1. World Wide Web

• ***Alapfogalmak:***

- erőforrás: minden ami URI-val azonosítható (információ erőforrás minden lényeges jellemzője továbbítható egyetlen üzenetben)

- reprezentáció: erőforrás állapotáról információkat kódoló adatok

- tartalomegyeztetés: erőforráshoz több reprezentáció kínálása, legjobb kiválasztása

- hivatkozás feloldás: URI használata hivatkozott erőforrás eléréséhez

formái: reprezentáció letöltése, létrehozása, módosítása, néhány vagy összes reprezentáció törlése

- felhasználói ágens: személy nevében cselekvő szoftver

• ***Szabványok fajtái:***

- de facto: gyakor használatból, piaci elfogadottságból származnak. Pl: QWERTY kiosztás

- de jure: helyi / állami / nemzetközi szintű szabályok által előírt szabványok. Pl: SI m.

- önkéntes közmegjegyzés: magánintézmények által meghatározott szabványok. Pl: TCP/IP protokollkészlet

• ***Szabványokért felelős szervezetek:***

IANA: *Internet Assigned Numbers Authority*

- internet működése alapjául szolgáló kódok / számok kiosztása
- DNS gyökérzóna felügyelete, karbantartása(.inf, .arpa)
- IP címek kiosztásának globális koordinálása
- nyilvántartás internet protokollokhoz használt kódokról, számokról

IETF: *Internet Engineering Task Force*

- internet szabványokat fejlesztő nemzetközi szabványügyi szervezet
- TCP/IP protokollkészlet fejlesztése
- bárki tagja lehet
- munkacsoportokban dolgoznak

RFC sorozat: *Request for Comments*

- internetről szóló műszaki és szervezeti dokumentumokat tartalmaz
- folyamatai:
    - **IETF** – Internet Engineering Task Force
    - **IAB** – Architecture Board
    - **IRTF** – Research Task Force
    - **Független beadványok**

Kiadott RFC-k soha nem módosulnak, hibákat javítják

- IETF RFC-k alsorozatai:

- **BCP**: *Best Current Practice*: irányelveket, folyamatokat dokumentálnak
  BCP index
- **STD**: *internet standard*
  STD index
- BCP-k és STD-k alsorozatokan számot kapnak, de RFC-k is megtartják
- BCP és STD több RFC-hez is tartozhat

Standard track: RFC-k érettségi szintjei

- Proposed standard
- Draft standard
- Internet standard

Internet-Draft: egyfajta demo verzió, fejlesztési céllal teszik elérhetővé
nem feltétlen kerülnek publikálásra
legfeljebb 3 hónapig érvényesek

W3C – Word Wide Web Consortium:

- webszabványok fejlesztésén dolgozó nemzetközi közösség, technológiákat meghatározó és szabványnak számító dokumentumokat publikálnak (ajánlások)
- nyílt szabványok alapelvei:
  - web mindenkinek: web elérése mindenki számára, mindentől függetlenül
  - web mindenhol: web eszközfüggetlen elérése
- szakmai jelentések éretségi szintjei:
  - **Working Draft**: munkaterv áttekintési céllal publikált dokumentum
    nem minden WD célja hogy ajánlás legyen
  - **Candidate Reccomendation** (tapasztalatszerzés): már áttekintett dokumentum, aktuális cél az előzetes javaslat
  - **Proposed Reccomendation** (javaslatterv): megfelelő minőségű ahhoz, hogy ajánlássá váljon
  - **Recommendation** (ajánlás): széles körben alkalmazott webszabvány
  - **Working Group Note**: dokumentum, melyet nem szánnak REC-nek, félbehagyott munkát dokumentál

- túlhaladott ajánlás: Superseded REC – újabb verzió váltotta fel
- elavult ajánlás: Obsolete REC – nincs elegendő piaci jelentősége, hogy a továbbiakban is implementálásra ajánlja a W3C

WHATWG: Web Hypertext Application Technoloy Working Group

- a közösség célja a HTML és hozzá kapcsolódó technológiák fejlesztése

2. Unicode:

Univerzális karakterkódolási szabvány, lefedi a világ összes modern és ősi nyelvét, egyéb írott szövegben használt szimbólum is része

- **Kódtér**: karaktereket kódoló egész számok tartománya
- **Kódpont**: egy adott karaktert kódoló egész szám a kódtérben
- **BMP**: Basic Multilingual Plane: első 2^{16} kódpontot (65536) tartalmazó sík (U+0000 – U+FFFF)

- **Karakterkódolások:**

- UTF-8:

- ábrázolás 1-4 byte (változó szélességű)
- U+0000 – U+FFFF tartomány 1 byte-on
- U+0080 – U+0FFF tartomány 2 byte-on
- BMP többi kódpontja 3 byte
- BMP kívüli kódpontok 4 byte

- UTF-16:

- minden kódpont 2 vagy 4 byte-on ábrázolt (változó szélességű) BMP karakterei 2, összes többi 4 byte-on
- hatékony feldolgozhatóság, hatékony tárhelyhasználat

- UTF-32:

- minden kódpont ábrázolása 4 byte-on (rögzített sélességű)
- leghatékonyabb feldolgozás, legnagyobb tárhelyigény

- ISO/IEC 8859:

- 8 bites szabványok
- 8859-1 Latin 1 – Ny-EU nyelvek
- 8859-2 Latin 2 – Közép EU nyelvek

- **Karakterek kifejezése:**

- CSS: \hhhh

- 6 hexadecimális karakter
- 6-nál kevesebbnél whitespace kell a végére

- JSON: \unnnn

- 4 hexadeximális karakter

• **XML/HTML:**

- #nnnn – 4 decimális
- #xhhhh 4 hexadecimális

3. Média típusok

Internet protkollon keresztül továbbított tartalmak formátumának jelzése

• **Felső szintű típus:**

- kisbetű-nagybetű érzéketlen

- application: többi felső szintű típusba nem tartozó alkalmazói programok által feldolgozott adatok részére

- audio

- font

- image

- message, multipart: összetett típusok, külfönféle média típusú objektumok egységbe zárását teszi lehetővé

- model: fizikai modell reprezentációk típusai

- text: karakterkódolás meghatározása, charset paraméter

- video

• **Altípusok**: nevének elején **fa** előtag határozza meg a regisztrációs fát használat körétől függ melyik fába regisztrálódik

Regisztrációs fák:

1. szabványok : *standards*
2. gyártói: *vendor*
3. személyes: *personal / vanity*
4. nem regisztrált: *unregistered x tree*

