EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

JSON and XML data formats, Web API

doc. Ing. Jan Roupec, Ph.D.

Brno University of Technology
Institute of Automation and Computer Science

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

FACULTY OF MECHANICAL ENGINEERING institute of automation and computer science

Data Transfer

- **special methods for:**
  - **data storage,**
  - **data manipulation and processing**
  - **data transfer (data transmission)**
- **Data transfer** is the process of using computing techniques and technologies to transmit or transfer electronic or analog data from one computer node to another. Data is transferred in the form of bits and bytes over a digital or analog medium, and the process enables digital or analog communications and its movement between devices.
- For us: interchange of data between processes – via computer network, direct communication between processes running in the same computer, using disc files, …
- We need some definition – Communication protocol (formal descriptions of digital message **formats** and rules)
- Several levels – physical (technical realization – Ethernet), data transport (TCP/IP), application (data structure and meaning)

Data Formats

- Binary form vs. Text form (human readable)
- Proprietary solution vs. Open Solution
- binary forms and proprietary solutions are important for programmers
- examples of text-based open data formats for general use (encoding of characters is important):
    - plain text
    - CSV – comma-separated values, spreadsheets or simple databases
    - **XML** – a general purpose markup language (standardized by W3C)
    - **JSON – JavaScript Object Notation**
- many other for special purposes:
    - YAML – superset of JSON, for configuration files
    - HTML – web pages
    - CSS – web pages styles
    - LaTeX – document markup language,
    - …

Data Formats

- data = values + structure
- no possibility for structure description in plain text and/or CSV
- CSV: table shape expected, rows, columns, 1st row may be used for attribute's names, 2D architecture
- **XML** and **JSON** – data structure and values
- data format should be common for many applications and purposes (to enable data exchange between incompatible systems)

XML Data Format

- e**X**tensible **M**arkup **L**anguage,
- designed to store and transport data,
- designed to be both human- and machine-readable,
- complex and complicated data structure may be described in XML,
- XML is not a programming language!
- Parsing of XML texts is supported in many programming languages (C, C++, C#, JavaScript, Python, php, ...)
- used for:
  - data transfer (especially web applications)
  - data export from databases
  - storing different documents (docx, xlsx, ...)
  - configuration data for computer applications (games, Android apps, ...)
  - ...

XML Data Format

- the following xml description is simplified but usable
- markup and content
- tag – markup construct
- element – logical document component
- short (but complex) example

example_type	book	
library	id	1001
	author	Dick Francis
	title	Wheels
	id	2026
	author	Agatha Christie
	title	The Moving Finger

XML Data Format

example_type	book	
library	id	1001
	author	Dick Francis
	title	Wheels
	id	2026
	author	Agatha Christie
	title	The Moving Finger

```
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

XML Data Format

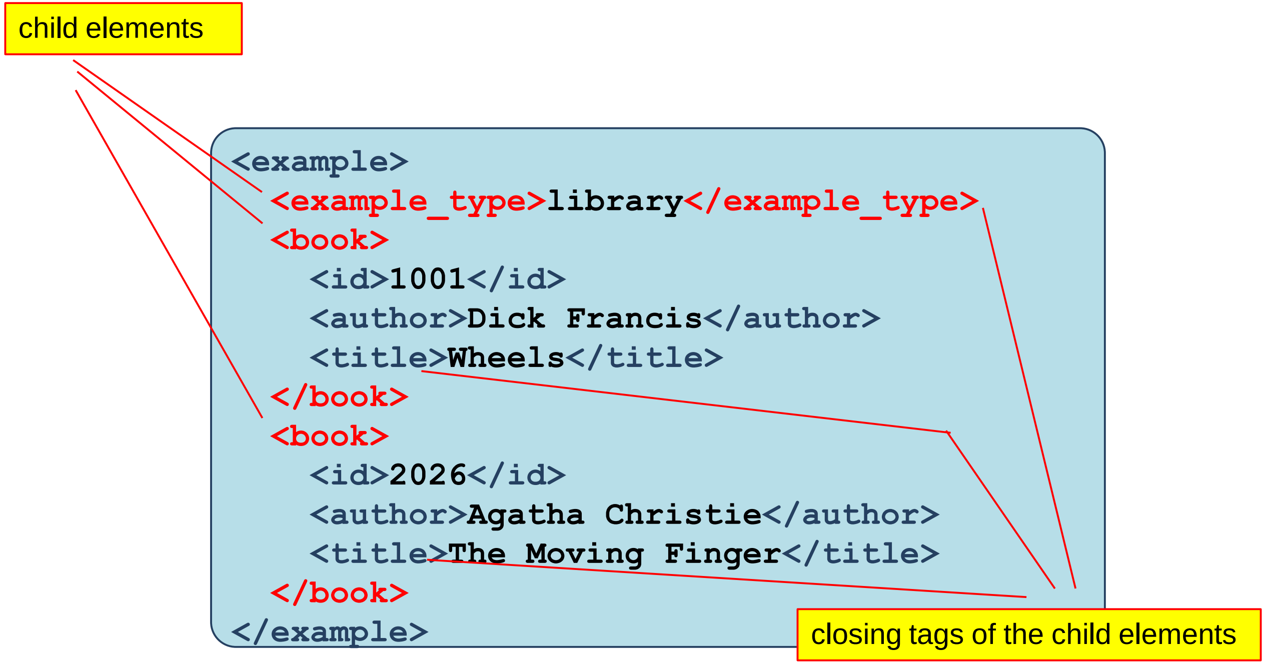
root element

the root element is a parent of all other elements

