

JSON adatmodell

JSON adatmodell

Az előadás anyaga

Prof. Dr. Kovács László: Adatkezelés XML környezetbe,
Jeszenszky Péter: XML és további irodalom alapján készült el

Témakörök

1. JSON adatmodell
2. JSON vs. XML
3. JSON adattípusok
4. YAML formátum
5. Mintafeladatok

Igényelt kompetenciák

- JSON adatmodell
- YMAL formátum
- Mintafeladatok
- Környezet: XML szerkesztő (Oxygen, EditIX, Eclipse,)

XML - Előnyök

- „Könnyen *alkalmazható webes rendszerekben.*
- *Keresés (web) helyett lekérdezéssel (DB) juthatunk információhoz.*
- *Univerzális adatcsere formátum, amely hozzájárul az üzleti alkalmazások szabványos kommunikációhoz.*
- Gyártófüggetlenség.
- Platformfüggetlenség.

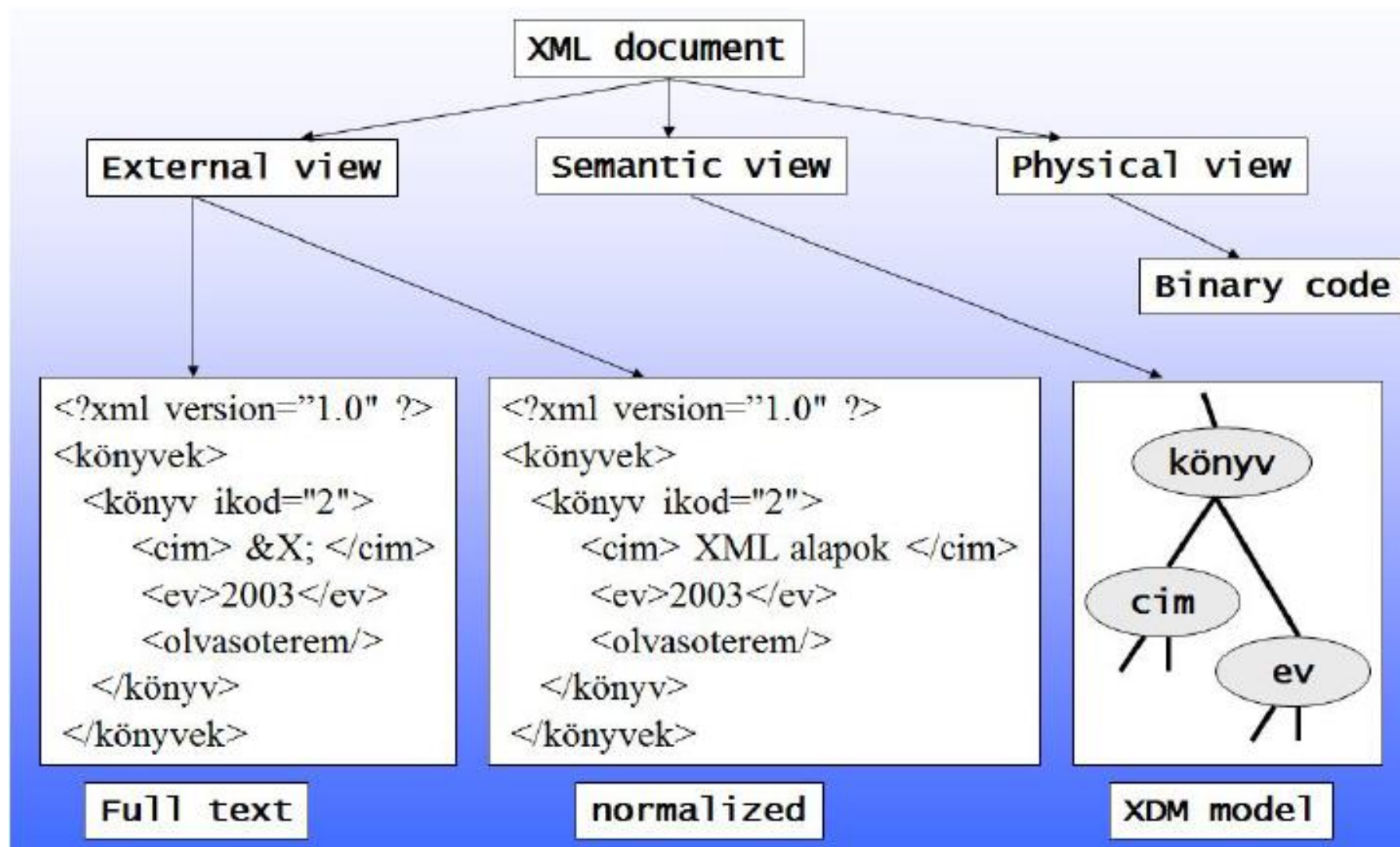
XML - Előnyök

- A számítógép képessé válik a *tartalom korlátozott megértésére*.
- Ezáltal lehetővé válik a tartalom *automatikus, gépi ellenőrzése*.
- Az iparban de-facto szabvány.

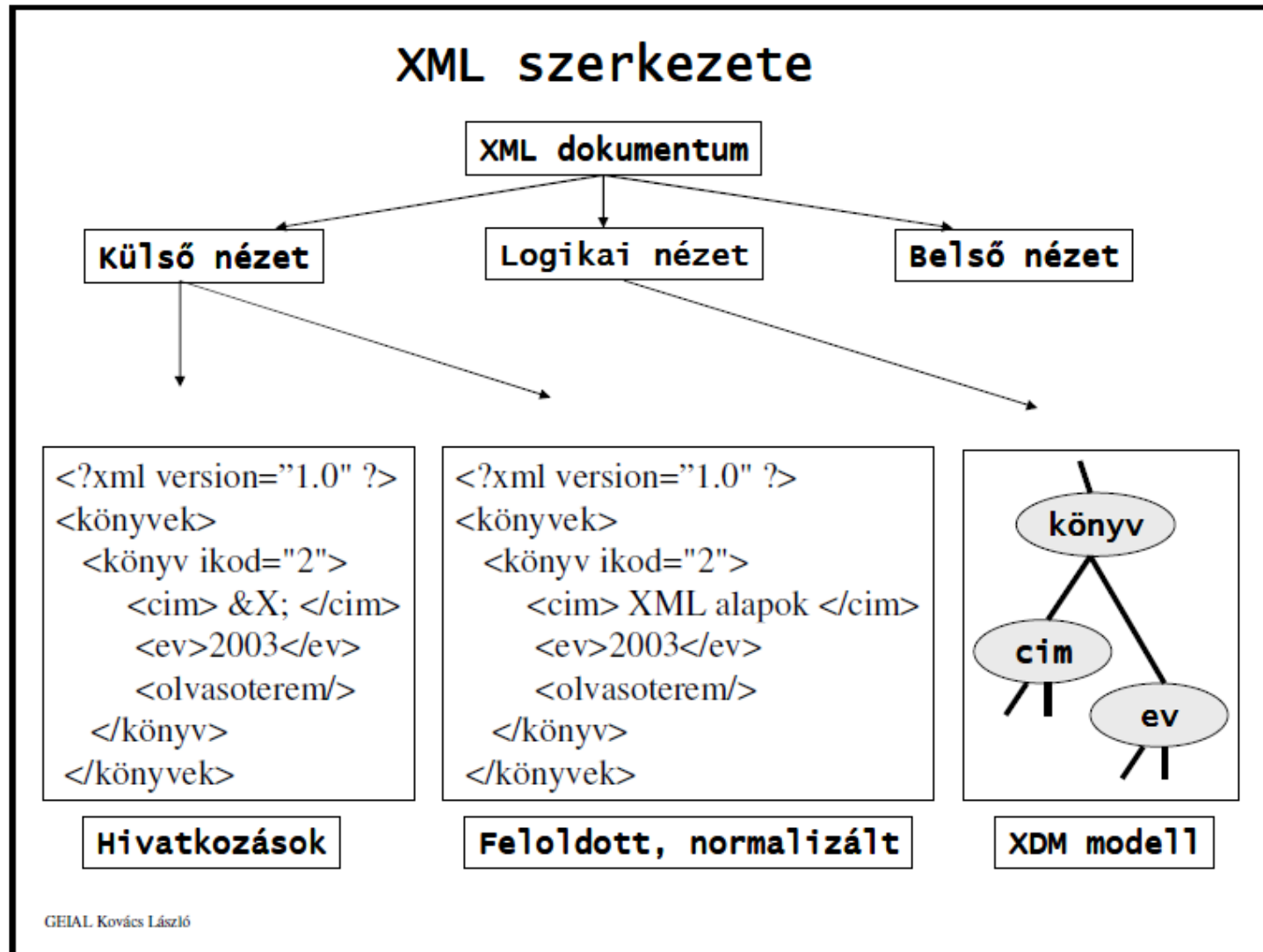
XML - Hátrányok

- Az eredeti szabvány *gyenge deklarációs rendszert* tartalmaz (DTD).
- A deklaráció *tervezési hibáinak javítása* igen költséges.
- *Bőbeszédű és nehézkesen* használható szintaxis.
- Nagy tárigény.
- Bonyolultság.
 - Se szeri, se száma az XML-hez *kötődő specifikációknak*.
- Mindezek ellenére fontos, együtt kell élni vele.”