1. internet közösség érdeklődésére szánt média típusok

pl. application / gzip

2. Nyilvánosan elérhető termékekhez kötődő médiatípusok

pl. application / vnd.ms-excel

3. Kísérleti, nem üzletszerűen terjesztett termékek

pl. audio / prs.sids

4. x. előtaggal kezdődik, privát módon használható lokális környezetben

- **Paraméterek:**
  - kisbetű-nagybetű érzéketlen
  - tetszőleges sorrendben mind max egyszer adható meg
  - értékeikre nincs előírt szintaxis
  - médiatípus regisztráció határozza meg a nevét és értékeit
- **Struktúrált szintaxis utótag:**
  - altípus: nevének a '+' karaktert követő része
  - médiatípus szerkezetét jelzi
  - ezeket is regisztrálni kell

  pl. +ber, + cbor, +der, +fastinfoset
- **Regisztrációs követelmények:**
  - meghatározhatja az alkalmazások számára az erőforrásrész-azonosító értelmezését
  - SSS regisztrációja meghatározhatja a szintaxist: alkalmazó média típusokhoz az erőforrásrész-azonosító feldolgozásának módját
- **Regisztrációs folyamat**: IANA adminisztrálja

4. URI

- URI: Uniform Resource Idenifier: weben használt globális azonosító.
- Absztrakt vagy fizikai erőforrást azonosító karaktersorozat
- URI-t akár tárgyi világ objektumaihoz is lehet rendelni.
- Kinézete sémanév, sémaspecifikus rész → szintaxisát sémaspecifikáció határozza meg
- IANA adminisztálja az URI sémákat.

- **URI sémák**: file, http:, about:
- **URI karakterek:**
  - fenntartott: / ? # [ ] @ ! ? $ & ' ( ) * + , ; =
  - nem fenntartott: A-Z, a-z, 0-9, - . _ ~

  karakterkódolást nem határoz meg

- **Százalékos kódolás:**
  - nem megengedett karakter használatához vagy fenntartott karakterek speciális jelentésének elnyomásához

  Lépései:

  1. karaktert ábrázoló oktet sorozatot tekintjük
  2. ezt a sorozatot karakterlánccá kódoljuk, melyben minden oktettet %HH módon ábrázolunk (2 hexadecimális számjeggyel)
- **Egyszerűen: fenntartott karakterek helyetesítése %HH alakban**

• **URI szintaxis:**

- általános szintaxisa: séma: hierarchikus rész [? lekérdezés] [# erőforrásrész]

- hierarchikus rész authority és path komponenseket tartalmazhat
// útvonal
- ha nincs authority , a path üres kell legyen vagy /-el kezdődnie

- authority: az URI további része fennhatóság alá tartozik

[userinfo '@'] host [:port] - az URI sémák meghatároznak egy portot

- path: részeit / választja el egymástól

- első ***'?'*** vagy ***'#'*** karakterig tart, ha nincs akkor URI végéig
- állományrendszerekben ***'.'*** és ***'..'*** is használható path részeként

- lekérdezés komponensei: eleje **'?'** , vége **'#'** vagy URI vége

- nem hierarchikus adatokat tartalmaz
- általában név=érték formájú, a párokat **'&'** választja el

- erőforrásrész-azonosító: eleje **'#'**, URI végéig tart

- lehetővé teszi a másodlagos erőforrás közvetett azonosítását
(másodlagos erőforrás lehet az első része)
- jelentését az elsődleges erőforrás elérése során kapott reprezentációk határozzák meg
- médiatípusok meghatározhatják az erőforrásrész azon formáját, az így azonosított másodlagos jelentését

- erőforrás azonosító jelentése:

• **Text / html típus:**

- dokumentum adott részét jelenti, állapotinformációt szolgáltat scriptek számára
Pl: https://…#t-77 → yt videó 77 mp-től kezdődik

Application / XML, TEXT / XML
X pointer Framework specifikációja adja a szintaxist és jelentést

• **URI-hivatkozás**: URI vagy relatív hivatkozás

Relatív hivatkozás adott környezetben értelmezett
bázis URI alapján URI formálható belőle
Feladásához algoritmust ad meg a specifikáció

• **Abszolút URI**: nem tartalmaz erőforrásrész azonosítót

Bázisként csak abszolút URI használható

- **Relatív hivatkozás feloldása:**

Bázis: **http:example/a/b/c?q**

Relatív	Eredmény
d	http://example/a/b/d
./d	http://example/a/b/d
/d	http://example/d
//localhost	http://localhost
?y	http://example/a/b/c?y
d?y	http://example/a/b/d?y
#z	bázis #2
" "	bázis
.	http:example/a/b/
./	http:example/a/b/
..	http:example/a/
../d	http:example/a/d
../../d	http:example/d/

- **URI-k összehasonlítása:**
  - Séma: Host komponensek kis-nagybetű érzéketlenek
  - Többi komponensre érzékenységet kell feltételezni, kivéve ha mást ír a séma
  - Ekvivalencia def: két URI ekvivalens ha ugyanazt az erőforrást azonosítják
  - Gyakorlatban az ekvivalencia összehasonlításra alapul
    összehasonlítás során normalizálás (nagybetű átalakítása érzéktlen komponensekben)

5. HTTP

- **Állapot nélküliség**: egymást követő kérések független kezelése
- **Kiterjeszthetőség**: metódusok, állapotkódok, fejlécmezők
- **http: 80**-as port, **https: 443**-as port alapérelmezés szerint

- **Szintaxis**: http/https :// host[: port][útvonal][? lekérdezések]

• **Üzenet absztrakció**: több verziót átfogó általánosítás

üzenet alkotóelemei:

- vezérlő adatok
- fejléc szakasz
- tartalom
- lezáró szakasz

Vezérlő adatok	• Üzenet elsődleges céljukat leíró vezérlő adatok • Kérésben: metódus, kérési cél, protokoll verziója • Válasz: állapotkód, opcionális indok, protokoll verzió
Fejléc szakasz	• Tartalom előtt küldött mezők a fejlécmezők
Tartalom	• Byte folyamként kerül továbbításra fejléc után • Content-Type és Content-Encoding határozzák meg a kódolást/formátumot • Szemantika: tartalom célját metódus szemantika határozza meg • Válaszban a tartalom célját a kérés metódusa, az állapotkód és a tartalmat leíró mezők határozzák meg
Lezáró szakasz	• Tartalom után küldött mezők • Ellenörző összegek, digitális aláírások, kézbesítési metrikák, utófeldolgozási információk • Lezáró mezőket fejléctől elkülönítve ajánlott tárolni, feldolgozni

• **Mezők**: adatok név-érték párok formájában

- Kezdetinformációk továbbítása a céljuk
  - üzenetet leíró metaadatok
  - reprezentáció metaadatok
  - információk kliensről / szerverről
  - erőforrás metaadatok
- küldésük és fogadásuk fejléc és lezáró szakaszban történik
- mező**nevek** US-ASCII karakterkészlet egy részhalmazából állhat
- mező**érték** US-ASCII karakterekből, szóközökből, vízszintes tabból állhat

  vezető, záró whitespace karaktereket felhasználás előtt el kell távolítani

  minden mező korlátozhatja az értékeit
- mezőszakaszok: mezősorokból állnak, tartalma mezőnév + mezősor érték

  sorrendjük nem lényeges
- mezők összes HTTP verzióhoz kerülnek meghatározásra

User-Agent fejlécmezők:

- felhasználói ágensről(xd) tartalmaz információkat, kérés származásától felhasználható válasz testreszabásához
- ajánlott minden kérésben elküldeni a fejlécmezőt
- egy vagy több termékazonosítóból áll melyeket megjegyzések követhetnek
- azonosítók felsorolása fontosság szerint csökkenően
- termékazonosító: név + verziószám (opcionális)
- megjegyzéseket zárójelek határolják

Metódusok:

• **GET**: információ lekérés
- cél egy kiválasztott reprezentációját kéri
- kliens kérheti, hogy a kiválasztott reprezentáció bizonyos részeit kapja meg
- GET-re adott válasz cache-elt

• **HEAD**: GET-tel azonos, de válaszban üzenettörzset nem küldhet, válasz cache-elt ha cache controlt nem jelez mást

• **POST**: kéri, hogy a cél dolgozza fel a mellékelt reprezentációt saját szemantikája szerint
- lehetőségek: adatok küldése feldolgozónak
  üzenet eljuttatása hírcsoportba
  új erőforrás létrehozása
  adatok hozzáfűzése létező reprezentációhoz
- csak frissítési információ cache-el

• **PUT**: szerver létrehozza vagy helyettesíti a cél állapotát
- sikeres ha 200(OK) üzenetet kapunk következő GET kérésre
- POST küldés – PUT hellyetesítés

• **DELETE**: cél erőforrás és aktuális felhasználási kapcs törlése
- sikeres ha válasz 200, 202, 204

• **Állapotkódok**: háromjegyű decimális
- első számjegy határozza meg a fajtáját

1xx – információs: végső válasz előtti előzetes válasz
2xx – siker: szerver megkapta / elküldte a kérést
3xx – átirányítás: kérés kiszolgáláshoz további műveletek
4xx – kliens hiba
5xx – szerver hiba

• **Tartalomegyeztetés:**
Rendelkezésre álló reprezentációk eltérőek lehetnek formátumban / kódolásban
Cél a legmegfelelőbb kiválasztása.

- Proaktív egyeztetés: szerver-vezérelt tartalomegyeztetés
- *Accept, Accept-Charset, Accept-Encoding, Accept-Language, User-Agnet* mezők alapján választ
- Egyeztetés után a válaszban jelzi mi alapján választotta az adott reprezentációt

Hátrány:
- nem hatékony minden kérésben az ágens feltételeit leírni
- bonyolult válaszadás szerver részről
- korlátozott válasz újrafelhasználás
- lehetetlen pontosan meghatározni a legjobb reprezentációt

Ajánlott: ha szerver első válaszban el akarja küldeni a legmegfelelőbb reprezentációt, további kérést elkerüli

- Accept fejlécmező: értéke média tartományok vesszővel elválasztott listája
- minden tartományt több típusparaméter követhet, azt relatív súly (?) követheti
- */* : összes típus
- type /* : adott típus összes altípusa
- type / subtype: adott médiatípus text/html
• eltérő lehet címsorban és letöltés után

- Location mező: jelzi hova irányít egy kérés

- **HTTP 1.1 formátum**:
1. kezdősor <metódus> <kérés-cél> “HTTP/1.1”
2. O – több fejlécsor (?)
3. üres sor – fejléc vége
4. üzenettörzs (opcionális)

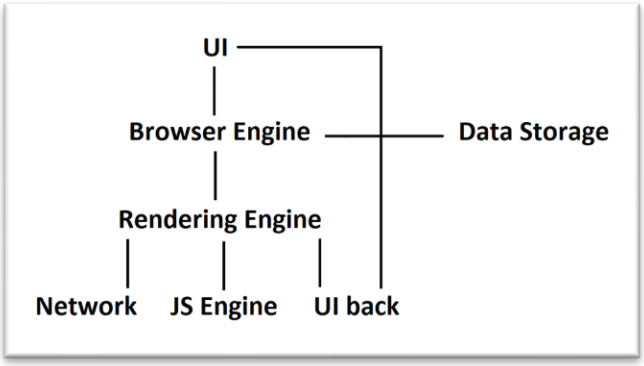
1.Kezdősor:
- kérés-cél a célt azonosítja
- gyakori formája útvonal [“?” lekérdezés]
- ha a cél üres a kliens ‘/’-t küld
- URI host és port – host mezőben kerül küldésre

- Válaszok 1. sora az állapotsor: “HTTP/1.1” <állapotkód> [indok-frázis]
- Mezősorok: mezőnév: opc. vezető whitespace + érték záró whitespace
- Mezősorokat CRLF zárja

4. Üzenetörés:
- kérés vagy válasz tartalmának hordozása
- azonos a tartalommal feltéve ha nem alkalmaz átviteli kódolást

6. Webböngészők

Felépítése és komponensei:



• **UI:** címsor, gombok, menük, minden látható rész kivéve ahol a lap megjelenítésre kerül

• **Browser Engine**: közvetítő szerep UI és Rendering Engine közt
- magas szintű interfész rendering engine manipulálásához

• **Rendering Engine**: célja webes tartalom megjelenítése

• **Hálózati komponens**: Hálózati kommunikáció megvalósítása
- platformfüggetlen interfész, melyen platformspecifikus API-kat használnak

• **JS Engine**: JS kód végrehajtása

• **UI backend**: UI elemek és weboldal megjelenítése
- platformfüggetlen interfész, melyen platformspecifikus API-kat használnak

• **Data Storage**: adatok ***perzisztens*** tárolása

Fő render motorok:

	Webkit	Blink	Edge HTML	Gecko
Fejlesztő (?)	Apple	Chromium Project	Microsoft	Mozilla Project
Licensz	LGPLv2 / Simplified BSD License	LGPLv2.1	Nem szabad	Mozilla Public License 2
Nyelv	C++	C++	C++	C, C++, JS, Rust
Használat	• Safari • Chrome (iOS) • Firefox (iOS) • GNOME web	• Chromium • Chrome • Edge • Firefox (iOS) • Vivaldi	• Edge	• Firefox

Asztali böngészők	Mobil böngészők
• Chromium • Firefox • Opera • Safari • MS Edge	• Google Chrome • Safari • Samsung Internet • UC Browser

Böngésző kidegészítők, kiterjesztések:

- kiegészítők: megjelenés vagy viselkedés testreszabása
- kiterjesztés: új funkciót ad hozzá vagy meglévőt módosít

Headless böngészők:

- GUI nélkül működnek
- programozottan viselkednek
- felhasználás: webes appok automata tesztelése, interakciók automatizálása, információk kinyerése, screenshotok készítése
- Open source headless böngészők: HtmlUnit, Puppeteer

About URI séma:

- böngészők hozzáférést adnak belső erőforrásaikhoz, beállítások, alkalmazásinfó

Chrome: *chrome://about*
Firefox: *about:about*
Opera: Chrome alap részhalmaza + pár extra. Sémanév: *opera:*
Safari: csak *about:blank*
MS Edge: néhány URI-t támogat pl: about:blank, *about:flags*
Chromium Edge: *edge://, edge://about*

