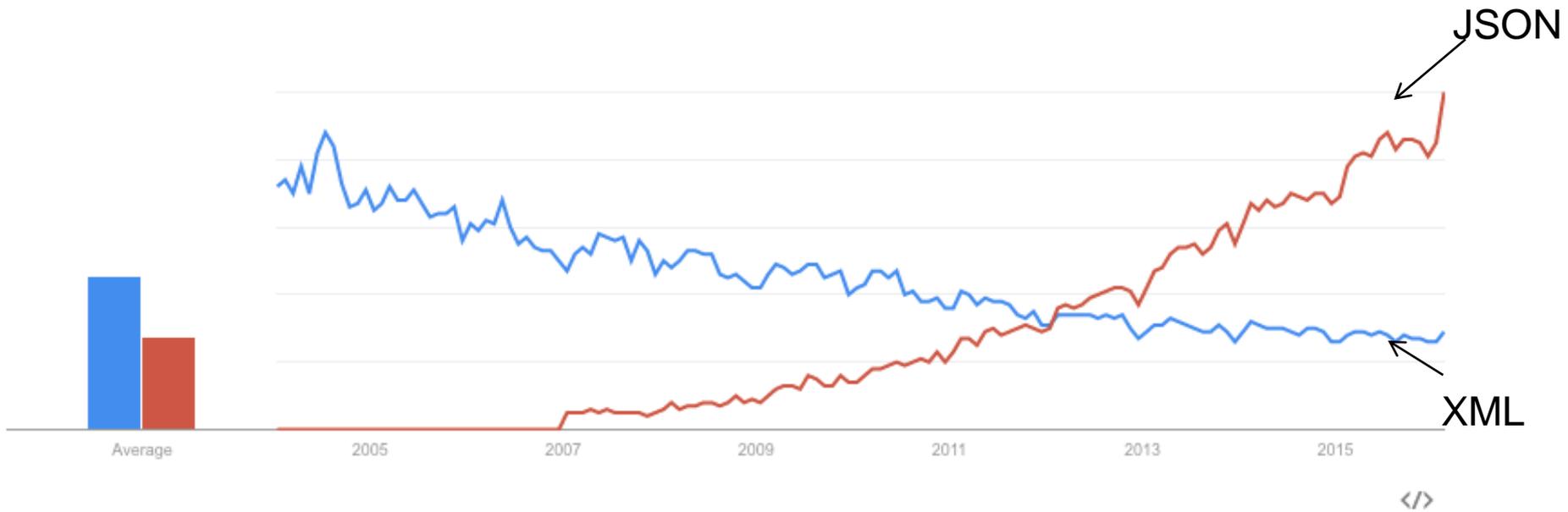
{JSON}

Objectives

- ❑ Learn the basics of JSON

- ❑ What is JSON?
- ❑ How JSON differs from XML
- ❑ Using JSON in JavaScript

Trends in XML and JSON usage



As a professional, it would be better have expertise in both XML and JSON

What is JSON?

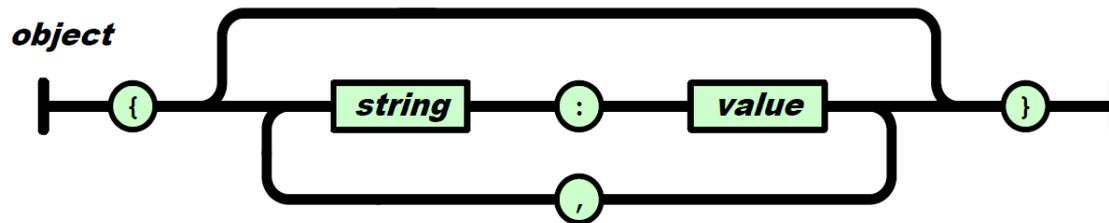
- ❑ JSON (JavaScript Object Notation) is a **lightweight alternative to XML** for data-interchange
- ❑ JSON code is **valid JavaScript**
- ❑ JSON is built on **two structures**:
 - A collection of name/value pairs
 - An ordered list of values
- ❑ JSON file extension: ***.json**

- ❑ JSON - represents **object data in a text format** so that it can be transmitted from one computer to another.