```xml
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

closing tag of the root element

XML Data Format



child elements

```
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

closing tags of the child elements

XML Data Format

subchild elements

the nesting depth of the elements is not limited

```
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

closing tags of subchild elements

XML Data Format

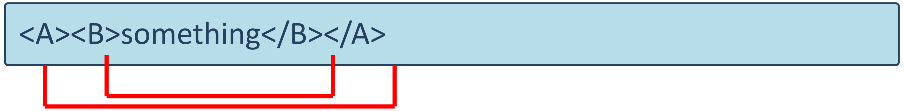
- XML elements can have **attributes** in name/value pairs just like in HTML.
- In XML, the attribute values must always be quoted:

```
<example type="library">
  <book category="crime story">
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book category="crime story"> >
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

XML Data Format

Nesting of Elements

- XML elements must be properly nested (like braces, no crossing).



```
<A><B>something</B></A>
```

Elements Naming Rules and Conventions

- case sensitive
- no predefined tag names
- name may contain letters, digits, hyphens, underscores, and dots
- name can start with a letter or underscore
- name mustn't start with the sequence **xml**

XML Data Format

Closing Tags

- All elements must have a closing tag.

```
<element>content</element>
<element></element>
<element />
```

empty element

self-closing empty element

The XML prolog

- Optional, if present, it must come first in the document.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- UTF-8 is the default character encoding for XML documents.
- The XML prolog does not have a closing tag.

Comments

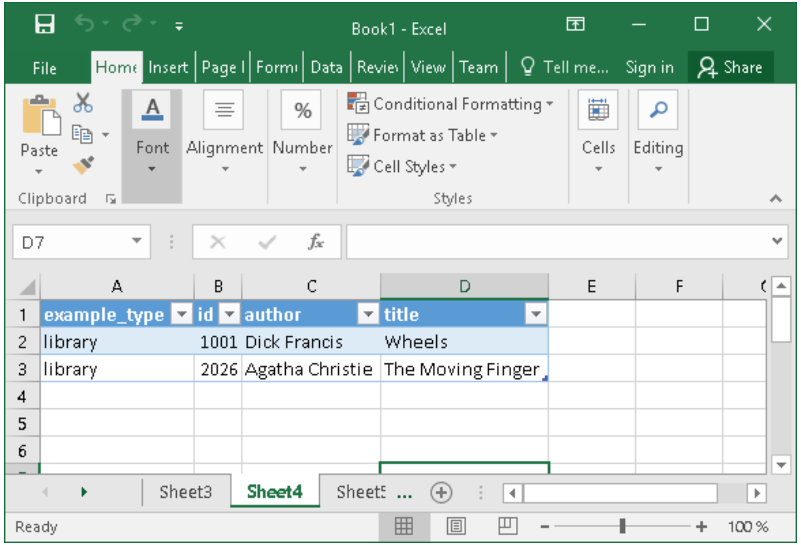
```
<!-- This is a comment -->
```

XML – Import to Excel

```
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

saved in test.xml

- Data
- From Other Sources
- From XML...