XML dokumentum szerkezete - ismétlés



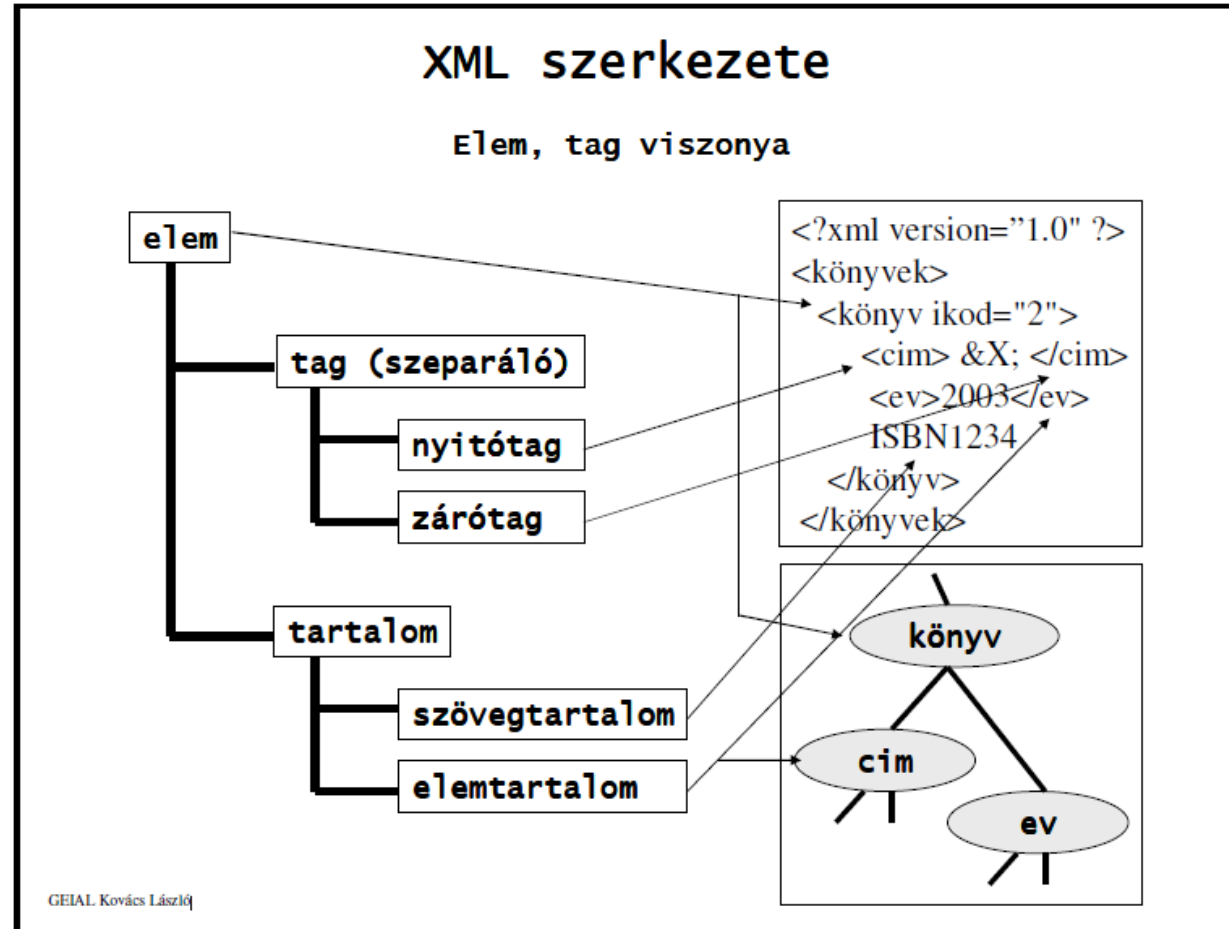
XML dokumentum szerkezete - ismétlés



Forrás: KovácsL

XML dokumentum szerkezete - ismétlés

Elem,
tag
viszonya



Forrás: KovácsL

JSON adatmodell – új igény

„Új igény

- gyors, egyszerű adatcsere

Megvalósítás:

- JSON: egyszerűsített XML.
- *Célja:* adatcsere.
- Napjainkban kibővül *séma* és *lekérdező nyelvekkel* is.”

JSON adatmodell - jellemzői

„A **JSON** (JavaScript Object Notation, JavaScript objektumjelölés)

- *Strukturált adatok* ábrázolására szolgál.
- *Szöveges formátum* - adatok tárolására és továbbítására.
- *Könnyűsúlyú szöveges nyelvfüggetlen adatcsere formátum.*
- Ember számára is *könnyen olvasható és írható.*
- Szoftverek által *könnyen generálható és feldolgozható.*

JSON adatmodell - ECMAScript

Az ECMAScript programozási nyelvből származik.

URL: <http://www.json.org/>

- A jelenleg aktuális a 10-es számú kiadás.
- Ecma International, *ECMAScript 2019 Language Specification*, Standard ECMA-262, 10th Edition, June 2019.

A jelenleg fejlesztés alatt álló verzió az ECMAScript 2020.

ECMA International

Nemzetközi nonprofit szabványosító szervezet.

- *Célterület:* infokommunikációs technológia (ICT), fogyasztói elektronika (CE).

Eredetileg 1961-ben alapították, jelenlegi nevén 1994 óta működik.

- *European Computer Manufacturers Association (ECMA)*

URL: <http://www.ecma-international.org/>

JSON vs. ECMAScript

A JSON az ECMAScript szintaxisan alapul, de nem teljesen kompatibilis vele.

További információk:

[URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON)

JSON adatmodell - története, jellemzői

A JSON formátumot eredetileg *Douglas Crockford* specifikálta.

Douglas Crockford (amerikai számítógépes programozó és vállalkozó, részt vett a JavaScript nyelv fejlesztésében) volt az első, aki *meghatározta és népszerűsítette* a JSON formátumot.

A JSON-t a *State Software* cég használta 2001-től.

A JSON hivatalos *MIME-típusa* `application/json`.

A JSON fájlok kiterjesztése pedig `.json`.

JSON adatmodell - jellemzői

A JavaScript szkriptnyelvből alakult ki adatstruktúrák és asszociatív tömbök reprezentálására (JSON-ban objektum).

A JavaScripttel való kapcsolata ellenére nyelvfüggetlen, több nyelvhez is van értelmezője.

Példa: JSON string

```
'{"name": „LL", "age":30, "car":null}'
```

JSON felhasználása

- *Böngészőbővítményeket és weboldalakot tartalmazó JavaScript alapú alkalmazások* írásakor használják.
- A JSON formátumot a *strukturált adatok sorosítására és továbbítására* használják *hálózati kapcsolaton* keresztül.
- Elsősorban a *szerver* és a *webalkalmazások* közötti adatátvitelre használják.
- *Modern programozási nyelvekkel* használható.

JSON adatmodell - megvalósítás

JSON: egyszerűsített XML.

Célja: adatcsere, napjainkban kibővül *séma és lekérdező nyelvekkel* is.

Kialakulása: könnyebbé tegye a *webszerveren* és a *böngészőben* futó kód közötti kommunikációt.

A böngészőben futó *JavaScript kód* a kapott adatot a beépített `eval()` függvénnnyel ki tudja értékelni.

JSON adatmodell - megvalósítás

Az `eval()` metódus *kiértékel* vagy *végrehajt* egy argumentumot.

Az eredmény - *szöveges karakterlánc* vagy *számérték*.

Pl.: kiértékeli a *karakterlánc-kifejezést*, majd visszaadja annak értékét. Az `eval("1 + 1")` eredménye 2."