- ❑ Many **REST web services** encode their **returned data in the JSON data format** instead of XML.
 - It provides a more concise format than XML to represent data.

JSON object

- ❑ A JSON **object** is zero or more string-colon-value pairs
- ❑ An **object** is an unordered set of name/value pairs
 - The pairs are enclosed within **braces**, { }
 - There is a **colon** between the name and the value
 - Pairs are separated by **commas**

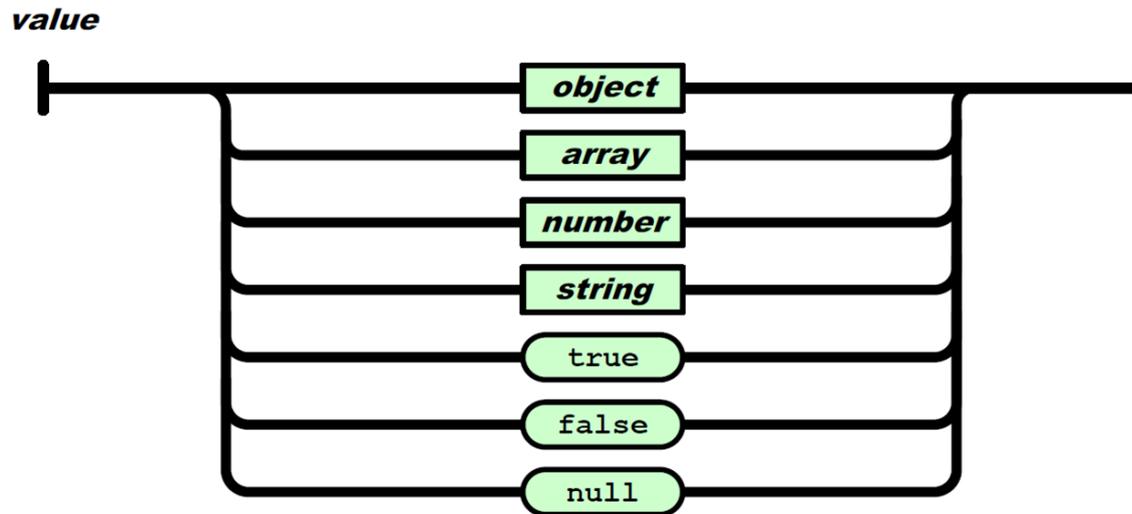


```
{  
  "name": "John Doe",  
  "age": 30,  
  "married": true  
}
```

Example of a JSON object:

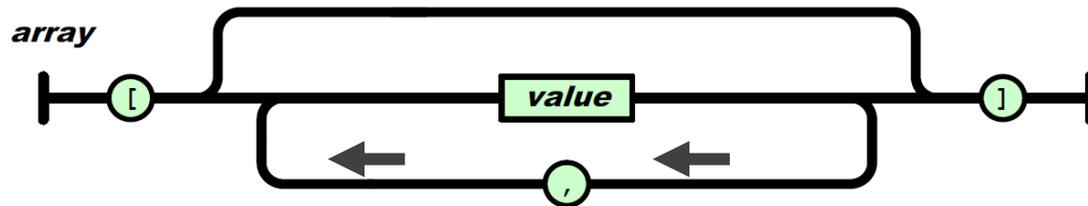
JSON value

- ❑ A JSON **instance contains a single JSON value**.
- ❑ A JSON value may be either an object, array, number, string, true, false, or null:



JSON array

- ❑ A JSON **array** is used to express a list of values.
- ❑ A JSON array contains zero or more values, separated by comma and wrapped within square brackets:



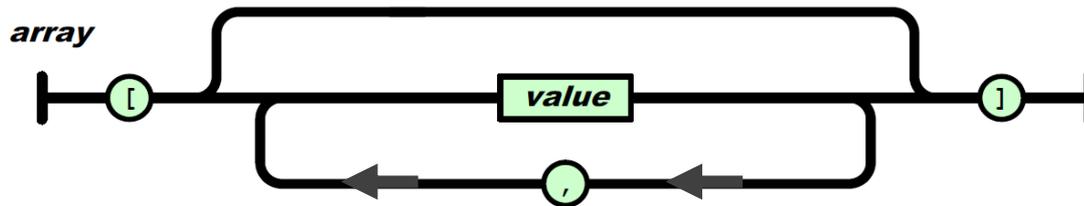
```
{ "name": "John Doe",  
  "age": 30,  
  "married": true,  
  "siblings": ["John", "Mary", "Pat"] }
```

