## World Wide Web

* + Alapvet˝o fogalmak: er˝oforr´as, reprezent´aci´o, tartalomegyeztet´es, hivatkoz´as-felold´as, felhaszn´al´oi a´gens
    - **Erőforrás:** Bármi, ami azonosítható egy URI-val
    - **URI (**Uniform Resource Identifier)**:** egységes erőforrás azonosító, a weben használt globális azonosító
    - **Reprezentáció:** erőforrás állapotról információkat kódoló adatok
    - **Tartalomegyeztetés:** egy erőforráshoz több reprezentáció kínálása, és ezek közül a legmegfelelőbb kiválasztása
    - **Hivatkozás-feloldás (dereferencing):** egy URI használata egy hivatkozott erőforrás eléréséhez
    - **Felhasználói ágens:** a web ágensek egy fajtája, egy személy nevében cselekvő szoftver, pl. webböngésző.
  + A szabváányok három fajt´aja eredetu¨k szerint: de facto szabv´anyok, de jure szabv´anyok, ¨onk´entes k¨ozmegegyez´eses szabv´anyok
    - **De facto szabványok**: a gyakori használatból vagy a piaci elfogadottságból származnak.
      * Példa: QWERTY billentyűzetkiosztás, TeX, PDF (2008 előtt).
    - **De jure szabványok**: helyi, állami és/vagy nemzetközi szintű szabályozók által előírt szabványok
      * Példa: Nemzetközi Mértékegységrendszer (SI), PDF 2008 után
    - **Önkéntes közmegegyezéses szabványok**: különböző magánintézmények (például szakmai egyesületek és szervezetek, akkreditált szabványügyi szervezetek és ipari konzorciumok) által meghatározott szabványok.
      * Példa: az Internet protokollkészletet (közismert nevén TCP/IP), HTML, CSS.
  + Web szabványokért felelős szervezetek: IANA, ISO, IETF, az RFC sorozat; W3C; WHATWG
    - **IANA: Internet Assigned Numbers Authority:** Az internet működésesnek alapjául szolgáló kódok és számok kiosztását koordinálja
      * DNS gyökérzóna felügyelete
      * .int, .arpa felső szintű tartományok üzemeltetése
      * IP címek kiosztásának globális koordinálása
      * Különféle internet protokollokhoz használt kódok és számok nyilvántartása
      * Az IANA egy funkció, melyet jelenleg egy nonprofit cég működtet, az Internet Corporation for Assigned Names and Numbers (ICANN)
    - **IETF: Internet Engineering Task Force:** internet szabványokat fejlesztő nemzetközi szabványügyi szervezet.
      * TCP/IP fejlesztése
      * Nincs formális tagság, bárki számára nyílt részvételi lehetőség
      * A szakmai munka munkacsoportokban történik
      * 1986. jan. 16. alapítás
      * Az internet szabványokhoz kötődő specifikációkat az RFC dokumentumsorozatban publikálja
    - **RFC: Request for Comments:** Az internetről szóló műszaki és szervezeti dokumentumokat tartalmaz.
      * 1969-ben indult az ARPANET részeként
      * Az RFC Editor szerkeszti, adja ki és kategorizálja
      * RFC sorozatot 4 folyamatra osztják:

IETF folyam

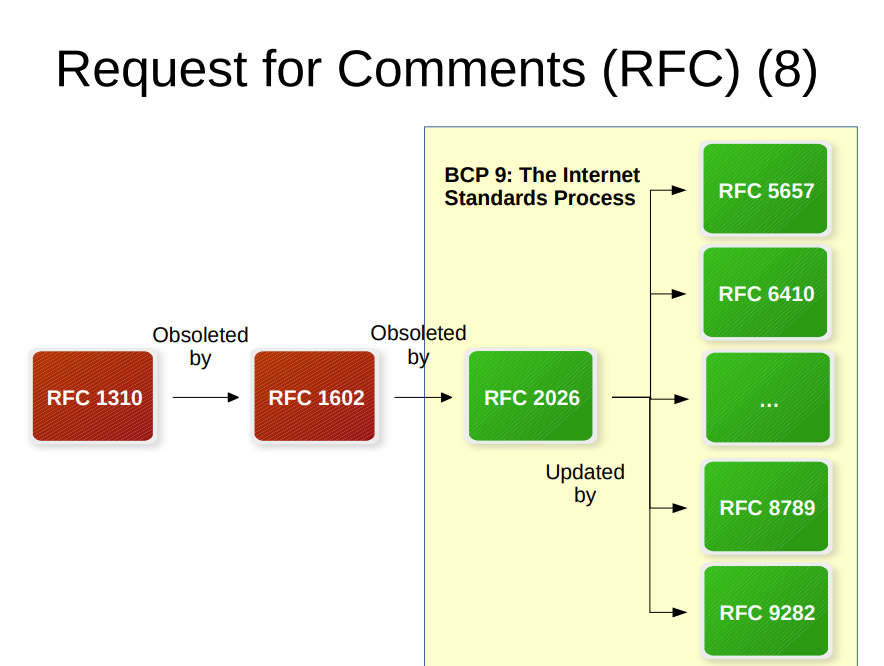
IAB folyam

IRTF folyam

Független beadványok folyama

* + - * Minden RFC-t egy szám azonosít, pl: RFC 9110
      * Minden RFC elérhető ASCII szövegként
      * Sosem módosulnak

A hibákat hibajegyzékek javítják

* + - * Két fontos alsorozat:
        + BCP: Best Current Practice
        + STD: Internet Standard
    - **W3C:** World Wide Web Consorrium
      * Nemzetközi közösség, ahol tagszervezetek, főállású alkalmazottak és a nyilvánosság munkálkodnak együtt a webszabványok fejlesztésén
      * A W3C ajánlásoknak nevezett, webtechnológiákat meghatározó és webszabványoknak számító dokumentumokat publikál.
      * **Tervezési alapelvek:**

**Web mindenkinek:** a web elérhető kell, hogy legyen mindenki számára, a hardverektől, szoftverektől, hálózati infrastruktúrától, anyanyelvtől, kultúrától, földrajzi elhelyezkedéstől, vagy a fizikai és szellemi képességektől függetlenül.