JSON - Mire használható?

1. A szervertől kapunk egy string-ként értelmezhető ún. *JSON sztringet*.

Pl.: `'{"nev": „Lilla”, "kor":18, "tel": "0670", email": "nem tudom"}',`

2. Rendeljük hozzá egy változóhoz:

```
let ServerAnswer = '{"nev": „Lilla”, "kor":18, "tel": "0670", email": "nem tudom"}',
```

JSON - Mire használható?

3. A string-et ezután egy *JavaScript függvénnnyel JavaScript objektummá* alakítjuk - azaz *parse*-juk.

```
JSON.parse()
```

```
Pl.: let obj = JSON.parse(serverAnswer);
```

Kapott objektum: `név, kor, mobil, email`

4. A attribútumnak van *értéke*, amelyhez hozzá tudunk férni:

```
let personNev = obj.nev; // „Lilla”
```

```
let personKor = obj.kor; // 18
```

Fejlesztőkörnyezet - szabad szoftverek

„Szabad és nyílt forrású szoftverek:

- *Atom* (platform: Linux, macOS, Windows; licenc: *MIT License*)
<https://atom.io/>; <https://github.com/atom/atom>
Javasolt csomag: *Pretty JSON* <https://atom.io/packages/pretty-json>
- *Eclipse* (platform: Linux, macOS, Windows; licenc: *Eclipse Public License 2.0*) URL: <https://www.eclipse.org/>

Javasolt bővítmények:

- *JavaScript Development Tools (JSDT)*
<https://www.eclipse.org/webtools/jsdt/>

JSON Editor Plugin

<https://marketplace.eclipse.org/content/json-editor-plugin>

Fejlesztőkörnyezet - szabad szoftverek

- *Notepad++* (platform: Windows; licenc: GPLv2)

<https://notepad-plus-plus.org/>

Javasolt bővítmény: *JSToolNpp*

- <http://www.sunjw.us/jstool/npp/>
- <https://github.com/sunjw/jstoolnpp>
- *Visual Studio Code* (platform: Linux, macOS, Windows; license: MIT License)
 - <https://code.visualstudio.com/>
 - <https://github.com/Microsoft/vscode>

Fejlesztőkörnyezet - nem szabad szoftverek

Nem szabad szoftverek:

- *<Oxygen/> XML Editor* (platform: Linux, macOS, Windows)
<https://www.oxygenxml.com/>
URL: https://www.oxygenxml.com/xml_editor/json_editor.html
- *IntelliJ IDEA* (platform: Linux, macOS, Windows) – szabad lehet, ha..
<https://www.jetbrains.com/idea/>
<https://www.jetbrains.com/help/idea/json.html>

JSON-P: Java API for JSON Processing

JSR 374: Java API for JSON Processing 1.1 (Final Release)

<https://www.jcp.org/en/jsr/detail?id=374>

- A Java EE 7-ben jelent meg (JSR 353).
- Lásd a *javax.json*, *javax.json.spi* és *javax.json.stream* csomagokat.

JSON megjelenítés böngészőkben

Firefox: tartalmaz beépített JSON megjelenítőt.

- URL: https://firefox-source-docs.mozilla.org/devtools-user/json_viewer/

Google Chrome: ajánlott kiterjesztések

- JSON Formatter
<https://chrome.google.com/webstore/detail/jsonformatter/bcjindccaagfpajjmafaapmmgkkgkghgoa/>

Forrás: JeszenszkyP

JSON adatmodell – hierarchikus felépítés

„Hierarchikus adatmodell

- XML-hez hasonló séma, de
- *Nincs attribútum elem,
nincs névtér,...*

Egyszerű példa:

```
{
  _id: <ObjectId>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

JSON adatmodell - sémai elemei

Séma elemei:

- adatbázis,
- kollekció (tábla),
- dokumentum (rekord),
- *mező*: tömb, dokumentum.

Egymásba ágyazott struktúrák.

Séma mentes.

JSON and BSON formátumok.

A screenshot of a code editor window titled "Output". The editor displays a JSON document with line numbers 1 through 24 on the left margin. The JSON structure is as follows: a root object with a "class" property containing an array named "student". This array contains three objects, each representing a student with properties for "vezeteknev" (surname), "keresztnev" (first name), "becenev" (nickname), and "kor" (age).

```
1 {  
2   "class": {  
3     "student": [  
4       {  
5         "vezeteknev": "Fekete",  
6         "keresztnev": "Peter",  
7         "becenev": "Petya",  
8         "kor": 22  
9       },  
10      {  
11        "vezeteknev": "Kek",  
12        "keresztnev": "Dora",  
13        "becenev": "Dorka",  
14        "kor": 20  
15      },  
16      {  
17        "vezeteknev": "Zsoldos",  
18        "keresztnev": "Andrea",  
19        "becenev": "Andi",  
20        "kor": 18  
21      }  
22    ]  
23  }  
24 }
```

JSON adatmodell - szintaktikája

Szerkezeti elemek szintaktikája:

- *struktúra*: { }
- *mező*: name : value
- *tömb*: [v , v ,] (fontos a pozíció)

A dokumentumok azonosítása

egy kulcs mezővel történik:

`_id:mező`

JSON adatmodell

A kulcsmező lehet a felhasználó által adott vagy rendszer által generált.

A N:M kapcsolat modellezése:

- Hivatkozás az **_id** mezőre
- Az értékek típusosak:
(number, string)

Összehasonlítás XML - JSON

Hasonlóságok:

- egyszerű szövegformátum,
- emberek által olvasható-írható,
- hierarchikus felépítésű,
- JavaScriptben parse-olható,
- adatokat küldhetünk AJAX hívásokkal.

Különbségek:

- nincs *end tag*,
- rövidebb,
- *gyorsabban írható és olvasható*,
- a JavaScriptben beépített `eval()` függvénnnyel parse-olható,
- *tömböket* használ,
- nincsenek *nyelvi kulcsszavak*.

JSON vs. XML

No .	JSON	XML
1.	A JSON a JavaScript Object Notation	Az XML az eXtensible Markup Language
2.	A JSON egyszerűen olvasható és írható.	Az XML kevésbé egyszerű, mint a JSON.
3.	A JSON-t könnyű megtanulni.	Az XML kevésbé egyszerű, mint a JSON.
4.	A JSON <i>adatorientált</i> .	Az XML <i>dokumentumorientált</i> .
5.	A JSON <i>nem nyújt megjelenítési képességeket</i> .	Az XML lehetővé teszi az adatok megjelenítését, mivel ez egy jelölőnyelv.

JSON vs. XML

6.	A JSON támogatja a tömböt.	Az XML nem támogatja a tömböt.
7.	A JSON kevésbé biztonságos, mint az XML.	Az XML biztonságosabb.
8.	A JSON fájlok emberileg olvashatóbbak, mint az XML.	Az XML fájlok kevésbé olvashatók az ember számára.
9.	A JSON csak a <i>text and number data type</i> támogatja.	Az XML számos adattípust támogat, pl.: szöveget, számot, képeket, diagramokat, grafikonokat stb. „

JSON vs. XML

„A JSON és az XML közös jellemzői:

- Egyszerűség (egyértelműen a JSON a nyerő).
- Az ember számára is könnyen írható és olvasható formátumok (szöveg alapú szabvány).
- Szoftverek által könnyen generálható és feldolgozható (itt is egyértelműen a JSON a nyerő).
- Formátum (egyértelműen a JSON a nyerő).

JSON vs. XML

A fő különbség az, hogy a

- JSON *adat-orientált*,
- XML *dokumentum-orientált*.

Adatszerkezetek ábrázolásához a JSON tökéletes választás.

- *Előnye:* az XML-hez képest, hogy kevésbe bőbeszédű.

JSON vs. XML

- *Dokumentum-középpontú* alkalmazásokhoz az XML-t használjuk.
- *Előnye:* a JSON-hoz képest, hogy kiterjeszthető, és hogy kiforrottabb infrastruktúra áll hozzá rendelkezésre (XML Schema, XSLT, XQuery, ...).

JSON adattípusok

Négy *primitív adattípus* ábrázolását teszi lehetővé:

- szám,
- string,
- logikai érték,
- null.

Két *strukturált típus*:

- Tömb,
- Objektum.

JSON alap adattípusai – primitív adattípus

- *Szám*: lehet *lebegőpontos* és *egész* (C és Java hasonló)

A különbség az, hogy *oktális* és *hexadecimális formátum* itt nem használható. Pl.: { "age" : 30 }

- *String*: idézőjelek közé zárt *Unicode karakter*, szükség szerint visszaper-jellel kivédve. Pl.: { "name" : "John" }

A karakter egy hosszúságú karakterláncnak felel meg.

Hasonlít a C vagy Java karakterláncaihoz.

JSON alap adattípusai – primitív adattípusok

- *Boolean*: értéke `true` (igaz) vagy `false` (hamis).

Pl.: `{"sale":true}`

- *JSON null*: a JSON értéke `null` lehet (üres érték)

Példa:

`{"middlename":null}`

JSON alap adattípusai – strukturált típusok

Tömb: értékek rendezett halmaza.

- A tömb [], (nyitó és zárójel) operátorokat használunk.
- Az értékeket , (vessző)-vel választjuk el egymástól.
- Az értékeknek *nem kell azonos típusúnak* lenniük.

Példa:

```
{  
  „hallgatok”: [ "John",  „1234",  "Peter" ]  
}
```

JSON alap adattípusai - strukturált

Objektum: név-érték párok rendezetlen halmaza.

Egy objektum {nyitó és záró kapcsosjel } zárul.

A ':' karakter választja el a *kulcsot* és az *értéket*.

A *név-érték* párok , (vessző)-vel tagoltak.

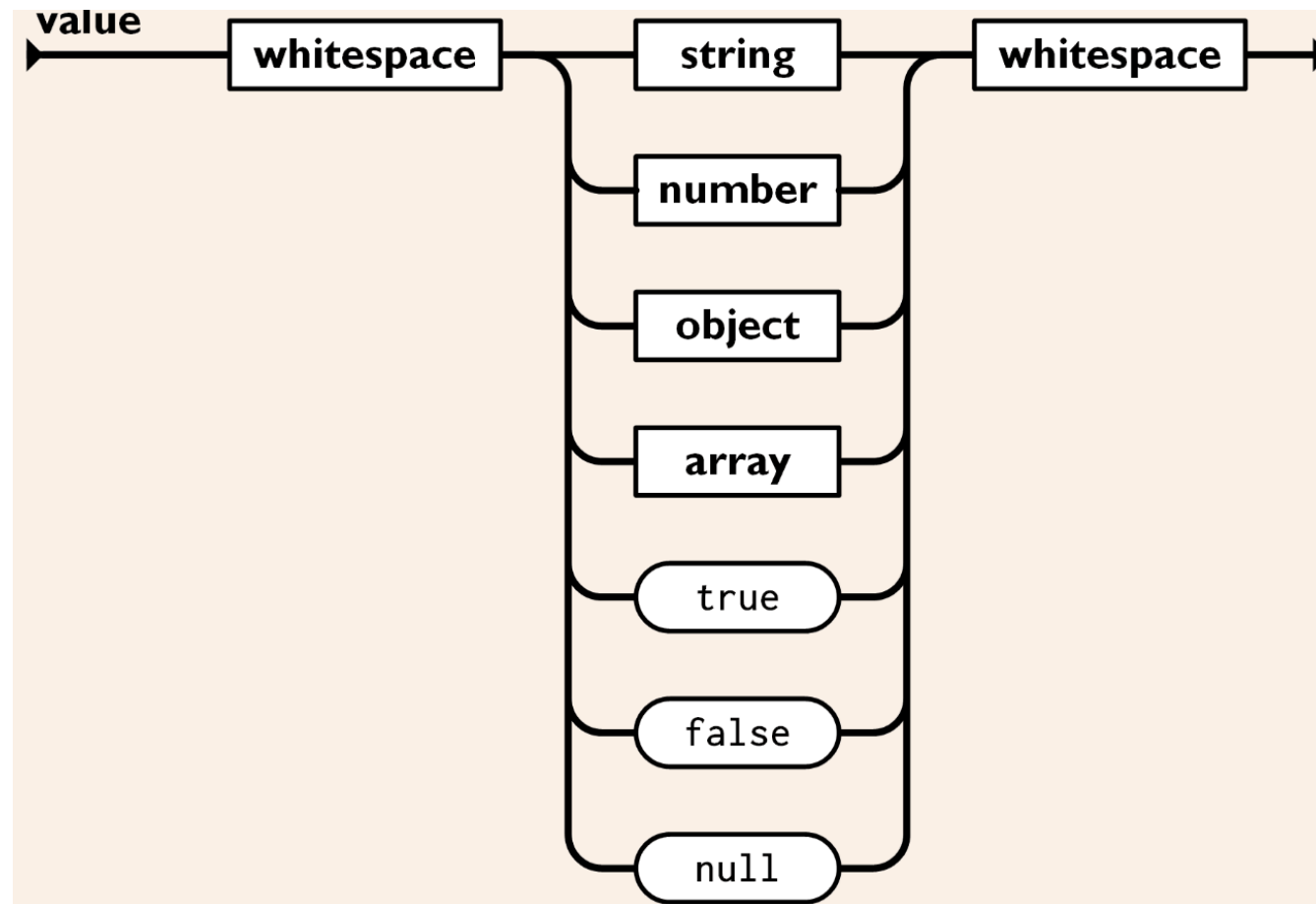
A *névnek*: sztringeknek kell lenni és különbözni egymástól.

Érték lehet: idézőjelek közé írt *karakterlánc*, *szám*, *logikai igaz/hamis*, *null*, *objektum* vagy *tömb*.

Tokenek

- A JSON szöveg tokenek olyan sorozata, mely megfelel a JSON érték nyelvtani szabálynak.
- Tokenek:
 - Szerkezeti tokenek a {, }, [,], : és , karakterek.
 - Sztringek
 - Számok
 - Literális tokenek a true, false és null karakterláncok.
- Tokenek előtt és után megengedettek *whitespace* karakterek, melyek nem lényegesek.
 - *Whitespace* karakter: HT (U+0009), LF (U+000A), CR (U+000D), szóköz (U+0020)
 - A tokenek közül csak a sztringekben megengedettek *whitespace* karakterek.

JSON értékei - értékek

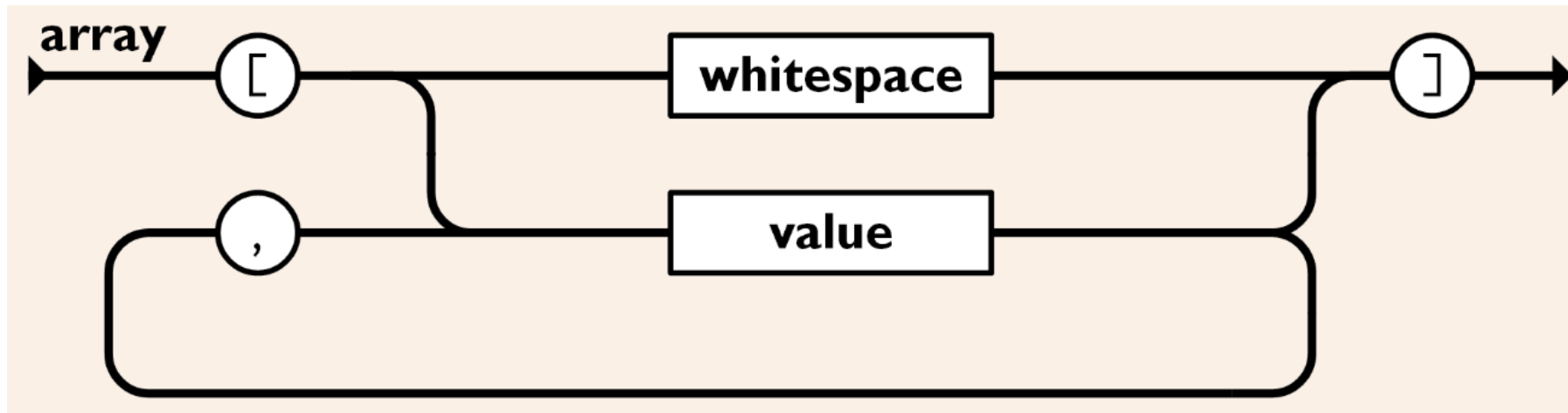


Sztringek

- *Unicode karakterek sorozatai*, melyeket idézőjelek (U+0022) határolnak.
- *Bármely karaktert* tartalmazhatják, azonban az alábbiakat csak levédve:
 - *idézőjel* (U+0022), *backslash* (U+005C), *vezérlő karakterek* (U+0000–U+001F)
- Speciális karakterek megadásához rendelkezésre állnak az *escape* szekvenciák: `\", \\, \t, \n, \r, ...`

Tömbök

- Tetszőleges *számú érték rendezett sorozata* (lehet üres).
 - Az elemek *különböző típusúak* is lehetnek.
 - Hivatkozás az elemre a elem számmal pl. [1] lehet



Tömbök - példa

- ["Athos", "Porthos", "Aramis", "d'Artagnan"]
- [9, 14, 19, 25, 26, 28]
- ["Pi", 3.141593, null, true]
- [[45.7370889, 16.1133866], [48.5852340, 22.8981217]]

JSON Array - példa

JSON Array of String