Example of a JSON array



Array of objects

- ❑ Each item in an array may be any of the seven JSON values.



```
{  
  "name": "John Doe",  
  "age": 30,  
  "married": true,  
  "siblings": [  
    {"name": "John", "age": 25},  
    true,  
    "Hello World"  
  ]  
}
```

The array contains 3 items. The first item is an object, the second item is a boolean, and the third item is a string.

No multiline strings

- ❑ JSON does not allow multiline strings.

Legal:

```
{  
  "comment": "This is a very, very long comment"  
}
```

Not legal:

```
{  
  "comment": "This is a very,  
  very long comment"  
}
```

Comments not allowed!

- ❑ You **cannot comment** a JSON instance document.
- ❑ There is no syntax for commenting JSON instances.

Comparison of JSON and XML

□ Similarities:

- Both are **human readable**
- Both have very **simple syntax**
- Both are **hierarchical**
- Both are **language independent**
- Both can be used by **Ajax**
- Both supported in **APIs** of many programming languages

□ Differences:

- XML is a **markup language** much like HTML
- **Syntax** is different
- JSON is **less verbose**
- JSON includes **arrays**
- **No attributes** in JSON
- **No namespaces** in JSON

Stop Comparing JSON and XML

<http://www.yegor256.com/2015/11/16/json-vs-xml.html>

XML and JSON, side-by-side

- XML/ JSON are ways of structuring data

```
<Book>
  <Title>Parsing Techniques</Title>
  <Authors>
    <Author>Dick Grune</Author>
    <Author>Ceriél J.H. Jacobs</Author>
  </Authors>
  <Date>2007</Date>
  <Publisher>Springer</Publisher>
</Book>
```

```
{
  "Book":
    {
      "Title": "Parsing Techniques",
      "Authors": [ "Dick Grune", "Ceriél J.H. Jacobs" ],
      "Date": "2007",
      "Publisher": "Springer"
    }
}
```