	A	B	C	D
1	example_type	id	author	title
2	library	1001	Dick Francis	Wheels
3	library	2026	Agatha Christie	The Moving Finger
4				
5				
6				

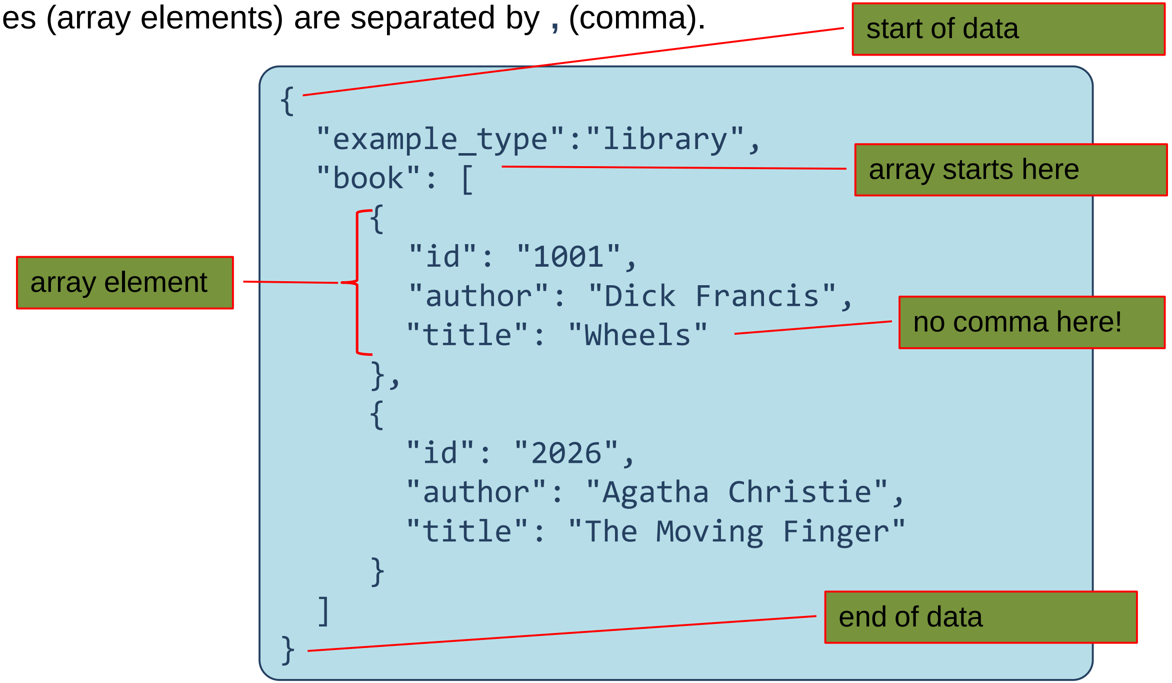
ai INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

JSON

- **J**ava**S**cript **O**bject **N**otation
- **text-based open standard** for human-readable data interchange,
- format was specified by Douglas Crockford (American programmer, born 1955 in Minnesota, games for Atari, now working in PayPal),
- related to but not limited to web programming,
- lightweight alternative to XML (in XML, almost 50% of characters are used for control tags and parameters),
- language independent, supported by many programming languages – JavaScript, C, C++, Java, Python, Perl, php, Matlab, ...
- Web services and APIs use JSON format to provide public data
- The filename extension is **.json**
- non-optimal for binary data transfer
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.

JSON – Syntax

- Data is represented in **name/value** pairs.
- Curly (compound) braces hold objects and each name is followed by : (colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays, values (array elements) are separated by , (comma).



```
{
  "example_type":"library",
  "book": [
     {
       "id": "1001",
       "author": "Dick Francis",
       "title": "Wheels"
     },
     {
       "id": "2026",
       "author": "Agatha Christie",
       "title": "The Moving Finger"
     }
  ]
}
```

start of data

array starts here

array element

no comma here!

end of data

JSON – Data Types

- **number** – a signed decimal number that may contain a fractional part and may use exponential E-notation (no difference between integer and floating point)

- **string** – a sequence of Unicode characters (may be empty), backslash escaping

- **boolean** – true or false

- **array** – order list of values (zero or more)