```
[ "Vasárnap" , "hétfő" , "kedd" , "szerda" ,  
  "csütörtök" , "péntek" , "szombat" ]
```

JSON Array of Numbers

```
[12, 34, 56, 43, 95]
```

JSON Array of Booleans

```
[true, true, false, false, true]
```


JSON Array - példa

JSON Array of Objects

```
{  
  „hallgatok”: [  
    { "name": „Olga", "email": „olga@gmail.com", „kor”:40  },  
    { "name": „Dóra", "email": „dori00@gmail.com", „kor”:33},  
    { "név": „Lilla", "email": "lilla@gmail.com", "kor":22},  
    { "name": „Sara", "email": „sara@gmail.com", „kor”:11}]  
}
```

JSON Array - példa

JSON többdimenziós tömb

```
[  
  [ "a" , "b" , "c" ],  
  [ "m" , "n" , "o" ],  
  [ "x" , "y" , "z" ]  
]
```

JSON Comments

A JSON nem támogatja a *megjegyzéseket*. Ez nem szabvány.

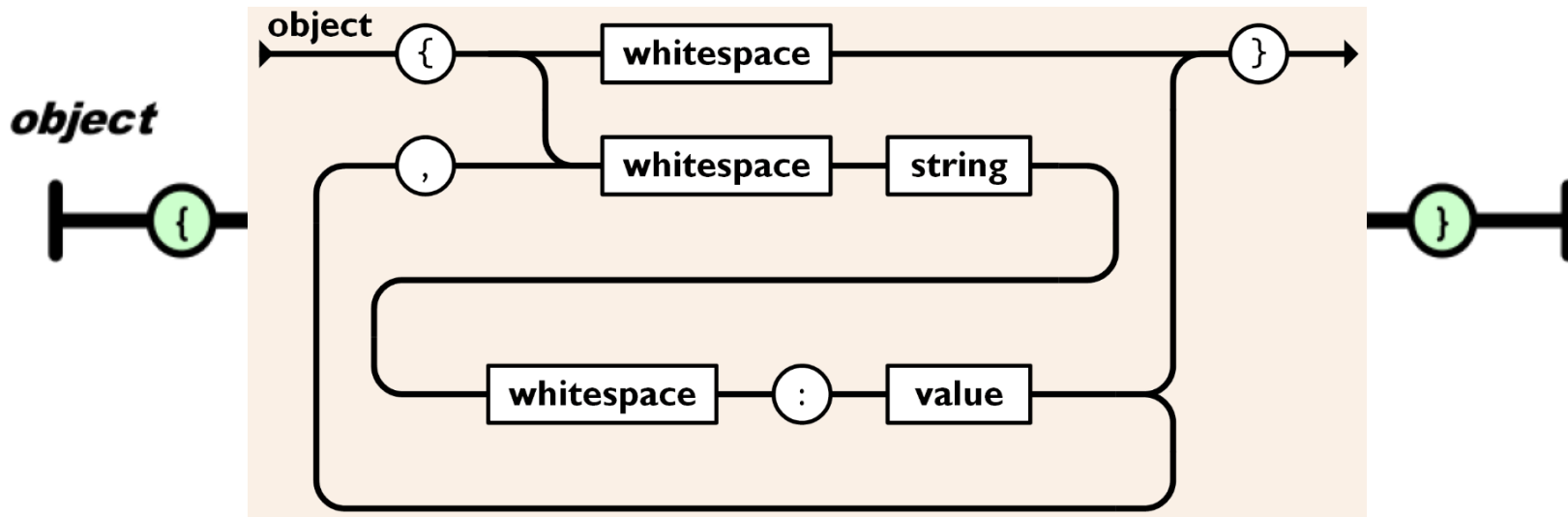
De hozzáadhat egy un. *extra attribútumot* a megjegyzéshez a JSON objektumban.

```
{
  "alkalmazott" : {
    "név" : „Lili” ,
    "fizetés" : 560000 ,
    "megjegyzés" : "Kedves Sara"
  }
}
```

JSON Objects

Tetszőleges számú *név-érték* párból állnak.

- A *név* tetszőleges *string*, az *érték* tetszőleges *JSON érték*.
- A *név-érték* párokra a **tag** (*member*) elnevezést is használjuk.



JSON Objects - példa

JSON objektum

```
{  
  „hallgato” : {  
    „név” :      ”Dóra” ,  
    "fizetés" :   560000,  
    "házas" :    ”férjezett”  
  }  
}
```

JSON objektum karakterláncokkal

```
{  
  "name" : „dora” ,  
  "email" : „dora2022@gmail.com”  
}
```

JSON Objects - számokkal

A JSON *lebegőpontos formátumban* támogatja a számokat.

A szám lehet:

- *egész* (0-9),
- *tört* ((33, .532 stb.) és
- *kitevő* (e, e +, e-, E, E +, E-).

```
{  
  "egész szám" : 34,  
  "tört" : 1.2145,  
  "exponent" : 6.61789e + 0  
}
```

JSON Objects – logikai érték

JSON objektum *logikai elemekkel*

```
{  
  "első" : igaz,  
  "második" : hamis  
}
```

JSON beágyazott objektum

```
{  
  „keresztnev” : „Sara” ,  
  „vezeteknev” : „Kis” ,  
  „eletkor” : 27 ,  
  „cim” : {  
    „utca” : „Feher 2” ,  
    „varos” : „Miskolc” ,  
    „iranyitoszam” : „3515”  
  }  
}
```

Karakterkódolás

RFC (Request For Comments) 8259:

JSON szöveg különböző rendszerek közötti átvitelekor az *UTF-8 karakterkódolást* kell használni.

XML vs. JSON – melyiket használjuk?

Több ezer karakter rögzítése esetén pl.: *5000 karakter XML-be*, akkor a mérleg a JSON felé tolódik (kevesebb adat jut át az egyik szerverről a másikra vagy vissza).

Tehát, az *adat továbbítása szempontjából* a JSON oldalán szól.

A *JSON validálásának* ellenőrzése szintén több URL rendelkezik. Például:

[URL: https://JSONLint.com/](https://JSONLint.com/)

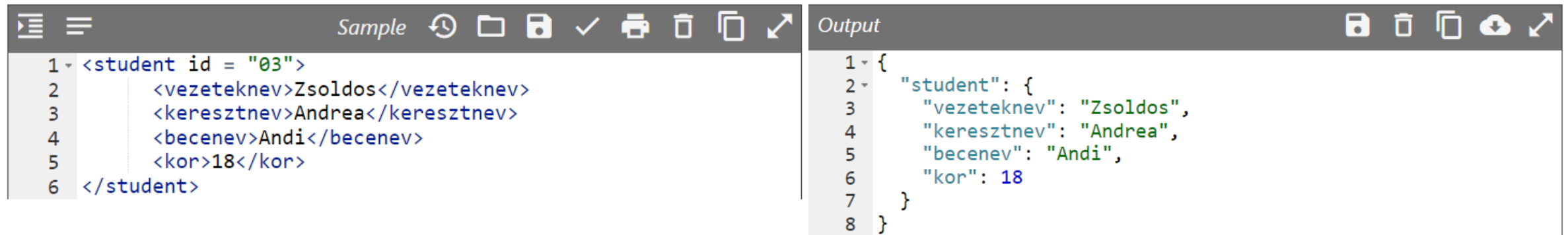
Itt ellenőrizhető a JSON megfogalmazás.

XML vs. JSON

Fontos: az átalakítást mindig egy *parse-l* végezze.

- Önerőből NE írjunk JSON-t – erre valók a *parse-k*.
- *Szintén előnye a JSON-nek*

Készítsünk egy XML-t, majd konvertáljuk át JSON-re egy online konvertálóval.



The screenshot displays an online XML to JSON converter interface. It is divided into two main sections: 'Sample' on the left and 'Output' on the right. The 'Sample' section contains an XML document with the following structure:

```
1 <student id = "03">
2     <vezeteknev>Zsoldos</vezeteknev>
3     <keresztnev>Andrea</keresztnev>
4     <becenev>Andi</becenev>
5     <kor>18</kor>
6 </student>
```

The 'Output' section shows the resulting JSON object after conversion:

```
1 {
2   "student": {
3     "vezeteknev": "Zsoldos",
4     "keresztnev": "Andrea",
5     "becenev": "Andi",
6     "kor": 18
7   }
8 }
```

JSON

A JSON-be a feltűnhet, hogy *számokat* idézőjel nélkül tárolja, ha *szöveggént szeretnénk* értelmezni, akkor „” kell tenni.

Az XML parse-nél külön meg kell mondani.

XML to JSON Converter:

<https://www.freeformatter.com/xml-to-json-converter.html>

XML vs. JSON

Az XML emberi szem számára olvashatóbb (látjuk az elejét és a végét).

Példa: *.html kód* megjelenítése

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

```
<html>
<body>

<h1>Ez a címen</h1>

<p>Ez a paragrafus</h1>

</body>

</html>
```

Ez a címen

Ez a paragrafus

XML to JSON konverter

<http://www.utilities-online.info/xmltojson/?save=bbda4cb9-17d0-4277-9829-4f8219d7dbc2-xmltojson#.Xd-KR-hKhPY>

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="autok5.xsl" ?>
<autok>
  <auto rsz="ABC-100">
    <tipus> Fiat </tipus>
    <ar> 21233 </ar>
    <szin>piros</szin>
    <tulaj>
      <nev>Zoli</nev>
      <varos>Eger</varos>
    </tulaj>
  </auto>
  <auto rsz="ABC-101">
    <tipus> Skoda </tipus>
    <ar> 44233 </ar>
    <szin>fehér</szin>
    <tulaj>
      <nev>Peti</nev>
      <varos>Miskolc</varos>
    </tulaj>
  </auto>
  ..
  ..
```

JSON

```
{
  "autok": {
    "auto": [
      {
        "-rsz": "ABC-100",
        "tipus": " Fiat ",
        "ar": " 21233 ",
        "szin": "piros",
        "tulaj": {
          "nev": "Zoli",
          "varos": "Eger"
        }
      },
      {
        "-rsz": "ABC-101",
        "tipus": " Skoda ",
        "ar": " 44233 ",
        "szin": "fehér",
        "tulaj": {
          "nev": "Peti",
          "varos": "Miskolc"
        }
      }
    ]
  }
}
```

JSON Schema

JSON dokumentumok érvényesítéséhez JSON-alapú sémanyelvet használunk.

- Webhely: <https://json-schema.org/>
- Aktuális verzió: 2020. 12.
- A legszélesebb körben támogatott verzió a 2018-ban kiadott draft-07 verzió.

JSON Schema

A JSON-séma egy *deklaratív nyelv*, amely lehetővé teszi a JSON-dokumentumok *annotációját* és *validálást*.

Előnyei:

- Leírja a meglévő data format(s).
- Ember és gép által olvasható dokumentációt biztosít.
- Ellenőrzi azokat az adatokat, amelyek:
 - Automatizált tesztelés.
 - Kliens által használt adatok minőségének ellenőrzése.

BSON - Binary JSON

BSON - (*bee – sahn*) bináris *adatcsere* formátum.

- Specifikáció: <http://bsonspec.org/>

A *MongoDB* NoSQL adatbázis-kezelő rendszer használja.

URL: <https://www.mongodb.org/>

- *Adattárolás és hálózati adatátvitel.*

BSON { 01000010
01010011
01001111
01001110 }

<https://bsonspec.org/>

BSON - Binary JSON

A JSON adattípusainak kiterjesztése:

Pl.: dátum típus, időbélyeg, BinType, ...

- Nincs azonban *number* adattípus, helyette *int32*, *int64* és *double* adattípusok használata.

- *Basic types*

<code>byte</code>	1 byte (8-bits)
<code>int32</code>	4 bytes (32-bit signed integer, two's complement)
<code>int64</code>	8 bytes (64-bit signed integer, two's complement)
<code>uint64</code>	8 bytes (64-bit unsigned integer)
<code>double</code>	8 bytes (64-bit IEEE 754-2008 binary floating point)
<code>decimal128</code>	16 bytes (128-bit IEEE 754-2008 decimal floating point)

BSON - Binary JSON

BSON is támogatja a *dokumentumok és tömbök beágyazását* más dokumentumokba és tömbökbe.

BSON jellemzői:

- *Könnyűsúlyú:* hálózati adatátvitel.
- *Átjárható:* MongoDB használják.
- *Hatékony:* kódolás és dekódolás gyorsan végrehajtható, mert C nyelv adattípusait használja.

JSON vs. BSON közötti különbségek



JSON	BSON
A JSON a JavaScript objektum jelölése.	A BSON egy bináris Javascript objektum jelölés.
Ez egy szabványos fájlformátum.	Ez egy bináris fájlformátum.
A JSON néhány alapvető adattípust tartalmaz, pl.: karakterláncot, számokat, logikai értéket és nullát.	A BSON tartalmaz néhány további adattípust, pl. dátumot, időbélyeget stb.
Adatbázisok, mint az <i>AnyDB</i> , a <i>Redis</i> stb., JSON formátumban tárolják az adatokat.	A <i>MongoDB</i> adatait BSON formátumban tárolják.
A JSON kevesebb helyet igényel a BSON-hoz képest.	A BSON több helyet igényel, mint a JSON.

JSON vs. BSON közötti különbségek



Viszonylag kevésbé gyorsabb, mint a BSON.	Gyorsabb a BSON-hoz képest.
Adatátvitelre szolgál.	Adatok tárolására szolgál.
JSON-fájlból való olvasás során, <i>végig kell mennie a teljes tartalma.</i>	A BSON-ban az indexet használják,
A JSON formátumot nem kell elemezni, mivel az már ember által olvasható.	Elemezni kell, mivel a gépek könnyen értelmezhetik.
A JSON objektumok és tömbök kombinációja, ahol az objektum kulcs-érték párok gyűjteménye, míg a tömb az elemek rendezett listája.	A BSON további információkat nyújt, pl.: a karakterlánc hosszát és az objektum altípusait. A BinData és a dátum a BSON által támogatott további adattípusok.

GSON – Google JSON

A GSON egy nyílt forráskódú Java-könyvtár, amely Java objektumokat JSON-ba szerializál és deszerializál.



- A GSON a Google JSON *parser* és *Java generátor*.
- A Google belső használatra fejlesztette ki a GSON-t, de később nyílt forráskódú.

Download Gson Archive

- Download: `gson-2.3.1.jar` – Javában használóknak.

GSON User Guide – hasznos információk

<https://sites.google.com/site/gson/gson-user-guide>

BOON

- A Boon egy Java alapú eszköz, amellyel a JSON adatokat *hatékonyan és gyors módon* kódolható vagy dekódolható.
- Használhatjuk a Boon JSON parser-t, ha befoglaljuk a *Boon JAR* fájlt a Java alkalmazásba.

További hasznos információ:

https://www.tutorialspoint.com/boon/boon_quick_guide.htm

<https://jenkov.com/tutorials/java-json/gson-installation.html>

BOON

Download Boon Archive

- *Download:* boon-0.34.jar

URL: <https://mvnrepository.com/artifact/io.fastjson/boon>

JSON kérdések

Több kérdés is felmerülhet a JSON-nél?

- Kérdés van egyszerűbb megoldás a JSON-nél?
- Miért kellene még mindig operátorok?
- Miért nem lehet e nélkül elkészíteni a feladatot?

Válasz: LEHET

YAML formátum

YAML formátum (YAML - Nem Markup Language), amely egy újabb módja az *adatok tárolásának*.

A fájl kiterjesztése: `.yaml`

A YAML egy adat **sorosító nyelv** (szerializáció), - közvetlenül *olvasható és írható* emberi szemmel.

Célja: a memóriában tárolt adatok egyszerű lemezre mentése és visszatöltése.

YAML formátum - szerializáció

A sorosítás során létrehozunk egy állományt és egy sorosítást kezelő objektumot.

A kiválasztott objektumban tárolt adatok az állományba kerüljenek sorban egymás után.

Láncolt lista és körkörös hivatkozások kezelésére is képes.