XML and JSON, side-by-side

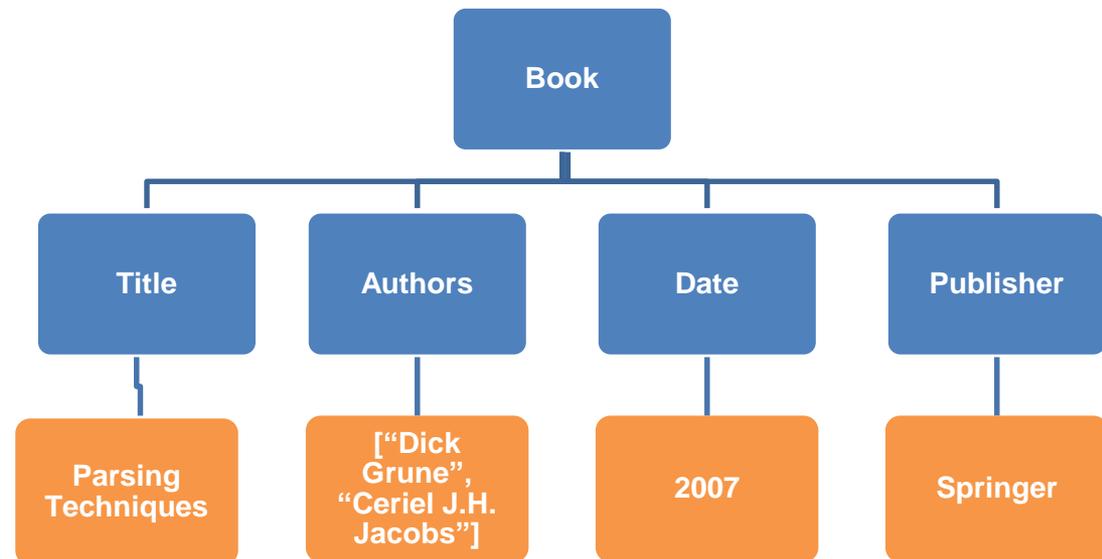
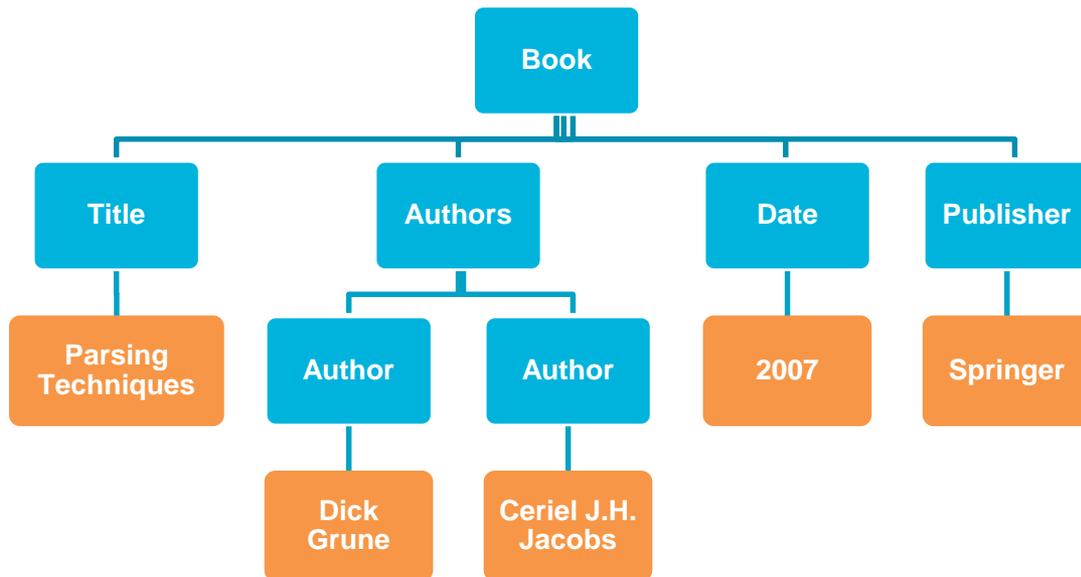
```
<Book>
  <Title>Parsing Techniques</Title>
  <Authors>
    <Author>Dick Grune</Author>
    <Author>Ceriél J.H. Jacobs</Author>
  </Authors>
  <Date>2007</Date>
  <Publisher>Springer</Publisher>
</Book>
```

Creating lists in XML
and JSON

```
{
  "Book":
  {
    "Title": "Parsing Techniques",
    "Authors": [ "Dick Grune", "Ceriél J.H. Jacobs" ],
    "Date": "2007",
    "Publisher": "Springer"
  }
}
```

Represented as a tree

An XML document is a tree



A JSON Object is a tree

Trees are well-studied

- ❑ The **tree data structure** has been well-studied by computer scientists and mathematicians.
- ❑ There are many **well-known algorithms** for processing and traversing trees.
- ❑ Both XML and JSON are able to leverage this.

JSON supports Complex structures

- ❑ Using JSON you can define **arbitrarily complex structures**

```
{
  "Book":
  {
    "Title": "Parsing Techniques",
    "Authors": [ "Dick Grune", "Ceriél J.H. Jacobs" ]
  }
}
```

```
{
  "Book":
  {
    "Title": "Parsing Techniques",
    "Authors": [
      {"name": "Dick Grune", "university": "Vrije Universiteit"},
      {"name": "Ceriél J.H. Jacobs", "university": "Vrije Universiteit"}
    ]
  }
}
```

XML Schema for Book

```
<xs:element name="Book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Title" type="xs:string" />
      <xs:element name="Authors">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Author" type="xs:string" maxOccurs="5"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Date" type="xs:gYear" />
      <xs:element name="Publisher" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Springer" />
            <xs:enumeration value="MIT Press" />
            <xs:enumeration value="Harvard Press" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Equivalent JSON Schema

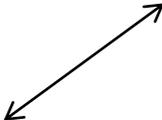
JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents.