**Web mindenhol:** a web elérhető kell, hogy legyen a legkülönfélébb eszközökről.

* + - * **Története:**

A MIT-n alapították 1994 októberében.

Igazgatója Tim Berners-Lee, a web feltalálója és megalkotója.

1996 óta 300-nál több ajánlást jelentetett meg.

* + - * **Működése:**

A fejlesztést munkacsoportok végzik.

A munkacsoportok által előállított eredmények többek között szakmai jelentések, tesztkészletek és nyílt forrású szoftverek.

A munkacsoportok a szóban forgó terület szakértőiből állnak, akik az alábbiak lehetnek:

W3C alkalmazott,

egy tagszervezetet képviselő egyén (általában egy tagszervezet alkalmazottja),

egy meghívott szakértőként közreműködő egyén.

* + - **Web Hypertext Application Technology Working Group (WHATWG)**
      * + A Web fejlődése iránt elkötelezett közösség, mely böngészőkben implementálható szabványokat fejleszt.
        + **Szabványok**:

DOM

Fullscreen API

HTML

URL

XMLHttpRequest

* + - * **Történet:**

2004-ben alapították az Apple, a Mozilla Foundation és az Opera Software programozói, akik elégedetlenek voltak a W3C a HTML fejlesztésére irányuló tevékenységével.

* + - * **Működés**:

Működését az irányítócsoport (Steering Group) koordinálja, melynek tagjai az Apple, Google, Microsoft és a Mozilla.

* + - * **Részvétel:**

Nyitott a részvétel lehetősége a nyilvánosság számára.

* + - * **Fejlesztési modell:**

A WHATWG „élő szabványoknak” nevezett specifikációkat fejleszt, melyek folyamatosan frissülnek.

Az élő szabványok felhasználását a CC BY 4.0 licenc szabályozza.

1. **Unicode**
   * **Mi az Unicode?**
     + Univerzális karakterkódolási szabvány írott karakterekhez és szöveghez.
     + Lefedi a világ összes modern és ősi nyelvének összes karakterét.
     + Tartalmaz továbbá műszaki szimbólumokat, írásjeleket és sok más olyan karaktert, melyet írott szövegben használnak.
     + Széles körben használt és támogatott.
   * **Alapvető fogalmak:**
     + **Kódtér (codespace):** a karaktereket kódoló egész számok tartománya.
     + **Kódpont (code point):** a kódtér egy eleme, egy karaktert kódoló egész szám. pl: U+265F
     + **Basic Multilingual Plane (BMP):** 
       - Az első 65 536 kódpontot (U+0000–U+FFFF) tartalmazó sík (0. sík).
       - A gyakran használt karaktereket tartalmazza a világ összes modern írásrendszeréhez, valamint számos történelmi és ritka karaktert is tartalmaz.
       - A BMP-be tartozik a Unicode karakterek többsége szinte minden szöveges adat esetén.
   * **Karakterkódolások:**
     + **UTF-8**
       - 1-től 4 byte-on tárolva (változó szélességű kódolás)
         * Az U+0000 – U+007F tartományba eső 1 byte-on (128 ASCII karakter)
         * Az U+0080 – U+07FF tartományba eső 2 byte-on
         * A BMP összes többi kódpontját 3 byte-on
         * A BMP-n kívüli kódpontokat 4 byte ábrázolja
       - egy kódpontot ábrázoló byte-sorozat első byte-ja meghatározza a sorozat hosszát (egyszerű feldolgozhatóság)
       - A használt byte-ok szempontjából a legtömörebb kódolás
       - Kevésbé hatékony kelet-ázsiai rendszerek esetén
     + **UTF-16**
       - Minden kódpont 2 vagy 4 byte-on ábrázolt (változó szélességű kódolás)

BMP karaktereinek ábrázolása 2 byte

Minden más karakter 4 byte

* + - * Kompromisszum a hatékony hozzáférés és tárolás között
    - **UTF-32**
      * Minden kódpont ábrázolása 4 byte-on (rögzített szélességű kódolás)
      * Legegyszerűbb karakterkódolás
      * Feldolgozás szempontjából leghatékonyabb,
      * Tárolás szempontjából legkevésbé hatékonyabb
    - **ISO/IEC 8859**
      * 8 bites karakterkódolási szabványok: ISO/IEC 8859-1 … 8859-16
      * Számunkra fontosak:

ISO/IEC 8859-1 (LATIN-1): Nyugat-európai nyelvekhez

ISO/IEC 8859-2 (LATIN-2): Közép-európai nyelvekhez

* + Unicode karakterek kifejezése
    - **CSS**
      * \hhhhhh (6x h) formájú vezérlősorozatok, ahol hhhhhh a Unicode karakter kódpontját ábrázoló, legalább 1, és legfeljebb 6 karakterből álló **hexadecimális** karaktersorozat
    - **JSON**
      * Sztringekben a BMP-hez tartozó Unicode karakterek megadhatóak \uhhhh formájú vezérlősorozatokkal, ahol hhhh a kódpontot ábrázoló 4 **hexadecimális** számjegy, pl: \u00A9
    - **XML, XHTML**
      * Szövegben, attribútumértékekben és literális egyed értékekben Unicode karakterek kifejezhetők az alábbi formájú karakterhivatkozásokkal:
        + &#nnnn, ahol nnnn a kódpontot ábrázoló **decimális** számjegy
        + &#xhhhh, ahol hhhh a kódpontot ábrázoló **hexadecimális** számjegy
    - **HTML**
      * &név formájú nevesített karakterhivatkozások
        + pl: &star (U+2606)

1. **Média típusok**
   * **Mik a média típusok?**
     + Internet protokollon keresztül továbbított tartalmak formátumának jelzésére szolgálnak
     + Két részből állnak:
       - Felső-szintű típus (top-level type)
       - altípus (subtype)
     + pl: text/html; charset=UTF-8
     + Felhasználások:
       - HTML
       - HTTP
       - Levelező kliensek
       - Böngészőprogramok
       - Keresőmotorok
   * **Komponensei:**
     + **Felső-szintű típus(Top-level type)**
       - application: A többi felső szintű típus alá nem sorolható adatokhoz, alkalmazói programok által feldolgozott adatokhoz
       - audio
       - font: nevei kisbetű-nagybetű érzéketlenek.
       - image
       - message, multipart: összetett típusok, melyek lehetővé teszik különféle média típusú objektumok egységbe zárását
       - model: viselkedési vagy fizikai modell reprezentációk típusai (pl. 3d modellek)
       - text: Sok idetartozó alípus definiál egy charset paramétert a karakterkódolás meghatározásához
       - video
     + **altípus (subtype)**
       - Altípus regisztrálása egy adott fába történik a használat körétől függően
         * Neve hordozza, hogy melyik regisztrációs fába tartozik

Nevének elején a *fa.* előtag határozza meg a regisztrációs fát

Regisztrációs fák: Szabványok fája (standards tree), Gyártói fa (vedor tree), Személyes fa (personal or vanity tree), Nem regisztrált x. fa (unregistered x. tree)

* + - **Paraméterek**
      * kisbetű-nagybetű érzéketlenek
      * tetszőleges sorrendben adhatóak meg
      * értékükre nincs előírt szintaxis
      * neveit, lehetséges értékeit és jelentését a média type regisztráció határozza meg
      * pl: Content-Type: message/http; version = 1.0; msgtpye=request
    - **Struktúrált szintaxis utótag**
      * Az altípus nevének a + karaktert követő részét nevezik így
      * A média típus szerkezetét jelzi
      * Ezeket is regisztrálni kell
      * A jelenleg felhasználható utótagok:
        + +ber, +cbor, +cbor-seq, +der, +fastinfoset, +gzip, +json, +json-seq ….
      * Példa a használatukra:
        + application/calendar+json, application/calendar+xml

1. **URI**
   * **Mi az URI?**
     + Egységes erőforrás azonosító (uniform resource identifier)
     + Absztrakt vagy fizikai erőforrást azonosító tömör karaktersorozat
       - Egy erőforrás nem feltétlenül érhető el a weben
       - URI-kat hozzá lehet rendelni akár a tárgyi világ objektumaihoz és fogalmakhoz is
     + Minden URI egy sémanévvel kezdődik, melyet egy ’:’ karakter választ el a séma-specifikus résztől
     + Az URI sémákat adminisztráló szerv: IANA
   * **URI sémák:**
     + file
     + http/https
     + mailto
     + about
   * **URI karakterek, százalékos kódolás:**
     + URI-ban megengedett karakterek
       - Fenntartott karakterek: ':', '/', '?', '#', '[', ']', '@', '!', '$', '&', ''' , '(', ')', '\*' , '+', ',', ';', '='
       - Nem fenntartott karakterek: 'A', …, 'Z', 'a', …, 'z' '0', …, '9', '-' , '.', '\_', '~'
       - A specifikáció nem határoz meg karakterkódolást
     + Százalékos kódolás (percent-encoding):
       - Nem megengedett karakterek használatához vagy fenntartott karakterek speciális jelentésének elnyomásához
       - Például a szóköz karaktert %20 módon kell kódolni
       - használhatóak az A…F és az a…f hexadecimális számjegy karakterek is
       - Példa:
         * file:///media/Movies/What's Up, Tiger Lily? (1966)/
         * Ebből lesz:

file:///media/Movies/What%27s%20Up%2C %20Tiger%20Lily%3F%20%281966%29/

* + **URI szintaxis, a host, port, útvonal, lekérdezés és erőforrásrész-azonosító komponensek**
    - URI szintaxis:
      * Hierarchikus felépítés
        + A komponensek felsorolása balról jobbra haladva fontosság szerint csökkenő sorrendben
        + Általános szintaxis:

*séma ’:’ hierarchikus rész [ ’?’ lekérdezés] [ ’#’ erőforrásrész]*

* + - * + A hierarchikus rész egy autoritás és egy útvonal komponenst tartalmazhat, szintaxisa:

A képen szöveg látható

Automatikusan generált leírás*’//’ autoritás útvonal* vagy *útvonal*

* + - Autoritás komponens:
      * Nevét onnan kapta, hogy fennhatósága alá tartozik az URI további része által meghatározott névtér
      * Szintaxisa:
        + [userinfo ’@’] *host* [’:’ *port*]
        + Az URI sémák meghatározhatnak egy alapértelmezett portot

pl. http sémánál 80 az alapértelmezett port

* + - Útvonal komponens:
      * Útvonal részek ’/’ karakterrel elválasztott sorozata, amely lehet üres
      * Az első ’?’ vagy ’#’ karakterig, ezek hiányában pedig az URI végéig tart
    - Lekérdezés komponens:
      * A ’?’ karakter jelzi az elejét
      * ’#’ karakterig, vagy annak hiányában az URI végéig tart
      * Nem hierarchikus adatokat tartalmaz
      * Gyakran név = érték formájú, & karakterekkel elválasztott név-érték párokat tartalmaz
    - Erőforrásrész-azonosító:
      * Az URI utolsó része
      * # karakterrel kezdődik, az URI végéig tart
      * Lehetővé teszi egy másodlagos erőforrás közvetett azonosítását egy elsődleges erőforrásra történő hivatkozáson keresztül
        + A másodlagos erőforrás lehet az elsődleges erőforrás része
      * Jelentését az elsődleges erőforrás elérése során kapott lehetséges reprezentációk határozzák meg, ezek médiatípusa.
        + A média típusok megszorításokat szabhatnak az erőforrásrész-azonosító formájára, meghatározhatják az így azonosított másodlagos erőforrások jelentését.
      * Hivatkozás-feloldás során mindig eltávolításra kerül
      * A séma specifikációk olyan URI szintaxist kell, hogy meghatározzanak, melynek csak abszolút (erőforrás azonosító részt nem tartalmazó) URI-k felelnek meg.
  + **Az erőforrásrész-azonosító jelentése:**
    - * text/html média típus:

Az erőforrásrész azonosító a dokumentum adott részét jelenti, vagy állapot információt szolgáltat szkriptek számára.

Az erőforrásrész-azonosító feldolgozását részletesen a HTML5 specifikáció határozza meg

Például a https://www.w3.org/blog/news/#w3c\_footer URI esetén az erőforrásrész-azonosító a w3c\_footer azonosítójú elemet jelenti. –

Például a https://www.youtube.com/watch?v=w0ffwDYo00Q#t=77 URI esetén az erőforrásrész-azonosító azt jelzi, hogy mely pozíción kell elkezdeni a videó lejátszását (a 77. másodperctől).

* + - * + application/xml, text/xml és \*/\*+xml média típusok

Az erőforrásrész-azonosító szintaxisa és jelentése az XPointer Framework specifikáción alapul

Például a <https://www.w3.org/TR/xml/#sec-bibliography> URI esetén az erőforrásrész-azonosító a sec-bibliography azonosítójú elemet jelenti a dokumentumban.

* + **URI-hivatkozás, relatív hivatkozás**
    - **Abszolút URI:** olyan URI, amely nem tartalmaz erőforrásrész-azonosítót
    - **URI-hivatkozás:** URI vagy relatív hivatkozás
    - **Relatív hivatkozás:** kb. egy URI séma-specifikus része, vagy annak egy megfelelő végszelete, lehet akár üres karakterlánc is
      * A relatív URI kifejezést a specifikáció nem használja
      * Míg egy URI mindig a használat környezetétől függetlenül azonosít egy erőforrást, egy relatív hivatkozás egy adott környezetben értelmezett
      * egy bázis-URI alapján URI-vá lehet feloldani
      * A relatív hivatkozások feloldásához egy algoritmust ír le a specifikáció
  + **URI-k összehasonlítása:**
    - A séma és a host komponensek kisbetű-nagybetű érzéketlenek.
    - **Ekvivalencia** egy lehetséges definíciója:
      * URI-k akkor ekvivalensek,ha ugyan azt az erőforrást azonosítják
        + Ez a definíció gyakorlati szempontból használhatatlan, mert nincs mód az erőforrások összehasonlítására
      * Gyakorlatban az ekvivalencia megállapítása az URI karakterláncok összehasonlításán alapul

1. **HTTP alapok**
   * **Jellemzők: állapotnélküli protokoll, kiterjeszthetőség**
     + **Mi a http**: Állapotnélküli alkalmazásszintű kérdés/válasz protokollok egy családja, melynek egy közös általános interfésszel, kiterjeszthető szemantikával, és önleíró üzenetekkel teszik lehetőé a rugalmas interakciót hálózati alapú hiperszöveg információs rendszerekkel.
     + **Állapotnélküli protokoll:**
       - Minden egyes kérés jelentése a többiétől külön értelmezhető
     + **Kiterjeszthető:**
       - A HTTP számos általános kiterjesztési pontot határoz meg, melyek révén új verzió kiadása nélkül vezethetők be a képességek a protokollba, mint például metódusok, állapotkódok vagy mezők
   * **A http és https URI séma**
     + A HTTP definiálja a http és https URI sémákat
       - Egy http vagy https URI eredet szerverét a host azonosító és az opcionális portszám határozza meg
       - Az útvonal komponens és az opcionális lekérdezés komponens egy lehetséges cél erőforrást azonosít az eredet szerver névterében.
     + Egy http vagy https URI előfordulása nem jelenti azt, hogy mindig egy HTTP szerver figyel az URI által meghatározott eredet szerveren.
     + A https sémán keresztül elérhető erőforrásoknak nincs közös identitása a http sémával
     + A http URI séma
       - Célja, hogy lehetővé tegye erőforrások azonosítását egy olyan potenciális eredet szerveren, mely egy adott porton vár TCP kapcsolatokra
       - szintaxis:

http://" host [":" port] [útvonal] ["?" lekérdezés]