JSON - YAML formátum konvertálás

JSON-ről YAML-ra is van lehetőség konvertálni.

<https://www.json2yaml.com/>

```
{  
  "Person": {  
    "name": „Lilla”,  
    "age": 20  
  }  
}
```

```
Person:  
  name: Lilla  
  age: 20
```

A 3 kötőjel jelzi a fájl kezdetét.

Legnagyobb különbség a formátumok között az a nyelvtani.

Mindegyik hűen ábrázolja egy objektum struktúráját.

YAML formátum jellemzői

- Az `.yaml` fájlok '---' 3 kötőjellel kezdődnek, jelezve a dokumentum kezdetét.
- A kulcsérték-párokat : *kettőspont* választja el.
- A listák - **kötőjellel** kezdődnek.

Validálása

URL: <https://codebeautify.org/xml-to-yaml>

Konvertáló

URL: <https://codebeautify.org/xml-to-yaml>

XML - YAML konvertálás - mintapélda

```
Sample
1 <class>
2
3   <student id = "01">
4     <vezeteknev>Fekete</vezeteknev>
5     <keresztnev>Peter</keresztnev>
6     <becenev>Petya</becenev>
7     <kor>22</kor>
8   </student>
9
10  <student id = "02">
11    <vezeteknev>Kek</vezeteknev>
12    <keresztnev>Dora</keresztnev>
13    <becenev>Dorka</becenev>
14    <kor>20</kor>
15  </student>
16
17  <student id = "03">
18    <vezeteknev>Zsoldos</vezeteknev>
19    <keresztnev>Andrea</keresztnev>
20    <becenev>Andi</becenev>
21    <kor>18</kor>
22  </student>
23 </class>
```

Ln: 6 Col: 18 size: 587 B

```
Output
1 class:
2   student:
3     - vezeteknev: Fekete
4       keresztnev: Peter
5       becenev: Petya
6       kor: 22
7     - vezeteknev: Kek
8       keresztnev: Dora
9       becenev: Dorka
10      kor: 20
11     - vezeteknev: Zsoldos
12       keresztnev: Andrea
13       becenev: Andi
14       kor: 18
15
```

Ln: 15 Col: 0 size: 280 B

JSON - YAML konvertálás - mintapélda

```
Sample
1 {
2   "class": {
3     "student": [
4       {
5         "vezeteknev": "Fekete",
6         "keresztnev": "Peter",
7         "becenev": "Petya",
8         "kor": 22
9       },
10      {
11        "vezeteknev": "Kek",
12        "keresztnev": "Dora",
13        "becenev": "Dorka",
14        "kor": 20
15      },
16      {
17        "vezeteknev": "Zsoldos",
18        "keresztnev": "Andrea",
19        "becenev": "Andi",
20        "kor": 18
21      }
22    ]
23  }
24 }
```

Ln: 24 Col: 1 size: 485 B

```
Output
1 class:
2   student:
3     - vezeteknev: Fekete
4       keresztnev: Peter
5       becenev: Petya
6       kor: 22
7     - vezeteknev: Kek
8       keresztnev: Dora
9       becenev: Dorka
10      kor: 20
11     - vezeteknev: Zsoldos
12       keresztnev: Andrea
13       becenev: Andi
14       kor: 18
15
```

Ln: 15 Col: 0 size: 280 B

JSON - TEXT konvertálás - mintapélda

```
Sample
1 {
2   "class": {
3     "student": [
4       {
5         "vezeteknev": "Fekete",
6         "keresztnev": "Peter",
7         "becenev": "Petya",
8         "kor": 22
9       },
10      {
11        "vezeteknev": "Kek",
12        "keresztnev": "Dora",
13        "becenev": "Dorka",
14        "kor": 20
15      },
16      {
17        "vezeteknev": "Zsoldos",
18        "keresztnev": "Andrea",
19        "becenev": "Andi",
20        "kor": 18
21      }
22    ]
23  }
24 }
```

Ln: 24 Col: 1 size: 485 B

```
Output
1 vezeteknev keresztnev becenev kor
2 Fekete Peter Petya 22
3 Kek Dora Dorka 20
4 Zsoldos Andrea Andi 18
5
```

Ln: 5 Col: 0 size: 112 B

YAML formátum

A YAML formátummal *gyorsan tudunk adatokat továbbítani* az interneten az egyik helyről a másikra.

Közvetlenül is tudunk *XML-ről YAML konvertálni*:

<https://jsonformatter.org/xml-to-yaml>

<https://www.json2yaml.com/>

<https://codebeautify.org/xml-to-yaml>

JSON – JAVA, JAVACRIPT, PHP, AJAX

„A különböző programozási nyelvek lehetőséget adnak *a JSON készítésére/konvertálására*.

- **JSON with Java**
- JSON with PHP
- JSON with Python
- JSON with Perl
- JSON with Ruby
- **JSON with JavaScript**

Java JSON

A `json.simple` könyvtár lehetőséget biztosít, hogy *JSON-adatokat olvassunk és írjunk Java-ban*.

Tehát, *kódolhatjuk és dekódolhatjuk* a JSON objektumot a Java-ban a `json.simple` könyvtár használatával.

Java JSON – package instal

Instal *json.simple.jar*

A *json.simple* telepítéséhez be kell állítania a *json.simple.jar* környezeti változót, vagy hozzá kell adnia a Maven függőséget.

URL: <http://www.java2s.com/Code/Jar/j/Downloadjsonsimple11jar.htm>

2. A maven dependency hozzáadásához a következő kódot kell az *pom.xml* fájlba beírni.

<https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple>

Java JSON API

Az `org.json.simple 2.1.2.jar` csomag a `JSON API` fontos osztályait tartalmazza.

Hasznos információk:

URL: <https://stleary.github.io/JSON-java/index.html>

Java JSON API

Az `org.json.simple 1.1.1.jar` csomag a `JSON API` fontos osztályait tartalmazza.

- `JSONValue`
- `JSONObject`
- `JSONArray`
- `JSONString`
- `JSONNumber`

URL: <https://stleary.github.io/JSON-java/index.html>

Java JSON API - JSONObject

JSONObject kulcs- és értékpárok rendezetlen gyűjteménye.

Támogatott főbb metódusok:

- *get(String key)* – megkapja a kulccsal társított objektumot,
- *opt(String key)* – megkapja a kulccsal társított objektumot, különben *null*,
- *put(String key, Object value)* – beszúr vagy lecserél egy kulcs-értékpárt az aktuális *JSONObject*-ben.

Java JSON API – JSONObject - példa

A *put()* metódus argumentumában megadhatjuk a *kulcsot* és az *értéket*:

```
JSONObject bl = new JSONObject();
```

```
// put() metódust meghívása
```

```
bl.put("name", „LL”);
```

```
bl.put(„kor”, "22”);
```

```
bl.put(„város”, „Miskolc”);
```

```
System.out.println(bl);
```

Java JSON API – JSONObject Map osztály

Létrehozunk egy *MAP*, majd argumentumként átadhatjuk a *JSONObject* konstruktorának – példa *(eredmény un.)*:

Ehhez be kell importálni:

```
import java.util.Map;  
Map<String, String> map = new HashMap<>();  
map.put("name", „LL”);  
map.put(„kor”, "22");  
map.put(„város”, „Miskolc”);  
JSONObject ll = new JSONObject(map);
```


Java JSON API – JSONArray

JSONArray az értékek rendezett gyűjteménye.

Az értékek lehetnek: *szám, karakterlánc, logikai érték, JSONArray, JSONObject, JSONObject.NULL* etc.

JSONArray van egy konstruktora, amely fogad egy karakterláncot, és elemzi azt *JSONArray* létrehozásához.

Java JSON API – JSONArray

JSONArray osztály *metódusai*:

- *get(int index)* – a megadott index értéket adja vissza (0 és teljes hossz – 1 között),
- *opt(int index)* – az indexhez tartozó értéket adja vissza (0 és teljes hossz – 1 között).
- *put(Object value)* – objektumérték hozzáfűzése a *JSONArray*-hez.