- **null** – an empty value

JSON – Import to Matlab

- **jsondecode** – JSON text to Matlab
- **jsonencode** – from Matlab internal representation to JSON format
- functions exist in Matlab ver. 2016b and higher

```
{
  "example_type":"library",
  "book": [
     {
          "id": "1001",
          "author": "Dick Francis",
                                 "title":
"Wheels"
     },
     {
        "id": "2026",
        "author": "Agatha Christie",
        "title": "The Moving Finger"
     }
  ],
}
```

saved in test.json

```
>> fname = 'test.json';
>> val =
jsondecode(fileread(fname))

val = 

  struct with fields:

 example_type: 'library'
         book:        [2×1 struct]

>> val.book(2).author

ans =

    'Agatha Christie'

>>
```

Matlab to JSON

```
>> books = [struct('id', '1001', 'author', 'D. Francis', 'title', 'Wheels')
struct('id', '2026', 'author', 'A. Christie', 'title', 'The Moving Finger')]

books =

  2×1 struct array with fields:

    id
    author
    title

>> data = struct('type', 'library', 'book', books)

data =

  struct with fields:

    type: 'library'
    book: [2×1 struct]

>> jsonencode(data)

ans =

    '{"type":"library","book":[{"id":"1001","author":"D. Francis","title":"Wheels"},
{"id":"2026","author":"A. Christie","title":"The Moving Finger"}]}'

>>
```

JSON – Import to Matlab

- another example – measured data

```
{
    "interval" : "0.1",
    "values" : [11,12,13.5,15,12,10]
}
```

saved in values.json

```
>> fname = 'values.json';
>> val = jsondecode(fileread(fname))

val = 

  struct with fields:

    interval:       '0.1'
      values:      [6×1 double]

>> val.values(3)

ans =

   13.5000

>> 
```

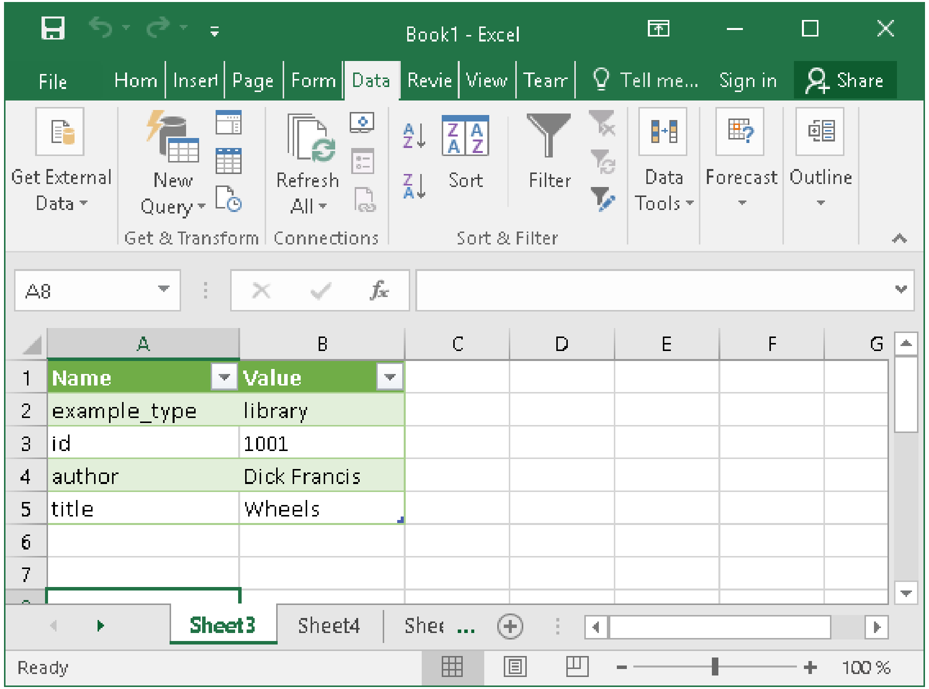
JSON – Import to Excel

- simple example – should be a table

```
{
            "example_type":"library",
            "id": "1001",
            "author": "Dick Francis",
            "title": "Wheels"
}
```

saved in values.json

- Data
- New Query
- From File
- From Text
- select JSON file



Name	Value
example_type	library
id	1001
author	Dick Francis
title	Wheels

Web API

WWW service – brief history

- World Wide Web
- march 1989, CERN (Conseil Européen de Recherche Nucléaire), Geneva, European institute (paid by 18 states)
- the need to distribute information to researchers (working in different locations/states/continents), originally in the field of high energy physics field
- Tim Berners-Lee – proposed a hypertext system to link together documents stored on different computers connected to the network
- document is described using HTML language (HyperText Markup Language)
- **http** protocol (hyper text transfer protocol)
- documents linked by the subject
- gradual expansion to World Wide Web
- march 1991: **Tim Berners-Lee**: HTML (Hyper Text Markup Language) V0.9 definition, assumed the existence of a documents generator, not direct use of html

Web API

WWW service – brief history (cont.)