7. Markdown

Pehelysúlyú jelölőnyelv sima szövegformázási szintaxissal

- Könnyen írható, olvasható
- Különféle kimeneti formátumba alakítható
- Egyszerű felhasználókat célozza meg
- Szintaxisát sima e-mail formázás ihlette

Felhasználása:

- Kollaborációs eszközök, platformok
- Tartalomkezelő szoftverek
- Közösségi platformok
- Műszaki / tudományos publikálás

Szabványosítás: Common Mark

- Markdown egyértelmű szintaxis specifikációja tesztkészlettel kiegészítve

Egyéb változatok: GFM, Pandoc's Markdown

Jelölőnelvek: szövegek, képek, egyéb elemek ellátása valamilyen jelölésrendszerrel

- szövegrészek ellátása metaadatokkal szövegtől elkülönülve

8. XML

Szintaxis strukturált dokumentumok ábrázolása, melyel lehetséges az automatikus feldolgozás

XML vs HTML:

• nincs előre definiált címkészlet • célja adatok leírása • adatcsere formátumként használják	↔	• előre definiált címkészlet • prezentációs nyelv • tekinthető XML speciális alkalmazásának xHTML

Előnyei:

- Egyszerű
- Nyílt
- Gyártó- és platformfüggetlen
- Univerzális adatcsere formátum
- Tág infrastruktúra
- De-facto szabvány az iparban

Hátrányai:

- bőbeszédű, nehezen használható szintaxis
- nagy tárigény
- bonyolult

Dokumentumközpontú XML:

- Folyó szöveg, jelölésekkel kiegészítve, lényeges az elemek sorrendje
- Változatos szerkezet, emberi fogyasztásra szánt - xHTML

Adatközpontú XML:

- Nagy számú adatelem alkotója, sorrend kevésbé lényeges, szerkezete kevésbé random
- Gépi feldolgozásra szánt – SVG

9. XML 1.0

Dokumentumok: olyan szöveges objektum, mely szabvány előírás szerint jól formált

- Fizikai szerkezet: egyedeknek nevezett egységekből állnak
- Logikai szrkezet: deklarációkból, elemekből, foldolgozási utasításokból állnak

Jólformáltság:

- gyökérelem tartalmazza a többi elemet (felső szintű)
- minden nyitó cimkéhez záró tartozik
- elemek egymásba ágyazottak, nem fedhetik egymást
- adott dokumentumban minden hivatkozott egyed jólformált

Elemek: minden elemet nyitó és záró cimke határol, vagy üres elem cimke alkot. nyitó-záró, üres cimkében adott nevet elemtípusnak nevezünk

- elemekhez meg lehet adni attribútum specifikációkat (név-érték párok)
- névadás: számmal nem kezdődhet vagy nem tartalmazhat

Specifikus karakterek: **&** és **<**, **>** jelölőhatárolóként használt

- helyettük karakterhivatkozásokat, egyed hivatkozásokat kell használni
- &amp, &gt, &/t

Jelölők:

- nyitócimke: <név (atribútumspecifikáció)*S?>
  pl: **<title XML:lang=”hu”>**
- zárócimke: </név S?>
  pl: **</title>**
- üres elem: <név (S attr.spec.)*S?>
  pl: **<img src=”…”>**
- karakterhivatkozás:
  &#nnnn – n decimális
  &#hhhh – h hexadecimális
- egedhivatkozás: hivatkozás névvel azonosítot egyed tartalmára
  pl: **&név → &amp**
  paraméteregyed-hivatkozás: **%inline**
- megjegyzés:
  <!--szöveg -->
- feldolgozási utasítások: <? név (S szöveg)?>
  pl: **<? xml.sylesheet type = “text/css” href=”style.css”>**
- CDATA szakasz határolók: olyan karakteres adat levédése, mely amúgy jelölő lenne
  pl: **<![CDATA[szöveg]]>**
- XML deklaráció: meghatározza az XML használt verziószámát
  pl: **<? xml version = “ “>**
  <? xml version = ‘1.0’ encoding=”UTF-8”>
- dokumentumtípus deklaráció: egy osztályhoz egy nyelvtant mehatározó jelölő deklarációkat tartalmaz. Ilyenekre mutat a név, a gyökérelem típusát írja elő

DTD:

Dokumentumtípus-definíció: megszorítások írhatóak a dokumentum logikai szerkezetére, a tárolási egységek használatát támogatja.

Jelölő deklarációkból áll.

- a dokumentumtípus-deklaráció mutathat külső DTD-re, tartalmazhat belső DTD alkészletet, vagy mindkettőt
- a 2 alkészlet alkotja egy dokumentum DTD-jét
- ha két alkészlet van a belső megelőzi a külsőt, ez lehetővé teszi a külsőben lévő egyed-attribútumlista-deklarációk felülírását

Érvényesség: akkor érvényes egy XML dokumentum, ha tartozik hozzá dokumentumtípus-deklaráció és a dokumentum eleget tesz a DTD általi megszorításoknak

Elemtípus-deklarációk: elem tartalmára ír előr megszorítást (egy elemtípus nem deklarálható többször)

Üres elemek: az így deklarált elemeknek nem lehet tartalma

- ***<!ELEMENT br EMPTY>***

Elemtartalom: elemtartalmú az elemtípus, ha csak elemgyermekeket tartalmazhatnak

- megjegyzések és feldolgozási utasítások is megengedettek
- a deklaráció egy tartalommodellt határoz meg, ami reguláris kifejezéshez hasonló minta:
  - használható: sorozat, alternatív lista
  - előfordulások számára:

?	0 vagy 1 előfordulás
+	1 <= előfordulás
*	0 <= előfordulás

Vegyes tartalom: karakteres adatokat tartalmazhat elemegyedekkel vegyítve

- korlátozható az elemgyereke típusa, de sorrendje, száma nem
- pl: ***<! ELEMENT message (#PCDATA | value)*>***

Érvényesség: Ugyanaz a név nem szerepelhet egynél többször a deklarációban.