Java JSON API – JSONArray - példa

Létrehozunk egy *JSONArray* objektumot, majd hozzáadunk és lekérhetünk elemeket a *put()* és *get()* metódusokkal:

```
JSONArray kk = new JSONArray() ;
```

```
kk.put(Boolean.TRUE) ;
```

```
kk.put („szöveg”) ;
```

```
JSONObject k1 = new JSONObject() ;
```

```
k1.put("name", „LL”) ;
```

```
k1.put („kor", "22”) ;
```

```
k1.put („varos", „miskolc”) ;
```

```
kk.put(k1) ;
```

Java JSON - mintapélda

Adott a következő XML dokumentum: *XMLNeptunkod.xml*
k:/ Java munkafajlok/XML/JSON/XmlNeptunkod.xml

Java JSON - mintapélda

Adott a következő XML dokumentum: *XmlNeptunkod.xml*
–konvertálja át

k:/ Java munkafajlok/XML/JSON/XmlNeptunkod.json

```
{
    "student": {
        "nev": "1Laszlo",
        "kor": 21,
        "fizetes": 1000000,
        "allitas": true
    }
}
```

Java JSON – JSON kódolás - mintapélda

Írjon egy Java programot, mely a JSON objektumot JSON string kódolja a Java-ba – JSON munka mappába.

k:/ Java munkafajlok/XML/JSON/JsonNEPTUNKOD.java

Java JSON tömbkódolás a List használatával

Írjon egy Java programot, mely JSON tömböt kódolja a *Java List* segítségével – JSON tömbbe.

URL: <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

K:/ Java munkafajlok/XML/JSON/JSONArrayList.java

JSON string dekódolása a Java-ban

Írjon egy Java programot, mely *JSON string dekódolását* végzi Java-ban.

K:/ Java munkafajlok/XML/JSON/JSONDecode.java

JSON tömb dekódolása Java-ban

Példa: a **JSONObject**-et és a **JSONArray**-t használja, ahol a

- JSONObject egy java.util.Map,
- a JSONArray, pedig egy java.util.List,

így a *Map* vagy *List* szabványos műveleteivel érheti el őket.

Adott a következő karakterlánc:

```
String s = "[0,{\"1\":{\"2\":{\"3\":{\"4\":[5,{\"6\":7}]}}}}],„
```

Írattassa ki a tömb 2. elemét ill. a mező 1. elemét.

JSON tömb dekódolása Java-ban

Írjon egy Java programot, mely JSON tömb dekódolja a *Java* segítségével, ahol JSONObject (java.util.Map) és JSONArray (java.util.List)

K:/Java munkafajlok/XML/JSON/JSONDecodeNEPTUNKOD.java

Írattassa ki

- a tömb 2. elemét ,
- a mező 1. elemét.
- {},
- [5]
- [5,2]

JSON file olvasása Java-ban

Készítsen egy Java programot, amely egy *JSON dokumentumot olvas be* és a feldolgozás után megjeleníti a konzolon.

Használja a `JSONParser`, a `JSONObject` és a `JSONArray` osztályokat.

Hasznos infók: <https://www.geeksforgeeks.org/iterate-map-java/>

A JSON dokumentum neve: *vizsgakNeptunkod.json*

Package neve: neptunkod

Projekt nev: JSONParseNeptunkod

Class name: JSONReadNeptunkod

JSON file írása Java-ban

Készítsen egy Java programot, amely egy JSON dokumentumot *ír ki egy fájlba és a konzolra.*

Használja a `JSONObject` és a `JSONArray` osztályokat.

Package neve: neptunkod

Projekt nev: JSONParseNeptunkod

Class name: JSONWriteNeptunkod

File name: vizsgak1Neptunkod.json

JSON használata JavaScriptben

A JSON a *JavaScript objektumok* formátuma.

- A `parse()` metódus használható a *JSON szöveg JSON objektumra* történő konvertálására.
- A `toJSONString()` metódus használható a *JSON objektum JSON szövegre* történő konvertálására.

```
var myJSONObject = {"bindings": [  
    {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},  
    {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},  
    {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}  
    ]  
};
```

```
var myObject = eval('(' + myJSONtext + ')');
```

JSON – JavaScript dokumentum

Adott a következő XML dokumentum! *xmlNeptunkod.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <element>
    <author>Nyékyné Dr. Gaizler Judit</author>
    <language>Java 2 I-II. - Útikalauz programozóknak 5.0</language>
  </element>
  <element>
    <author>Reiter István</author>
    <language>C# programozás lépésről lépésre</language>
  </element>
</book>
```

JSON - JavaScript

Konvertálja át a *xmlNeptunkod.xml* dokumentumot
xmlNeptunkod.json dokumentumra.

```
{  
  "book": [  
    {  
      "language": "Java XML and JSON",  
      "author": "Nyékyné Dr. Gaizler Judit"  
    },  
    {  
      "language": "C# programozás lépésről lépésre",  
      "author": "Reiter István"  
    }  
  ]  
}
```

JSON megvalósítása - JavaScript

D:\Java munkafajlok\XML\JSON

A *JSONNeptunkod.json* kódot írjuk meg JavaScript-be.

Mentés: *xmlNeptunkod.html*

JSON megvalósítása - JavaScript

```
≡ <script language = "javascript" >

    document.write("<h2>JSON megvalósítása - JavaScript</h2>");
    var object1 = {"language": "Java 2 I-II. - Útikalauz programozóknak 5.0", "author" : "Nyékyné Dr. Gaizler Judit"};

    document.write("<h3>Language: " + object1.language+"</h3>");
    document.write("<h3>Author: " + object1.author+"</h3>");

    var object2 = { "language": "Adatkezelés XML környezetbe", "author" : "Dr. Kovács László" };
    document.write("<h3>Language: " + object2.language+"</h3>");
    document.write("<h3>Author: " + object2.author+"</h3>");

    document.write("<hr />");
    document.write(object2.language + " OOP, " + "a könyv szerzője: " + object2.author);
    document.write("<hr />");
</script>
```

JSON parse - Text JS objektummá konvertálás

`JSON.parse()` használjuk – *JSON Text-be* megírt adatok JS objektummá kerülnek elemzésre.

Írjon egy JS programot, amely, egy adott TXT-t konvertálja át *JS objektummá*:

```
'{"name": "BB", "code": 3515, "city": "Miskolc-  
Egyetemváros"}'
```

JSON parse - JSON tömbbe megírt adatok JS tömbbe kerülnek elemzésre

D:/Java munkafajlok/XML/JSON/karakttoJSarray.html

Írjon egy JS kódot, amely a JSON tömböt, JS tömbbé konvertálja.

Adott a következő JSON tömb. Írattassa ki a tömb 3. elemét!

```
' [ "Toyota", "Trabant", "WW", "KIA" ] ,
```

Eredmény:

JSON tömbbe megírt adatok JS tömbbe kerülnek elemzésre

KIA

JSON parse - Dátumok elemzése

A dátum objektumok nem engedélyezettek a JSON-ban.

Ha dátumot kell megadnia, írja be *karakterláncként*, később konvertálhatjuk a **dátumot objektummá**.

Írjon egy JS kódot, amely egy adott *karakterláncot konvertál dátummá*.

Adott a következő karakterlánc

```
'{"name":"BB", "birth":"2000-11-10",  
"code":3515, "city":"Miskolc-Egyetemváros"}';
```

JSON parse - Függvények elemzése

A függvények nem engedélyezettek a JSON-ban.

Ha függvényt kell beillesztenie, írja be karakterláncként, majd később vissza konvertáljuk dátummá.

Írjuk be a code helyére: `"code": "function() {return 3515;}"`

Karakterlánc konvertálása függvénnnyé

```
'{"name": "BB", "birth": "2000-11-10",  
"code": 3515}', "city": "Miskolc-Egyetemváros"}';
```

JSON `stringify()` használata

A JSON elterjedt használata az adatok cseréje
webszerverrel/webszerverről.

*Amikor adatokat küldünk egy webszervernek, az adatoknak
karakterláncnak kell lenniük.*

*Alakítson át egy JavaScript-objektumot - sztringgé a
JSON.stringify() segítségével.*

JS object konvertálása JSON karakterláncná - JSON.stringify().

JSON objektum konvertálása JSON karakterlánccá

D:/Java munkafajlok/XML/JSON/JSobjecttokarakt.html

Adott a következő JSON objektum:

```
{name:"BB", code:3515, city:"Miskolc-  
Egyetemváros"};
```

Írjon egy kis programot, amely a *JSobjektum*-ot konvertálja JSON string-é – használja a *JSON.stringify()*

Felhasznált irodalom

- Kovács László: Adatkezelés XML környezetbe
<http://moodle.iit.uni-miskolc.hu/login/index.php>
- JSON Tutorial
<https://www.javatpoint.com/java-json-example>
- JSON Basics Tutorial
<https://www.tutorialspoint.com/json/index.htm>