- the first public use – January 1992, Geneva
- April 1993: approx. 60 www servers
- the oldest www browsers used text mode only (UNIX)
- march 1993: students in NCSA (National Center of Supercomputer Applications, The University of Illinois at Urbana-Champaign), browser Mosaic
- Mosaic was also a client for earlier protocols (ftp, nntp, gopher)
- Mosaic usually described as the first graphical web browser, but there were older ones (WorldWideWeb - Nexus, ViolaWWW)
- Netscape Navigator: Marc Anderssen (NCSA NCSA, one of main Mosaic programmers) and Jim Clark (Silicon Graphics) established Netscape Communications

Web API

web service principles

- uses **http** or **https** protocols – hyper text transfer protocol (secure)
- client/server architecture
- usually client is a web browser (Firefox, ...), but may be something else (Excel, ...)
- request is sent from client to server – URL
- server sends reply – some data, usually in html format, but may send any form of text data (plain text, xml, json, ...), client uses received data (for displaying, ...)

Web API

URL (Uniform Resource Locator)

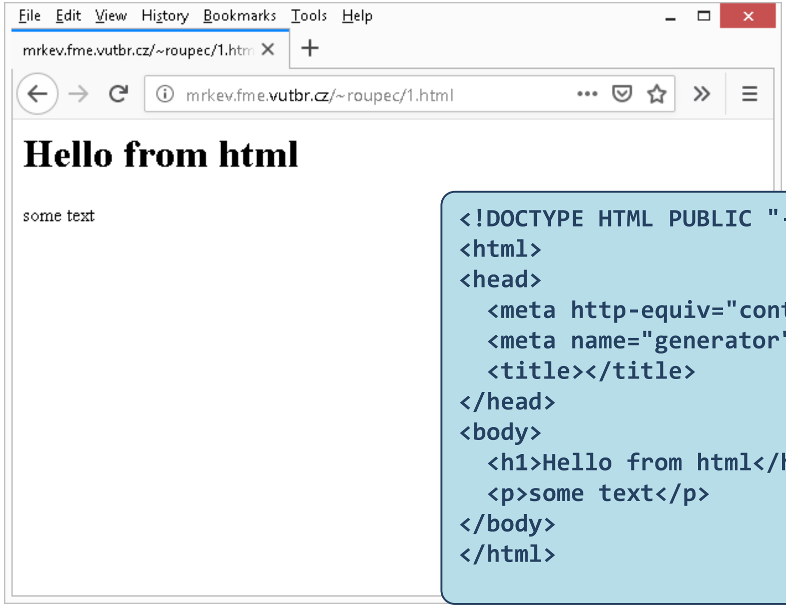
one address structure for different services

type://user:password@computer:port/path?query#fragment

- **type** – protocol to be used (http, ftp, ...)
- **user**, **password** – fields exist, but it's not recommended to use them
- **computer** – identification of the server (name or IP address)
- **port** – identification of the (server) program, usually omitted, defaults (80 for http, 443 for https)
- **path** – specification of the file requested (path + file name), defaults for file names in server configuration, usually index.html or index.php
- **query** – extra parameters for the web server
- **fragment** – specification of the part of the document (bookmark inside the document, position for scrolling)

Web API

Simple static web page


File Edit View History Bookmarks Tools Help
mrkev.fme.vutbr.cz/~roupec/1.htm
mrkev.fme.vutbr.cz/~roupec/1.html