Ha nincs megadva vagy üres a port alkomponens, akkor a 80-as TCP port az alapértelmezés

* + - A https URI séma
      * Célja, hogy lehetővé tegye erőforrások azonosítását egy olyan potenciális eredet szerveren, mely egy adott porton vár TCP kapcsolatokra és mely képes http kommunikációhoz biztonságossá tett TLS kapcsolat létrehozására
      * URI szintaxis

https://" host [":" port] [útvonal] ["?" lekérdezés]

Ha nincs megadva vagy üres a port alkomponens, a 443-as TCP port az alapértelmezés

* + **Üzenet absztrakció**
    - Az RFC 9110 az üzenetek egy olyan absztrakcióját határozza meg, mely szerint egy üzenet a következőkből áll:

Vezérlő adatok

Fejléc szakasz

Tartalom

Lezáró szakasz

* + - Ez az üzenet absztrakció egy több http verziót átfogó általánosítás, melynek vannak olyan lehetőségei, melyek bizonyos verziókban nem találhatók meg
    - Az üzeneteket önleírónak szánják, azaz minden, amit egy fogadónak tudnia kell az üzenetről, megállapítható a (dekódolt) üzenet vizsgálatával.
    - **Vezérlő adatok:**
      * Az üzenetek az elsődleges céljukat leíró vezérlő adatokkal kezdődnek
        + Kérésekben a vezérlő adatok a metódust, a kérés célt és a protokoll verziót foglalják magukban
        + Válaszokban a vezérlő adatok az állapotkódot, az opcionális indok frázist és a protokoll verziót foglalják magukban
      * Minden HTTP üzenetnek van protokoll verziója
    - **Fejléc szakasz**
      * A tartalom előtt küldött vagy fogadott mezőket fejlécmezőknek vagy fejléceknek nevezzük
      * Egy üzenet fejléc szakasz fejléc mezősorok sorozatából áll
    - **Tartalom**
      * a HTTP üzenetek egy teljes vagy részleges reprezentációt hordozhatnak az üzenet tartalmaként
      * A tartalom átvitele egy bájtfolyamként történik a fejléc szakasz után
      * A Content-Type és a Content-Encoding fejlécmezők által meghatározott formátumban és kódolásban van
      * Tartalom szemantika:

A tartalom célját egy kérésben a metódus szemantika határozza meg

Például egy reprezentáció egy POST kérés tartalmaként a cél erőforrás által feldolgozandó információkat ábrázol.

Egy válaszban a tartalom célját a kérés metódusa, az állapotkód és a tartalmat leíró válasz mezők határozzák meg.

Például egy GET kérésre adott 200 (OK) állapotkódú válasz a cél erőforrás az üzenet létrehozásának pillanatában megfigyelhető aktuális állapotát ábrázolja

* + - **Lezáró szakasz**
      * A tartalom után küldött vagy fogadott mezőket lezáró mezőknek nevezzük
      * A lezáró mezők ellenőrző összegek, digitális aláírások, kézbesítési metrikák vagy utófeldolgozási állapot információk továbbítására használhatók
      * Egy üzenet lezáró szakasz lezáró mezősorok sorozatából áll
  + **Mezők**
    - A HTTP mezőket használ adatok név/érték párok formájában történő szolgáltatásához
    - Mezők szolgálnak a következő információk továbbítására
      * magát az üzenetet leíró metaadatok kérésekben és válaszokban (pl. DATE)
      * reprezentáció metaadatok kérésekben és válaszokban (pl. Content-Type)
      * információk a kliensről kérésekben (pl. User-Agent)
      * információk a szerverről válaszokban (pl. Server)
      * erőforrás metaadatok válaszokban (pl. Last-Modified)
    - A mezők küldése és fogadása az üzenetek fejléc és lezáró szakaszaiban történik
      * Egy üzenet fejléc vagy lezáró szakaszában küldött mezőt fejlécmezőnek, illetve lezáró mezőnek nevezünk
      * Bizonyos mezők, mint pl. az ETag, előfordulhatnak fejléc- vagy lezáró mezőként is.
    - **Mezőnevek:**
      * Egy mezőnév egy vagy több karakterből álló olyan sorozat, melyben az US-ASCII karakterkészlet egy részhalmaza használható csak.
      * A mezőnevek kisbetű-nagybetű érzéketlenek.
    - **Mezőértékek:**
      * Egy mezőérték egy olyan karaktersorozat, mely egy vagy több nyomtatható US-ASCII karakterből, szóközből és vízszintes tabulátorból áll.
      * A vezető és záró whitespace karaktereket a felhasználás előtt el kell távolítani.
      * A mezőkhöz előírható, hogy egyetlen tagot vagy pedig egy vesszővel elválasztott taglistát hordozzanak
      * Minden egyes mező korlátozhatja a megengedett értékek halmazát.
    - **Mezőszakaszok:**
      * Tetszőleges számú olyan mezősorból állnak, melyek mindegyike egy mezőnevet és egy hozzá tartozó mezősor értéket tartalmaz.
      * Amikor egy mezőnév megismétlődik egy szakaszban, az értéke egy olyan lista, melyben a mező a szakaszbeli mezősor értékei az előfordulásuk sorrendjében kerülnek összefűzésre, elválasztóként egy vessző karakterrel
      * Nem lényeges, milyen sorrendben fordulnak elő különböző mezőnevű mezősorok egy szakaszban
    - A HTTP specifikációk sok szabványos mezőt határoznak meg.
    - Az IANA adminisztrálja a HTTP mezőket.
  + **A User-Agent fejlécmező**
    - A felhasználói ágensről tartalmaz információkat, ahonnan a kérés származik.
    - Felhasználható a válasz testreszabásához, vagy pedig böngésző vagy operációs rendszer használatra vonatkozó elemzésekre.
    - A felhasználói ágens számára ajánlott minden egyes kérésben elküldeni a fejlécmezőt
    - A mezőérték egy vagy több termékazonosítóból áll, melyek mindegyikét nulla vagy több megjegyzés követi
      * A termékazonosítók felsorolása a fontosságuk szerint csökkenő sorrendben történik
      * Minden egyes termékazonosító egy névből és egy opcionális verziószámból áll.
      * A képen szöveg látható