- egyedhivatkozások a helyettesítő szövegükkel történő használat után az így deklarált elemet kar. adatok (DTA szakaszokat, megjegyzéseket, feldolgozási utasításokat kell tartalmazzon), valamint olyan gyerekeket, melyek típusa megegyezik a tartalommodellben adott névvel

Előre definiált egyedek:

- előre definiált egyedek speciális karakterekhez: *amp, /t, gt*
- ezeket minden XML feldolgozó fel kell ismerje
- a /t és amp egyedeket olyan belső egyedként kell deklarálni, melyek helyettesítő szövege egy megfelelő karakterre mutató karakterhivatkozás
  ehhez *'&'*-t le kell védeni: ***<!ENTITY /t, & s #60;">***
- gt, apos és quot: olyan belső egyedként kell deklarálni, melyek helyettesítő szövege egyetlen karakter vagy karakterhivatkozás az adott karakterre:
  <!ENTITY gt ">">
  <!ENTITY gt "&#38,#62;">

10. JS / ECMASCRIPT

- ECMASCRIPT: JS programozási nyelv szabványosítása
- Javascript: ECMASCRIPT megvalósítása különböző gyártók által
- JS motor: olyan szoftver, ami végrehajtja a JS kódot
- Főbb JS motorok:
  - **Spidermonkey** (Mozilla)
    - C/C++
    - Mozilla public license
  - **V8**
    - C++
    - New BSD License
  - **Chakra** (MS Edge)
    - MIT Licencse
  - **Nashhorn** (Oracle)
    - Java
    - GPLv2
  - **GraalVM** (CE) (linux, MACos)
    - Java
    - GPLv2
- **Node.js:**
  - V8 motorra épül
  - skálázható, hálózati alkalmazások létrehozására tervezték
  - böngészőn kívül futtatható, JS alkalmazások írhatóak vele kliens és szerver oldalon egyaránt
  - csomagok ökoszisztémája

11. JSON

JSON: könnyűsúlyú, szöveges, nyelvfüggetlen adatcsere formátum
- struktúrált adatok ábrázolása
- ember számára könnyen írható, olvasható formátum
- szoftverek által könnyen generálható / feldolgozható

JSON ↔ ECMASCRIPT
- JSON az ECMASCRIPT szintaxisán alapul, de nem teljesen kompatibilis vele

JSON ↔ XML
- XML alternatívája, ugyanazok az előnyök XML hátrányai nélkül

közös:
- egyszerű (JSON jobb)
- ember számára könnyen értelmezhető
- szoftver számára könnyen generálható, feldolgozható
- interoperabilitás
- nyiltság
- önleíró DB
- univerzális adatcsere formátumok

eltér:
- JSON adatorientált ↔ XML dokoumentumorientált
- JSON kevésbé bőbeszédű → adatszerkezet ábrázoláshoz alkalmas
- XML dokumentumközpontú alkalmazásokhoz jobb, kiterjeszthető, kiforrottabb infrastruktúrája van

Primitív típusok:
- számok: nincs korlátozás tartományukra és pontosságukra
- sztringek: Unicode karakterek sorozatai, melyeket idézőjelek határolnak
  - bármely karaktert tartalmazhat , *“;* \ levédve
  - vezérlő karakterek szintén csak levédve
  - speciális karakterekhez – szokásos escape szekvenciák
  - \unnnn módon is megadhatóak

Struktúrált típusok:
- tömbök: tetszőleges számú érték rendezett sorozata
  lehet üres, elemek típusa eltérhet
- objektumok: tetszőleges számú név-érték párok (member)

// RFC 8259:

- objektum interoperábilis, ha a tagok nevei egyediek
- nem egyedi nevekkel az alkalmazások eltérően viselkedhetnek
- nem minden elemző esetén állapítható meg a tagok sorrendje

XML – JSON konverzió:

```
<file>
      <uri> http://www.w3.org/icons/w3c_home.png </uri>
      <size> 1936 </size>
      <contentType> image/png </contentType>
      <lastModified> 2006-07-24T14:58:332 </lastModified>
</file>
```

JSON:

```
{
“file” {
      “uri” : “https://www.example.com/asd.png”
      “size”: 1936
      “contentType”: “image/png”
      “lastModified”: “2006-07-24T14:58:332”
      }
}
```

JSON schema:

JSON objektum: application / JSON média típus által leírt információ
erőforrás, egy JSON érték

példány (instance): olyan objektum melyre egy séma vonatkozik

JSON séma:
- olyan JSON objektum mely példányokat ír le
- objektum vagy logikai érték
- a sémák egymásba ágyazhatók (root schema, subschema)
- média típus: application / schema + JSON

tulajdonság (property): obj. példány egy tagja

kulcsszó (keyword): egy sémaobjektum egy példányra vonatkozó tulajdonsága
megszorításokat fejeznek ki, vagy információkkal lát el példányt

szótár (vocab): adott célt szolgáló kulcsszavak halmaza
kulcsszavaink jól meghatározott szintaxisa, jelentése van

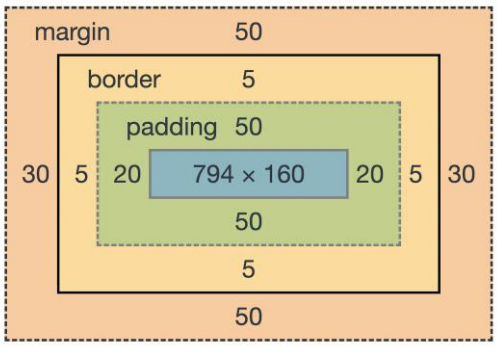
meta-séma: sémát leíró séma

12. CSS

- Strukturált dokumentumok megjelenítésének leírása (többféle eszközt támogat)
- A tartalom és formázása elválik egymástól
- W3C fejlesztése
- Szintjei: újabb szintek a korábbira épül, mely csak fejletebb verziók kezelnek

CSS level 1: túlhaladott W3C ajánlás
CSS level 2: - egyetlen dokumentum definiálja – CSS 2.1
- CSS 2.2 javitása jelenleg fejlesztés alatt áll
CSS level 3: fejlesztés alatt áll
- moduláris felépítésű: minden modul egy CSS részt definiál
CSS 2.1 specifikációt bővítik / cserélik le
- moduloknak is vannak szintjei
1-ről indulnak azok, melyeknek nincs CSS lvl 2 megfelelője
3-tól csak melyek CSS lvl 2-t frissítik
- modulok stabilitása eltérő
CSS level 4 és azon túl: nem létezik CSS level 4
- önálló modulok elérhetőek 4-es szintre, vagy tovább, de CSS 4 nincs

Dobozmodell:
- A CSS egy fa struktúrájú dokumentumot kap, melyet egy vásznon jelenít meg, olyan dobozfa struktúrával mely a dokumentum formázási szerkezetét ábrázolja
- A fa minden doboza a dokumentum egy megfelelő elemét ábrázolja
- A CSS minden elemhez nulla vagy töb dobozt generál az elem display tulajdonsága alapján



Szintaktikai elemek:
- karakterek: unicode karakterkészlet
- vezérlősorozatok: unicode megadáshoz \hhhhhh, legfeljebb 6 hexa karakter
- whitespace megadása végére, ha <6 hosszú
- speciális karakter elnyomása: \ pl: given\name-re illeszkedik given.name
- megjegyzések: /* */ közt
- deklarációs blokk: { } közt, deklarációk listája ;-vel elválasztva
a deklarációk tulajdonságnév: érték formájúak

- at-szabályok: stylesheet feldolgozásának speciális szabályai
    @karakterekel kezdődnek, melyet azonosító követ, ‘;’ zár vagy { }
    pl: @charset, @import
- szabályhalmazok: kiválasztó(k)ból és deklarációs blokkból állnak

Tulajdonságok:

- CSS által definiált paraméterek , melyekkel a dokumentum megjelenése vezérelhető
- tulajdonságnak neve-értéke van
- shorthand property (összevont tulajdonság): több CSS tulajdonság értékeinek egyidejű beállítása
    pl: margin → -top, -bottom, -left, -right