```
{
  "$schema": http://json-schema.org/draft-04/schema,
  "type": "object",
  "properties": {
    "Book": {
      "type": "object",
      "properties": {
        "Title": {"type": "string"},
        "Authors": {"type": "array", "minItems": 1, "maxItems": 5, "items": {"type": "string"}},
        "Date": {"type": "string", "pattern": "^[0-9]{4}$"},
        "Publisher": {"type": "string", "enum": ["Springer", "MIT Press", "Harvard Press"]}
      },
      "required": ["Title", "Authors", "Date"],
      "additionalProperties": false
    }
  },
  "required": ["Book"],
  "additionalProperties": false
}
```

Title with string type

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "Book": {
      "type": "object",
      "properties": {
        "Title": {"type": "string"},
        "Authors": {"type": "array", "minItems": 1, "maxItems": 5, "items": {"type": "string"}},
        "Date": {"type": "string", "pattern": "^[0-9]{4}$"},
        "Publisher": {"type": "string", "enum": ["Springer", "MIT Press", "Harvard Press"]}
      },
      "required": ["Title", "Authors", "Date"],
      "additionalProperties": false
    }
  },
  "required": ["Book"],
  "additionalProperties": false
}
```

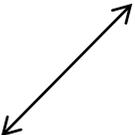
`<xs:element name="Title" type="xs:string" />`



Authors list

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "Book": {
      "type": "object",
      "properties": {
        "Title": {"type": "string"},
        "Authors": {"type": "array", "minItems": 1, "maxItems": 5, "items": {"type": "string"}},
        "Date": {"type": "string", "pattern": "^[0-9]{4}$"},
        "Publisher": {"type": "string", "enum": ["Springer", "MIT Press", "Harvard Press"]}
      },
      "required": ["Title", "Authors", "Date"],
      "additionalProperties": false
    }
  },
  "required": ["Book"],
  "additionalProperties": false
}
```

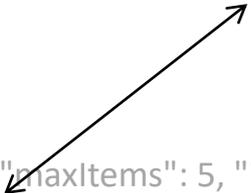
```
<xs:element name="Authors">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Author" type="xs:string" maxOccurs="5"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



Date with year type

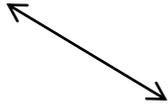
```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "Book": {
      "type": "object",
      "properties": {
        "Title": {"type": "string"},
        "Authors": {"type": "array", "minItems": 1, "maxItems": 5, "items": {"type": "string"}},
        "Date": {"type": "string", "pattern": "^[0-9]{4}$"},
        "Publisher": {"type": "string", "enum": ["Springer", "MIT Press", "Harvard Press"]}
      },
      "required": ["Title", "Authors", "Date"],
      "additionalProperties": false
    }
  },
  "required": ["Book"],
  "additionalProperties": false
}
```