        Automatikusan generált leírásA megjegyzéseket zárójelek határolják.
  + **A HTTP metódusok**
    - Elsődlegesen a metódusok jelzik a kérések célját.
    - A metódus szemantika tovább specializálható a kérésben bizonyos fejlécmezőkkel
    - A metódusnevek kisbetű-nagybetű érzékenyek.
    - A specifikáció által meghatározott szabványos metódusok:
      * CONNECT
      * **DELETE**
      * **GET**
      * **HEAD**
      * OPTIONS
      * **POST**
      * **PUT**
      * TRACE
    - Minden általános célú szervernek támogatnia kell a GET és HEAD metódusokat, a többi opcionális.
    - Az IANA adminisztrálja a HTTP metódusokat.
    - Egy cél erőforrás által megengedett metódusok az *Allow* fejlécmezőben sorolhatók fel
    - Ha a szerver nem ismeri fel vagy nem implementálja a metódust: *501: Not implemented* állapotkóddal ajánlott válaszolnia.
    - Ha felismeri és implementálja, de nem engedélyezi a cél erőforrásra: *405: Method not allowed*
    - **GET Metódus:**
      * A cél erőforrás egy aktuális kiválasztott reprezentációjának átvitelét kérelmezi
      * Az információ-visszakeresés elsődleges mechanizmusa
      * Egy kliens úgy módosíthatja a GET jelentését a kérésben a Range fejlécmező küldésével, hogy az csak a kiválasztott reprezentáció bizonyos részeinek átvitelét kérelmezi
    - **HEAD metódus:**
      * Azonos a GET metódussal, azzal a különbséggel, hogy a szerver nem küldhet tartalmat a válaszban.
      * A képen szöveg látható

        Automatikusan generált leírásÚgy használható metaadatok szerzésére kiválasztott reprezentációról, hogy reprezentáció adatok nem kerülnek átvitelre
    - **POST metódus:**
      * Azt kérelmezi, hogy a cél erőforrás dolgozza fel a kérésben mellékelt reprezentációt a saját szemantikájának megfelelően
      * Tipikus felhasználások:

Adatok küldése egy adatfeldolgozó folyamat számára

Egy üzenet postázása egy hírcsoportba, levelezési listára vagy blogba.

Egy új erőforrás létrehozása

A képen szöveg látható

Automatikusan generált leírásAdatok hozzáfűzése egy erőforrás létező reprezentációjához

* + - * **PUT metódus**

Azt kérelmezi, hogy kerüljön létrehozásra vagy helyettesítésre a cél erőforrás állapota a kérésben mellékelt reprezentáció által meghatározott állapottal.

A képen szöveg látható

Automatikusan generált leírásEgy adott reprezentációt tartalmazó sikeres PUT kérés arra enged következtetni, hogy egy következő GET kérés ugyanarra a cél erőforrásra egy 200 OK állapotkódú válaszban elküldött ekvivalens reprezentációt eredményez.

* + - * **DELETE metódus**

Azt kérelmezi, hogy az eredet szerver törölje a cél erőforrás és aktuális funkcionalitása közötti kapcsolatot.

Ha a cél erőforrásnak egy vagy több aktuális reprezentációja van, akkor ezeket az eredet szerver vagy megsemmisíti, vagy nem, a kapcsolódó tárterület vagy felszabadításra kerül, vagy nem, az erőforrás természetétől és az eredet szerver általi implementációtól függ.

A képen szöveg látható

Automatikusan generált leírásViszonylag kevés erőforrás engedi meg a DELETE metódust.