Hello from html

some text

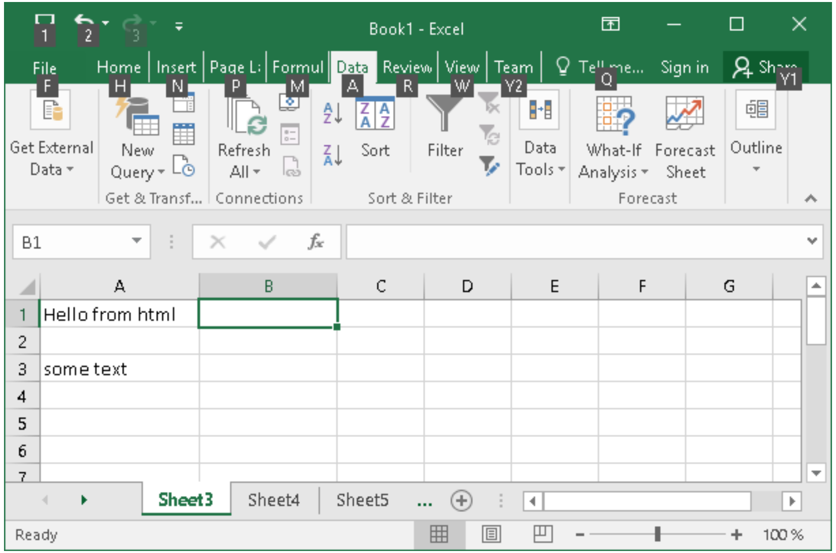
saved in 1.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">
  <meta name="generator" content="PSPad editor, www.pspad.com">
  <title></title>
</head>
<body>
  <h1>Hello from html</h1>
  <p>some text</p>
</body>
</html>
```

Web API

Using other clients – from Web to Excel

- Data
- From Web
- type URL (http://mrkev.fme.vutbr.cz/~roupec/1.html)
- Import



	A	B	C	D	E	F	G
1	Hello from html						
2							
3	some text						
4							
5							
6							
7							

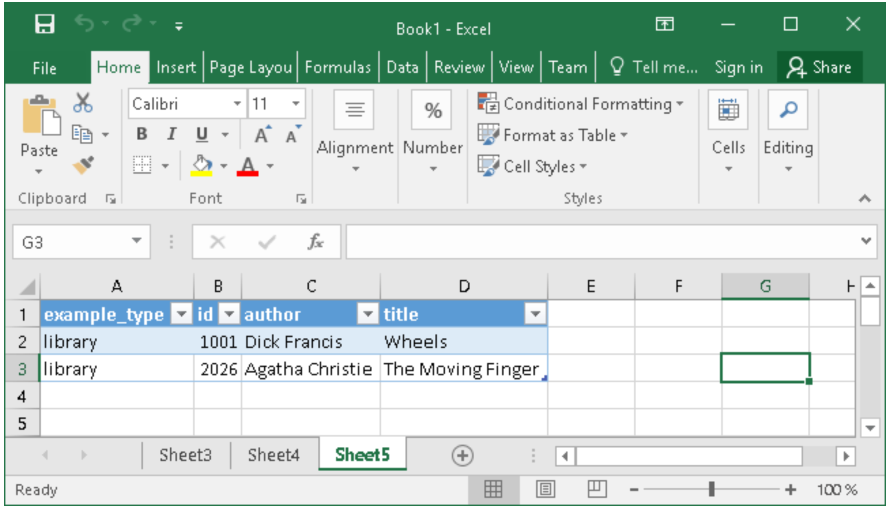
Web API

From Web to Excel – server sends xml data