`<xs:element name="Date" type="xs:gYear" />`



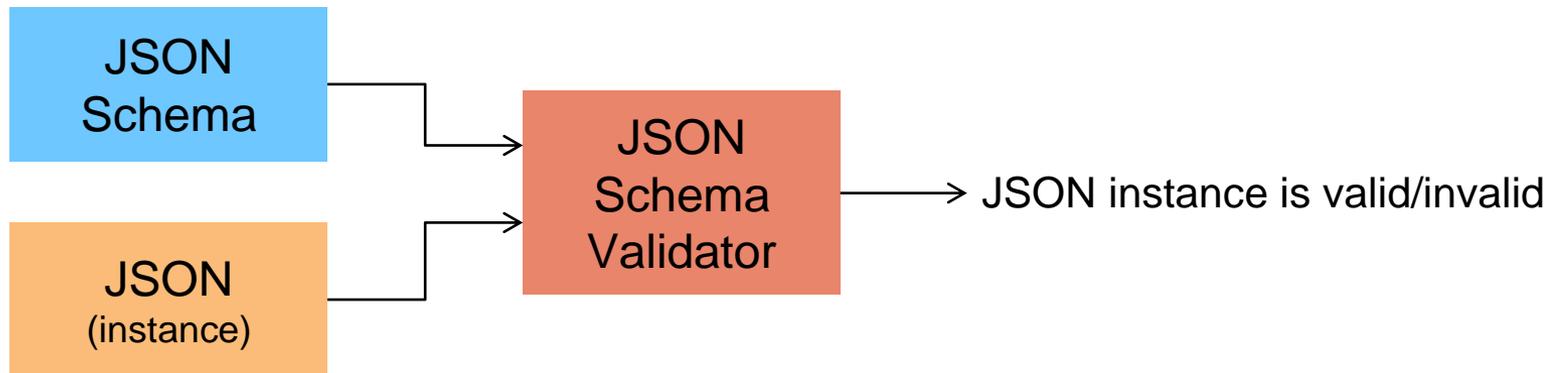
Publisher with enumeration

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "Book": {
      "type": "object",
      "properties": {
        "Title": {"type": "string"},
        "Authors": {"type": "array", "minItems": 1, "maxItems": 5, "items": {"type": "string"}},
        "Date": {"type": "string", "pattern": "^[0-9]{4}$"},
        "Publisher": {"type": "string", "enum": ["Springer", "MIT Press", "Harvard Press"]}
      },
      "required": ["Title", "Authors", "Date"],
      "additionalProperties": false
    }
  },
  "required": ["Book"],
  "additionalProperties": false
}
```



```
<xs:element name="Publisher" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Springer" />
      <xs:enumeration value="MIT Press" />
      <xs:enumeration value="Harvard Press" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Validate JSON docs against JSON Schema



Online JSON Schema validator

<http://json-schema-validator.herokuapp.com/index.jsp>

Schema:

Paste your JSON Schema in here **1**

Data:

Paste your JSON in here **2**

Validate [\(load sample data\)](#)

Click on the validate button **3**

Validation results:

Results of validation is shown here **4**

JSON Schema validators

- ❑ To validate the syntax of JSON object against given Schema

<http://json-schema.org/implementations.html>

- ❑ To validate the syntax of JSON object without Schema

<https://jsonlint.com/>

Using JSON in JavaScript

- ❑ The JSON objects are programmatically constructed or downloaded from an external web service.
- ❑ The JSON information will be contained within a string.
 - >> **JSON.parse()** function is used to transform the string containing the JSON data into a JavaScript object:

```
var text = '{"artist": {"name": "Manet", "nationality": "France"}}';  
var a = JSON.parse(text);  
alert(a.artist.nationality);
```

- ❑ The jQuery library also provides a **JSON parser** that will work with all browsers

```
var artist = jQuery.parseJSON(text);
```

- (the **JSON.parse()** function is not available on older browsers)

Using JSON in PHP

- ❑ The JSON extension is **bundled** and **compiled** into PHP
- ❑ Converting a JSON string into a PHP object is straightforward:

```
<?php
// convert JSON string into PHP object
$text = '{"artist": {"name":"Manet","nationality":"France"}}';
$obj = json_decode($text);
echo $obj->artist->nationality;

// convert JSON string into PHP associative array
$arr = json_decode($text, true);
echo $arr['artist']['nationality'];
?>
```

- the `json_decode()` function can return either a PHP object or an associative array.

Using JSON in PHP

- ❑ Since JSON data is often coming from an external source, one should always check for parse errors before using it, which can be done via the `json_last_error()` function:

```
<?php
// convert JSON string into PHP object
$text = '{"artist": {"name":"Manet","nationality":"France"}}';
$anObject = json_decode($text);
// check for parse errors
if (json_last_error() == JSON_ERROR_NONE) {
    echo $anObject->artist->nationality;
}
?>
```

- ❑ To go the other direction (i.e., to convert a PHP object into a JSON string), you can use the `json_encode()` function.

```
// convert PHP object into a JSON string
$text = json_encode($anObject);
```

Example

demo_file.php

```
<?php
    $myObj->name = "John";
    $myObj->age = 30;
    $myObj->city = "New York";

    $myJSON = json_encode($myObj);

    echo $myJSON;
?>
```

json_encode() function to convert PHP objects into JSON

```
{"name":"John","age":30,"city":"New York"}
```

JavaScript on the client, using an AJAX call to request the PHP file

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "demo_file.php", true);
xmlhttp.send();
```

References

- ❑ “Internet & World Wide Web: How to Program 5th editions”
- ❑ “Fundamentals of Web Development” Book by Randy Connolly and Ricardo Hoar, 2015
- ❑ W3schools: <https://www.w3schools.com/>