* + - **Állapotkódok és fajtáik**
      * Egy válasz állapotkódja egy olyan háromjegyű decimális egész szám, mely a kérés eredményét és a válasz jelentését írja le, beleértve azt, hogy sikeres volt-e a kérés, és hogy milyen tartalom van benne mellékelve
      * Az összes érvényes állapotkód a 100 és 599 közötti tartományba esik.
      * Első számjegy a válasz fajtáját határozza meg:

**1xx (informáló):** A kapcsolat állapotát vagy a kérés előrehaladását közli

**2xx (siker):** a kérés sikeresen fogadva, feldolgozva, elfogadva

**3xx (átirányítás):** a felhasználói ágensnek további műveleteket kell végeznie, amit automatikusan el is végezhet

**4xx (kliens hiba):** a kérés rossz szintaxisú vagy nem értelmezhető

**5xx (szerver hiba):** azt jelzi, hogy az érvényes kérést a szerver nem teljesítette.

* + - * **Egy kliensnek nem kell értenie az összes állapotkódot, azonban a fajtáját kötelező tudnia, és megfelelően cselekedni.**
      * Főbb állapotkódok:

100 Continue

101 Switching Protocols

**200 OK Sikeres kérés**

201 Created

**202 Accepted Elfogadott kérés, folyamatban a feldolgozás**

204 No Content

206 Partial Content

301 Moved Permanently Az URI véglegesen megváltozott

302 Found ideiglenesen másik URI

400 Bad Request

401 Unauthorized érvénytelen hitelesítő adatok

403 Forbidden megtagadja a kérés teljesítését

404 Not Found

405 Method not allowed

500 Internal Server Error

501 Not implemented

503 Service unavailable

* + A képen szöveg látható

    Automatikusan generált leírásA képen szöveg látható

    Automatikusan generált leírásA képen szöveg látható

    Automatikusan generált leírás**Átirányítás**
  + **Tartalomegyeztetés**
    - Amikor a válaszok akár sikert, akár hibát jelző tartalmakat hordoznak, az eredet szerver gyakran többféle módon is ábrázolhatja a tartalmat, például különböző formátumokban, nyelveken vagy kódolásokkal
    - A különböző felhasználóknak / felhasználói ágenseknek eltérhet, melyik reprezentáció lenne számukra a legjobb
    - Ezért biztosít a HTTP a tartalomegyeztetéshez mechanizmusokat
    - Két fő tartalomegyeztetési minta:
      * **Proaktív egyeztetés (proactive negotiation):**

A szerver választja ki a reprezentációt a felhasználói ágens preferenciája alapján

**Szerver vezérelt (server driven) tartalomegyeztetés néven ismert.**

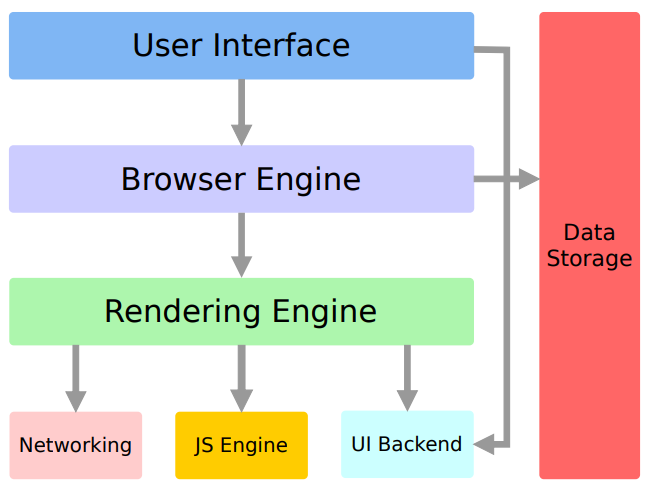
* + - * **Reaktív egyeztetés (reactive negotiation):**

A szerver választásra kínálja fel a felhasználói ágensnek a reprezentációk listáját.

**Ágens vezérelt (agent-driven)**

* + **Proaktív egyeztetés:**
    - A szerver algoritmussal választ egy reprezentációt a felhasználói ágens preferenciája alapján
    - **Előnyös:**
      * Amikor nehéz egy felhasználói ágens számára leírni a rendelkezésre álló reprezentációk közüli választás algoritmusát, vagy
      * amikor a szerver az első válaszban el kívánja küldeni az általa legjobbnak becsült reprezentációt
    - **Hátrányok:**
      * Lehetetlen a szerver számára meghatározni, hogy mi lenne a legjobb a felhasználónak
      * Nagyon nem hatékony a felhasználói ágens képességeit minden kérésben leírni, kockázatot is jelenthet
      * Bonyolítja az eredet szerver megvalósítását
      * Korlátozza a válaszok újrahasznosíthatóságát
    - **A képen szöveg látható

      Automatikusan generált leírás**A felhasználói ágensek az alábbi mezőket használhatják preferenciáik megadásához:
  + **Accept fejlécmező**
    - Értéke média tartományok egy vesszővel elválasztott listája, melyben minden egyes média tartományt 0 vagy több médiatípus paraméter (pl. charset), valamint egy opcionális relatív súly (azaz q paramméter) követhet.
    - Média tartomány:
      * \*/\* az összes médiatípus
      * type/\*: az adott type összes altípusa pl. text/\*
      * types/subtype: az adott médiatípus pl. text/html
  + **HTTP/1.1**
    - Szöveg alapú protokoll
    - Egy HTTP/1.1 üzenet kérés vagy válasz lehet
      * Az üzenetek egy kezdősorral kezdődnek, melyet CRLF követ (Carriage Return Line Feed)
    - A kezdősort nulla vagy több fejlécsor követi, melyeket együttesen fejléceknek vagy fejléc szakasznak neveznek
    - Egy üres sor jelzi a fejléc szakasz végét
    - Opcionálisan szerepelhet az üzenet végén egy üzenettörzs (message body)
    - A kérések kezdősorának szintaxisa:
      * <metódus> <kérés-cél> „HTTP/1.1”
    - A kérés-cél a cél erőforrást azonosítja, melyre a kérés vonatkozik
    - A válaszok első sorát állapotsornak nevezik és a következő szintaxisa van:
      * „HTTP/1.1” <állapotkód> [indok\_frázis] (Az opcionális indok frázis az állapotkód szöveges leírása)
    - Mezősorok:
      * Minden mezősor egy kisbetű-nagybetű érzéketlen mezőnévvel kezdődik, melyet egy pontosvessző, opcionális vezető *whitespace*, egy mezősor érték és táró *whitespace* követ.
      * A mezősorokat egy CRLF zárja