```
<example>
  <example_type>library</example_type>
  <book>
    <id>1001</id>
    <author>Dick Francis</author>
    <title>Wheels</title>
  </book>
  <book>
    <id>2026</id>
    <author>Agatha Christie</author>
    <title>The Moving Finger</title>
  </book>
</example>
```

saved in test.xml

- Data
- From Web
- type URL (http://mrkev.../~roupec/test.xml)
- Import



	A	B	C	D
1	example_type	id	author	title
2	library	1001	Dick Francis	Wheels
3	library	2026	Agatha Christie	The Moving Finger

Web API

Dynamic Web Pages

- for us: it's not very useful to import a content of static pages
- in general: not all web pages may be written as **static** pages
- client can ask for different file types (txt, xml, json, ...)
- static page on the server side: server sends just the content of the required file
- static pages are not possible to use for e-shop, information system, booking system, searching, help desk, ...
- solution:
  - run executable files on server (e.g. .exe is required in URL) – CGI (Common Gateway Interface)
  - include a code into html document and process this code by a server (e.g. php)
  - in both cases it's possible to access databases, to compute something, ...
  - in both cases output may be generated in many forms – html, xml, json, text, ...
- **php**
  - popular programming language for web applications
  - inspired by C/C++ languages
  - variable names must start with $
  - object oriented

Web API

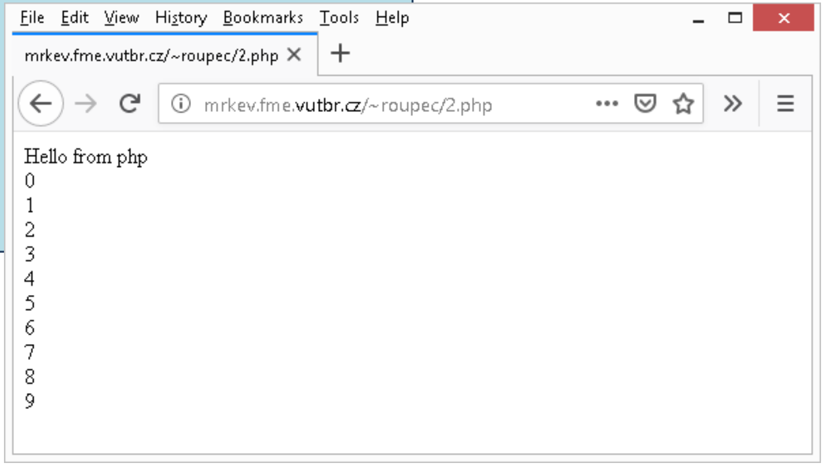
Static and dynamic web pages in general

- client obtains data from the server in the form of:
  - text in HTML/XHTML
  - JAVA applet
- on the server side:
  - document is saved in the text file and is sent without any changes (as it is) = static document on the server side
  - document is generated as the result of any executable task
    - URL is a location of any executable file (CGI)
    - SSI
    - html document contains executable part (statements – php)
- on the client side:
  - displaying of the html document received
  - scripts processing (html text may contain some statements – usually in JavaScript)

Web API

Example of the dynamic web pages in php languge

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">
  <meta name="generator" content="PSPad editor, www.pspad.com">
  <title></title>
</head>
<body>
<?php
  echo "Hello from php<br />";
  for($i = 0; $i < 10; $i++)
    echo "$i<br />";
?>
</body>
</html>
```


File
Edit
View
History
Bookmarks
Tools
Help
mrkev.fme.vutbr.cz/~roupec/2.php
mrkev.fme.vutbr.cz/~roupec/2.php

```
Hello from php
0
1
2
3
4
5
6
7
8
9
```

- mixture of html and php
- pure php is also possible

Web API

Dynamic Web Pages on the Client Side

- **JavaScript** is usually used (not the only possibility)
    - programming language
    - statements included in web documents
    - server doesn't take a care of them, simply sends them (as part of the document)
    - interpreted by a browser
    - inspired by C/C++ languages
    - no $ in variable names
    - object oriented
    - event-driven programming
    - all document elements are accessible from JavaScript (DOM)
    - possibility of creations and/or deletion of document elements
    - possibility to communicate with a web server (without reloading the page)

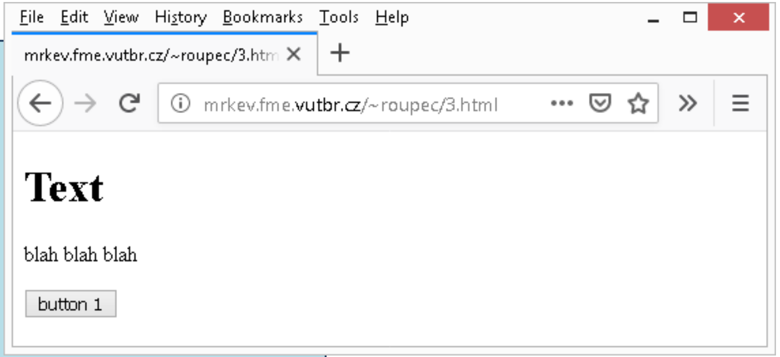
Web API

Dynamic Web Pages on the Client Side

```
<html>
...
</head>
<body>
<script language="JavaScript">

  function myfunc1()  {
    document.getElementById("id1").style.color = "green";
    document.getElementById("id1").innerHTML = "Changed";
  }

</script>
  <h1 id="id1">Text</h1>
  <p>blah blah blah</p>
  <input type="button" value="button 1" onClick="myfunc1();">
  </body>
</html>
```


File Edit View History Bookmarks Tools Help
mrkev.fme.vutbr.cz/~roupec/3.htm
mrkev.fme.vutbr.cz/~roupec/3.html
button 1

