

1. Jellemző a DOM szabványt, ismeretese a működését a modell alapján (rajz is) a feldolgozás lépéseit, sorolja fel a DOM API interfészeit (adattagok, módszerek) funkcióit.

Document Object Modell - XML olvasó és író, fő objektum az SAX (Simple API for XML) API-ra épül - de az csak olvas. Befutást olvas be és így lehet módosítani, lassabb mint a SAX

Feldolgozás:

- beolvasás az XML-t és az egészét átolvassa - emiatt lassabb
- ~~- minden egy sort is megvár, begyűjt valamennyit (buffer)~~
- már a memóriában lévő forrást lehet módosítani

Interfészei:

Document: a beolvasott dokumentum

Element: egy elem

NodeList:

DOMException: hibahérvétel

Node: általános fa

~~DOM~~ - DOCUMENTBUILDER ^{FACTORY} létrehozása

- dokumentum betöltése
- utána lehet olvasni, írni
- doc. Document Element → root

Működés: XMLSerialization - el binarizáljuk az objektumot

2, Milyen szolgáltatásokat biztosít az XPath, nevére meg jellemzőre a kifejezések, azok adattípusait és operátorait, relatív és abszolút elérési vonal (példa is), valamint az XPath függvényeit és leírásait

XPath - az XML-ben való keresésre szolgál, fő aljektum, egy részlet ad vissza

a függvények art jelöli, hogy "merre induljon el" a keresés:

pl.: child, ~~ancestor~~ ^{előző} ~~parent~~ ^{szülő}, sibling, self - maga, descendant - ^{előző} ~~child~~ ^{utód},
 gyereke ~~common~~ testvér-node ^{csúspont}
 descendant-of-self - ^{előző} ~~valam~~ ^{ö maga}, parent: szülő,


rövidítések:

child ::

attribute ::

descendant-or-self::node()

self::node()

 összes

@

operatorok: +, -, *, /, %

functions: sum(), min(), max()

3, Mi az XML Schema szerepe az adathierarchia leírásában. Ismertesd az XML Schema elemi és összetett típusait, számszerűsítést, hivatkozások használatát - példákkal. Névtérrel kapcsolatban.

Az XML Schema korlátokra a megvalósuló nyelvvel, azaz a DTD-hoz hasonló csak itt XML formátumban, nem SGML illetve támogatja az adattípusokat és a névtérrel kapcsolatos névtérrel és absztrakt típusok is vannak benne

```
<xs:element name="nincs" type="pl:nev-tip">  
  <xs:key name="" />  
</xs:element>
```

Elemi típusok: csak egy értéktartomány leírhatása.

pl.: integer 1-100, vagy max 7 jegyű string - random

```
<xs:simpleType name="random">
```

```
<!-- ... -->
```

```
</xs:simpleType>
```

Összetett típusok: pl.: cím típus, bizonyult több elemi típusból

```
<xs:element name="cím">
```

```
  <xs:simpleType name="irsz" type="int">
```

```
  <xs:simpleType name="utca" type="string">
```

```
  ...
```

```
</xs:element>
```

Számszerűsítés: simpleType - megjelöl típus értéktartományának leírására

4, Jellemző a JSON adatmodellre, szerkezet, adattípus, összehasonlítás xml-ján. Mi a YAML formátum!

JSON - egyszerűbb, mint az XML, könnyebben olvasható.

Főleg a internet elterjedése miatt egy könnyebb XML helyettesítő.
~~Pl.:~~ ~~XML~~ ~~helyettesítő~~

Pl.:
{
 "név": "Borgomni",
 "kor": "24",
}

~~helyettesítő~~

az XML több felesleges adatot is tartalmaz, pl. zárótagok.

így a JSON fájl kisebb, tömörebb

adattípusok:

- új elem az XML-hez képest: tömb: []
- szöveg: név, életkor, stb.
- struktúra: { }

alapvetően mindezt stringként kell megadni
minden objektumnak lehet ID-ja

Főleg a weben használják

5, Sorolja fel az infoSet modell információk elemeinek típusait,
melyek valójában 3 típusú és névvelésen jellemző
XInclude rendszer cíkjai, jelölés elemek.