Kiválasztók:

- típus kiválasztó: CSS minősített név, gyakorlatban azonosító, megfelelő nevű elem illeszkedik rá
- általános kiválasztó: * formájú kiválasztó, minden elem illeszkedik rá, egyszerű kiválasztókból elhagyható, ami több elemet tartalmaz
- attribútum kiválasztók:
    - [att]: az att attribútummal rendelkező elemek illeszkednek rá
    - [att=érték]: elemek att attribútumának érétéke ‘érték’ illeszkedik
- osztály kiválasztó: .érték (pont!!!)
- id kiválasztó: #azonosító - szükséges id típus attribútum dokumentumban
- pszeudo osztályok: :azon vagy :azon(érték) alakú
    - olyan kiválasztás mely dokumentumon kívüli információn alapul, egyszerű kiválasztókkal nem kifejezhető
    - bizonyos pszeudo-osztályok kizárják egymást
- dinamikus pszeudo osztály:olyan p. o. amit egy elem megszerezhet vagy elveszíthet interakció hatására
- :lang (C) pszeudo osztály:
    - C nyelvű szöveget tartalmazó elemek illeszkednek rá
    (C egy CSS azonosító nyelvkód) pl. lang(hu)
    pl. magyar nyelvű idézet esetén magyar idézőjelek:

```
q:lang(hu){
      quotes: ”,,” “ ‘ ‘ “ “>>” “<<”;
      }
```

Kombinátorok:

- leszármaztatott kombináta: egyszerű kiválasztók két sorozatát elválasztó whitespace
- ha P és Q egyszerű kiálasztók két sorozata, akkor a P Q a Q-ra illeszkedő olyan elemek illeszkednek, melyek a P-re illeszkedő elemek leszármazottai

  pl: thead th {bgcolor: black}

• gyermek kombinátor: egyszerű kiválasztók két sorozatát elválasztó **">"**

- ha P és Q egyszerű kiválasztók két sorozata, akkor a **P>Q** kiválasztóra Q-ra illeszkedő olyan elemek illeszkednek, melyek P-re illeszkedő elemek gyermekei

• szomszéd testvér kombinátor: egyszerű kiválasztók két sorozatát elválasztó **"+"**

- ha P és Q egyszerű kiválasztók két sorozata, akkor P+Q kiválasztóra a Q-ra illeszkedő olyan elemek illeszkednek, melyek a P-re illeszkedő elemet követnek közvetlenül a dokumentumban
  - illeszkedő elemnek ugyanaz kell legyen a szülője
  - közöttük megengedettek olyan konstrukiók amik nem elemek

  pl: `h1 + p {text_indent : 0}`

Specifikusság:

- a specifikusság egy háromelemű (a,b,c) vektor, ahol a,b,c nem negatív egészek
- a vektorok rendezése lexikografikus
- meghatározása: **a** = kiválasztóban előforfuló ID-kiválasztók száma

  b = előforduló attr. kiválasztók & pszeudo-osztályok száma

  c = előforduló típus kiválasztók és pszeudo-elemek száma
- b : negáció psz.o.-t ignorálni kell, de az argumentumát nem

Stílus eredet:

- felhasználói ágenstől: biztosít alapértelmezett stíluslapot

  pl: Ff: resource: //gre-resources/
- felhasználótól származó saját stíluslapot adhat meg
- szerzőtől származó: link fejlécelemmel adható meg külső stíluslap

  <style></> is használható

  XML esetén: xml-stylesheet

Kaszkád:

- több különböző (eredetű) deklaráció szolgáltatja egy tulajdonság értékét egy elemhez
- a kaszkád folyamata alatt kerül meghatároásra, mely deklaráció határozza meg az adott elem tulajdonságának értékét

Menete:

- deklarációk sorbarendezése eredetük szerint csökkenően
- azonos eredetű deklarációkat specifikusság szerint csökkenően
    - azonos specifikuság esetén előfordulási sorrend dönt (későbbi az erősebb)
- a tulajdonság értékét a sorrend szerint első deklaráció adja

Szabályok sorrendje:

- azonos specifikusságú szabályoknál utolsó a legerősebb

Öröklés:

- tulajdonságérték továbbadása (szülőtől gyermeknek)
- bizonyos tulajdonságok értéke öröklődés alapján kerülnek meghatározásra, ha a kaszkád nem szól bele
- a specifikáció minden tulajdonsághoz meghatározza, öröklött-e
- kaszkádolt értéknél az inherit kényszeríti az öröklést
- minden link a szülőelemtől örökli color értékét

Kezdőérték:

- CSS specifikációja határozza meg minden tulajdonság kezdőértékét
- kezdőérték előírható úgy, hogy felhasználói ágenstől függjön
- ha kaszkád nem eredményez értéket és örökölt sincs, az alapértelmezett kerül beállításra
- tulajdonság kaszkádolt értékénél az *initial* kulcsszó esetén a kezdőérték lesz a meghatározott érték

13. CSS előfeldolgozók:

CSS-t generálnak a CSS írására szolgáló saját szintaxisukból, új lehetőségekkel egészítik ki a CSS-t

- közös jellemzők:

- egysoros megjegyzések: // - nem generál CSS-t fordításkor
- @import: böngészőktől eltérően kezelik a @import szabályokat
    - sima CSS importokhoz a böngészőnek rendeléskor HTTP kéréseket kel végrehajtson
    - az előfeldolgozók az importokat fordításkor végzik

- SASS: lehetőségei: változók, egymásbaágyazás, aritmetika, beépített függvények, asszociatív tömbök, kiterjesztés/öröklés import

Szintaxis: SCSS - CSS szintaxis kiterjesztése

```
$size: 2em;
table.chessboard {
      td {
            height: $size;
            width: $size;
      }
}
```

14. Web jelölőnyelvei:

HTML: a web elsődleges leíró nyelve, szemantikai szintű leíró nyelv és a kapcsolódó szemantikai szintű alkalmazásprogramozási interfészek a weben
elérhető oldalak készítéséhez melyek statikus dokumentumoktól dinamikus alkalmazásokig terjednek

HTML elemei: elemeknek, attribútumoknak és attr. értékeknek meghatározott szemantikája van, ezeket tilos NEM rendeltetésszerűen használni

minden elemnek van egy tartalommodellje, a tartalom meg kell feleljen a tartalommodell leírásának

DOM: a DOM fa egy dokumentum memóriabeli ábrázolása, egy API dokumentumok eléréséhez és manipulálásához
- minden csomópont egy API-val rendelkező objektum ábrázol, tehát manipulálható
- DOM interfészek WEB IDL-ben kerülnek leírásra
    - a WEB IDL egy interfészleíró nyelv, mely böngészőkben implementálható interfészek leírására szolgál
    - a HTML specifikáció a HTML elemek ábrázolásához a DOM interfészeket kiterjesztő további interfészeket határoz meg
    - a HTML implementációk megfelelési kritériumai is DOM műveletekkel vannak meghatározva
    - egy DOM fa szkriptekből manipulálható az oldalon