Text

blah blah blah

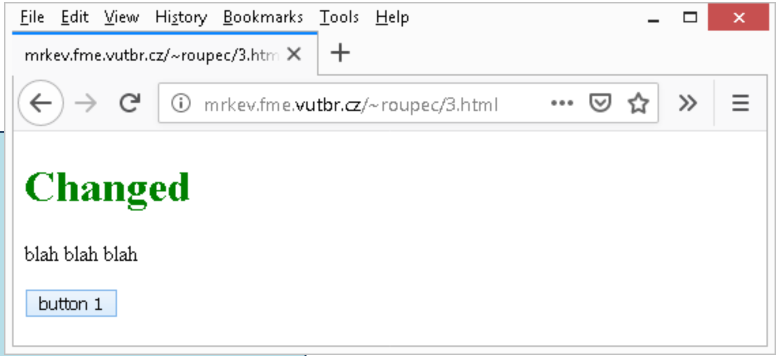
Web API

Dynamic Web Pages on the Client Side

```
<html>
...
</head>
<body>
<script language="JavaScript">

  function myfunc1()  {
    document.getElementById("id1").style.color = "green";
    document.getElementById("id1").innerHTML = "Changed";
  }

</script>
  <h1 id="id1">Text</h1>
  <p>blah blah blah</p>
  <input type="button" value="button 1" onClick="myfunc1();">
  </body>
</html>
```


File
Edit
View
History
Bookmarks
Tools
Help
mrkev.fme.vutbr.cz/~roupec/3.htm
mrkev.fme.vutbr.cz/~roupec/3.html

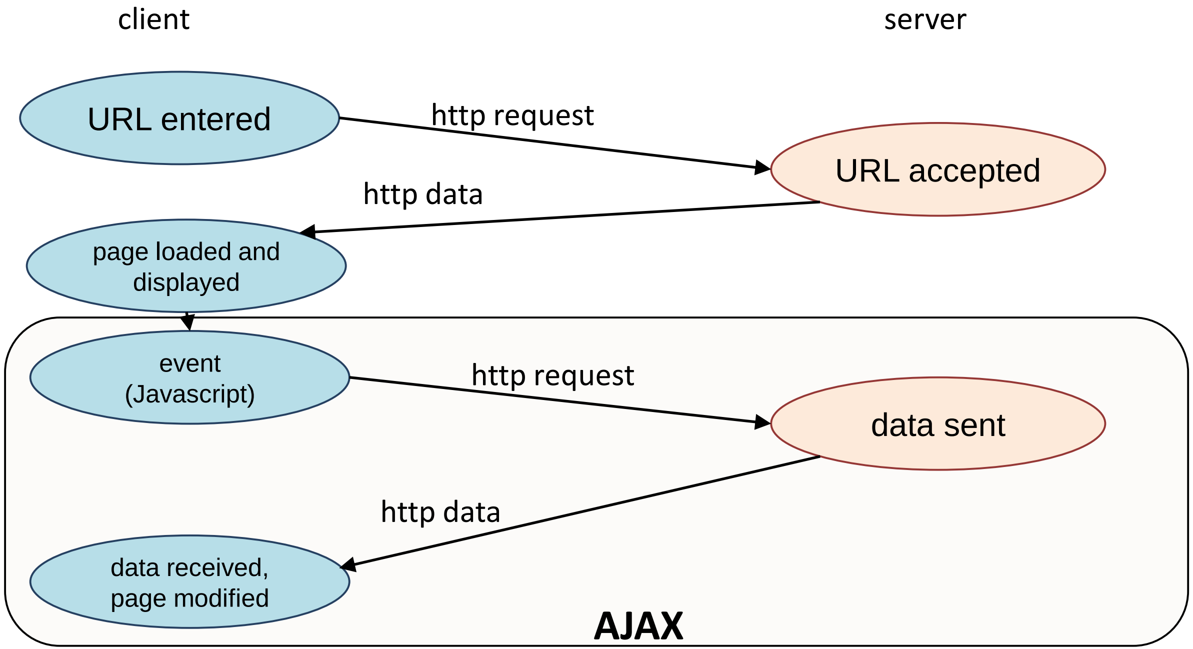
Changed

blah blah blah

button 1

Web API

AJAX – Asynchronous JavaScript and XML (AJAJ - ... JSON)



EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Studium moderní a rozvíjející se techniky VUT – SMART VUT

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING
Project reg. Nr.: CZ.02.2.69/0.0/0.0/18_056/0013325

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF MECHANICAL ENGINEERING