- **A HTML szintaxis** hasonlít SGML-hez, XML-hez, de saját feldolgozási szabályai vannak
    - kötelező a dokumentumtípus deklaráció
    - speciális karakterek: elem szövege nem tartalmazhat *<,>* vagy *&* karaktert
      attribútum nem tartalmazhat *&* karaktert
    - félreérthető & karakter: *&* melyet ASCII alfanumeirkus karakterek és *;* követ
    - elem és attribútumnevek kis-nagybetű érzéketlenek

- nem idézett attribútumérték szintaxis: ha attribútumérték nem tartalmaz whitespace karaktert és nem üres, elhagyható az " "
- logikai attribútum: logikai attribútum jelenléte az elem igaz értéket ábrázol
  hiánya hamis értéket
- ha az attribútum megjelenik, akkor értékének üresnek kell lenni, vagy olyan értéknek mely az attribútum nevével megegyezik
- void elemek: záró címké megadása tilos -
- idegen elem: nyitó cimkéje lehet önlezáró, vagy kellhet külön záró
- opcionális címke: bizonyos elemek zárócimkéje elhanyagolható
  pl: /i, ha újabb /i követi
  - ha nem lényegesek az elemek közötti whitespace karakterek
  - HTML nyitó elhanyagolható, ha elsőként nem megjegyzést tartalmaz, vagy nem megjegyzés követi
- nem támogatottak a névtér deklarációk
- CDATA szakaszok csak idegen tartalomban használhatók

- **Dokumentumtípus-deklaráció**: célja hogy a megjelenítés szabványos módon történjen
  - XML-ben nem kötelező, tetszőleges dokumentum deklaráció használható

15. Reszponzív webdizájn

Lehetővé teszi az eszközökhöz igazodó tartalom szolgáltatását

Elrendezések:

Statikus	Abszolút mértékegységben (általában pixel) rögzített szélességet használó elrendezés
Fluid	Relatív mértékegységben kifejezett szélességet használ
Adaptív	Statikus elrendezések egy sorozata, médialekérdezés után határoz meg különböző statikus elrendezést
Reszponzív	Fluid elrendezések sorozata, médialekérdezések után határoz meg különboző szélességet

- Pixel: CSS és fizikai pixelek különbözőek
- Referenciapixel: az a látószög mely alatt egy pixel látszik egy 96 dpi pixelsűrűségű eszközön kartávolságból nézve
  karhossz távolságból 1px ~ 0.26mm (1/96 inch)

- CSS abszolút hosszúság mértékegységek
  - egymáshoz képest rögzítettek, valamilyen fizikai mértékegységhez kötött

- kötése: fizikai mértékegységek a megfelelő fizikai mérésekhez való kötése vagy px referenciapixelhez kötése
- kis pixelsűrűségnél ajánlott px-hez kötni
- nagy pixelsűrűségnél ajánlott szabványos fizikai mértékegységhez kötni

- Nézetablak mértékegységek:

vw: nézetablak 1%-a
vh: nézetablak magasságának 1%-a
vmin: vw és vh közül a kissebbel egyenlő
vmax: vw és vh közül a magasabbal egyenlő

- Médialekérdezések:

- a média lekérdezés egy módszer a felhasználói ágens vagy eszköz bizonyos jellemzőinek vizsgálatára melyen megjelenik a dokumentum
- csak külső információtól függenek, kivéve ha máshogy írják elő

CSS-ben:

@import: ha a média lekérdezés nem illeszkedik, az import nem ervényesül
@media: feltételes csapatszabályok egy feltételt társítanak más CSS szabályok egy csoportjához
- minden feltételes csoportszabálynak van egy feltétele ami lehet igaz vagy hamis
Ha a feltétel igaz a CSS feldolgozónak használnia kell a csoportban meghatározott szabályokat

XML-ben:

- xml-stylesheet feldolgozási utasítás média pszeudo-attribútuma
a médiát adja meg melyre a hivatkozott stylesheet vonatkozik

```
<? xml stylesheet type=“text/css”
      media=”screen” href=”style.css”?>
```

HTML-ben: link, source, style-ra adható meg média attribútum

```
<link rel=”stylesheet” type=”text/css”
      media=”screen” href=”style.css”>
```

- Szintaxis: opcionális módosító, opcionális médiatípus, opcionális média feltétel
több lekérdezés vesszővel elválasztva kombinálható

- Kiértékelés: média lekérdezés igaz, ha:

- média feltétele igaz
- típus meg van adva, illeszkedik az eszköz típusára, ahol az ágens fut
- valamely komponens lekérdezése igaz, hamis ha az összes hamis

- agenteknek változáskor újra ki kell értékelni a lekérdezéseket

- Média típusok:

AU	Összes eszköz
Print	Nyomtatott megjelenítést célzó eszközök
Screen	print-re, speechr-re nem illeszkedő eszközök
Speech	Képernyőolvasók, hasonló eszközök

- Média jellemzők: média típusoknál finomabb teszt, ami bizonyos jellemzőt vizsgál

width	szélességet vizsgál
height	magasságot vizsgál
orientation	álló/fekvő tájolást vizsgál

- Kombinálás:

not	jellemzők negálása
and	- mindkettő összefűzés
or	- vagy

- Min-Max előtagok:

min: >=

max: <=

- Diszkrét típusú média jellemzőkhöz nem adható meg *min* vagy *max*

- Nézetablak:

- látható: oldalnak a kijelzőn látható része (visual viewpoint)
  elrendezési nézetablak, melyben a böngésző kirajzol egy objectet (layout viewpoint)
- viewpoint meta: layout viewpoint méretének és kezdeti nagyítási arányának megadása

16. HTTP haladó lehetőségei

GET:

- cél erőforrás egy aktuális kiválasztott reprezentációját kéri
- információ lekérési cél
- range fejlécmező küldésével módosítható, csak bizonyos részét kéri a kiválasztott reprezentációnak
- válasz gyórsítótárazható

POST:

- kéri, hogy a cél dolgozza fel a mellékelt reprezentációt saját szemantikája szerint
- lehetőségek:
    - adatok küldése adatfeldolgozó folyamatoknak
    - üzenet postázása blogra, hírcsoportba
    - új erőforrás létrehozása
    - adathozzáfűzés erőforrás létező reprezentációjához
- csak akkor gyorsítótárazható, ha explicit frissítést tartalmaz

Sütik:

- név-érték pár és kapcsolódó metaadatok, melyeket egy eredet szerver egy válasz set-cookie fejlécmezőben küld az agentnek
- attribútumokkal egy hatáskört határoz meg a sütikhez
- továbbiakban az ágens a név-érték párokat a cookie fejlécmezőben küldi vissza

Felhasználása:

- munkamenet-kezelés
- testreszabás
- felhasználó követés

Set-Cookie:

- felhasználó eltárolja az attribútumokat , amiket ebből kap
- továbbiakban, amikor az ágens HTTP kérést hajt végre a cookie mezőbe helyezi a nem lejárt sütiket (csak név-érték párokat)
- ha ágens olyan sütit kap melynek neve *Domain* és *Path* attribútuma megegyezik egy korábbival, lecseréli azt

Attribútumok:

expires	Süti lejáratának ideje
max-age	Lejáratának ideje (másodperc)
Domain	• Meghatározza mely szervereken lesz küldve a süti ha a (?) kihagyja az attribútumot csak eredet szerveren továbbít • Ágens mindent elutasít, ami hatásköre nem tartalmazza az eredet szervert
Path	• Süti hatáskörét útvonalakra korlátozza • Ha a szerver ezt kihagyja, az ágens a kért URI útvonal komponensének könyvtárát használja
Source	Sütik hatáskörének korlátozása biztonságos csatornákra
HTTP only	• HTTP kérésekre korlátozza a sütiket • kliens oldali API-k számára nem lesz elérhető

Harmadik féltől származó sütik:
- HTML oldal megjelenítésekor az ágens gyakran kér le erőforrásokat más szerverektől
- ezek a szerverek sütiket használnak a felhasználó követésére

Felhasználó követése:
- felhasználó tevékenységével kapcsolatos adagyüjtés több kontextuson keresztül, majd begyűjtött adatok megosztása kontextuson kívül
- alapja: IP, sütik, Etag, eszköz ujjlenyomat

Referer fejlécmező:
- ágens megadhatja azt az erőforrás azonosító URI-t amiből a cél származik

Védekezés:
- Refer küldésének tiltása:
    - Ff: network.http.senaReferHeader
    - Chrome: Refer Control
    - Opera: NoRef
- harmadik féltől származó sütik tiltása:
    - privát böngészés, automatikus törlés munkamenet végén
- bővítmények:
    - Adblock
    - Disconnect
    - uBlock

17. HTTP/2

- Késleltetési idő csökkentése: képek közvetlen beágyazása CSS-be
- Spriting: több kép kombinálása egy állományba
- Sharding: tartalom elosztása több szerveren
- Összefűzés: több CSS és JS állomány összefűzése
- Kicsinyítés: felesleges karakterek eltávolítása

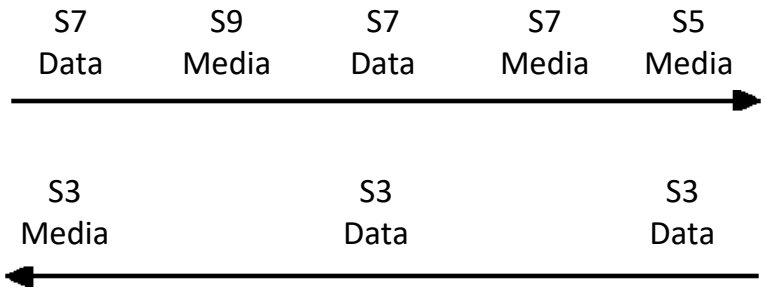
HTTP/2:

- optimalizált HTTP szemantika
- cél a hálózati erőforrások hatékony kihasználása, késleltetés csökkentése
- klienseknek csak 1 kapcsolatot kelljen fenntartani
- üzenetek formálása / átvitele eltérő

Újdonságok:

- **Multiplexelés**: független adatfolyamok használata
- **Forgalomvezérlés / rangsorolás**: multiplexelés hatékony megvalósítására
  - annyi adat kerüljön továbbításra, amennyit a fogadó kezelni tud
  - legfontosabb adatokat küldje elsőnek
- **Szerver Push**: spekulatív módon küld adatokat a kliensnek, melyekre szüksége lehet
- **Bináris protokoll**: binárisan formázza az üzeneteket – hatékonyabb küldés
- **HPACK**: fejlécmezők tömörítése

Üzenetek multiplexelése:



Keretek:

Hossz - 24 bit	
Típus - 8 bit	Jelzők - 8 bit
R	adatfolyam azon 31 bit
Adatrész	

- Hossz: adatrész hossza 24 bit
- Típus: felépítést és szemantikát határoz meg 8 bit
- Jelzők: logikai jelzők (típusfüggő) 8 bit

• R: fentartott célú 0 bit
• Adatfolyan id: 31 bit, adatfolyamot azonosítja

Keretfajták: kód / típus / funkció

0x0 – DATA: payload átvitele
0x1 – HEADERS: adatfolyam megnyitása, fejléc blokk töredék biztosítása
0x5 – PUSH_PROMISE: szerver_push megvalósítása
0xS – CONTINUATION: fejléc blokk töredékek sorozatának folytatása

Adatfolyamok:

- Egy HTTP/2 kapcsolat több adatfolyamot tartalmazhat egy időben
- Adatfolyamokat egyoldalúan hozhatja létre szerver vagy kliens, mindkét végpont lezáratja
- Keretek sorrendje lényeges, fogadó a kapott sorrendben dolgozza fel őket

Azonosításuk:

- kliens által nyitott adatfolyamat páratlan szám azonosítja
- szerver által nyitott adatfolyamat páros szám azonosítja
- 0x00 azonosítót vezérlő üzenetekhez használják
- új kapcsolatok azonosítójának nagyobbnak kell lenni minden korábbinál
- azonosítók nem újrafelhasználhatóak, hosszú kapcsolatnál kimerülhet az azonosító tartomány → új kapcsolat

HTTP/2 kérés-válasz:

- a kliens minden kérést új adatfolyamon küld el
- szerver ugyanezen küldi a választ
- üzenetváltáskor elhasználódik az adatfolyam
- választ váró keret zárja az adatfolyamot

Fejlécmezők:

- HTTP/2 kódolás miatt fejléc mezőket kisbetüssé kell alakítani pszeudo-fejlécmezőket használunk ehhez

Pszeudo-fejlécmezők:

- Kérés:

- method: HTTP metódust tartalmaz
- scheme: cél URI séma részeit tartalmazza
- authority: cél URI autoritás része, Host fejlécmező megfelelője
- path: cél URI útvonal és lekérdezés része

*// szerver szintű OPTIONS kéréseknél a * az értéke*

- Válasz:

- status: HTTP állapotkódot hordozza (minden válasz része)

- pszeudo fejlécmezőknek meg kell előzni a rendes fejlécmezőket

18. HTTP/3

HTTP szemantika QUIC-re leképezve
QUIC: UDP-re épülő biztonságos, általános célú átviteli protokoll

HTTP2-HTTP3 összehasonlítás:

Hasonlóság	Eltérés
• Ugyanazok a lehetőségek kliensek számára • Adatfolyamok • Szerver push • Feljéc tömörítés	• HTTP/2 impementálható TLS nélkül • HTTP/2 néhány keretfajtája nem szükséges, QUIC nyújt alternatívát (*CONTINUATION* és *PRIORITY* sincs HTTP/3-ban) • HTTP/3 a HPACK módosított változatát (QPACK) használja