1. **Webböngészők**
   * **A böngészők felépítése és komponense**
     + **Felhasználói felület**
       - Címsor, gombok, menük, stb…
       - Beletartozik a böngésző minden látható része, kivéve az a terület, ahol a weboldal megjelenik.
     + **Böngésző motor**
       - Egy magas szintű interfészt biztosít a renderelő motor manipulálásához
       - A renderelő motor és a grafikus felület között játszik közvetítő szerepet
       - Ez kezeli a tárhelyet, az adattároló komponenst
     + **Renderelő motor**
       - Ez a komponens hoz létre HTML, XML stb… kódból a felhasználó által látott weboldalt. (Ez meghatározott formázási szabályoknak megfelelően történik)
     + **Networking – hálózati komponens**
       - Ez a komponens felel a hálózati kommunikáció kezeléséért
         * HTTP kérések és válaszok
         * WebSocket API
         * WebRTC
       - Egy platformfüggetlen interfészt biztosít, mely mögött platform specifikus API-k kerülnek felhasználásra
     + **JS motor**
       - A JavaScript kódok végrehajtását szolgálja
     + **UI backend**
       - A böngészőben rajzolásért felelős
       - Egyaránt használt a felhasználói felület elemeinek, és a weboldalnak megjelenítéséhez.
       - Szintén platform független interfész, platform specifikus API-k felhasználásával.
     + **Tárhely**
       - Adatok perzisztens tárolása
         * HTTP sütik
         * HTTP gyorsítótárazás
         * Indexed Database API
         * Web Storage API
   * **A fő renderelő motorok, a fő asztali és mobil böngészők**
     + **WebKit**
       - Fejlesztő: Apple
       - Programozási nyelv: C++
       - WebKit-en alapuló szoftverek:

Safari

Google Chorme **for IOS**

Firefox **for IOS**

GNOME Web

* + - **Blink**
      * Fejlesztő: Chromium Project (Google)
      * Programozási nyelv: C++
      * Eredetileg a WebKit WebCore komponensének elágazásaként született
      * Szoftverek:

Chromium

Google Chrome, Google Chrome for Android

Chromium alapú Microsoft Edge

Opera

Vivaldi

* + - **Gecko**
      * Fejlesztő: Mozilla Project
      * Programozási nyelvek: C, C++, JavaScript, Rust
      * Szoftverek:

Firefox

Firefox for Android

SeaMonkey

* + - **Fő asztali böngészők:**
      * Chromium, Google Chrome
      * Firefox
      * Opera
      * Safari
      * Microsoft Edge
    - **A főbb mobil böngészők** 
      * Google Chrome (Android, iOS)
      * Safari for iOS
      * Samsung Internet for Android
      * UC Browser (Android, iOS)
  + **Böngésző kiegészítők, kiterjesztések**
    - Egy böngésző kiegészítő a megjelenését, vagy viselkedését testreszabó szoftvermodul
      * kiterjesztések, nyelvi csomagok, témák, plugin-ek
    - **Kiterjesztések:**
      * A böngésző viselkedését módosítja, új funkcionalitást ad hozzá, vagy meglévőt módosít
      * Webtechnológiákon alapulnak, mint például HTML, CSS, JavaScript
      * Minden kiterjesztésnek van egy .manifest állománya, mely metaadatokat tartalmaz, és egybefogja a komponenseket.
      * Apple, Google, Microsoft, Mozilla kezdeményezte: WebExtensions Community Group
        + Kiterjesztések szabványosítása
        + Ne kelljen teljesen újraírni a kiterjesztéseket ahhoz, hogy különböző böngészőkben működjenek
    - **Headless böngészők:** Egy headless böngésző egy *grafikus felhasználói felület nélküli* böngésző
      * Programozottan vezérelhető
    - Alkalmazások:
      * Webalkalmazások automatikus tesztelése
      * Weboldakkal történő interakció automatizálása
      * Képernyőképek készítése weboldalakról
      * Információ kinyerés weboldalakból (web scraping)
    - A Chromium, Google Chrome és Firefox tudnak **fej nélküli üzemmód**ban futniA képen szöveg látható

      Automatikusan generált leírás
    - Szabad és nyílt forráskódú headless böngészők:
      * HtmlUnit
      * Puppeteer
  + **Az *about* URI séma**
    - A böngészők használják ahhoz, hogy leírást adjanak erőforrásaikhoz
      * pl.: az *about:blank* URI egy üres oldalra hivatkozik
    - Néhány böngésző az *about* nevet saját nevére cseréli
      * Google chrome pl. *about:about* helyett *chrome://about* URI-t használja.
      * Az about:blank kivétel, melyet változatlanul hagynak.
    - Firefox:
      * Az *about* URI-k listája: *about:about*
    - Opera:
      * A chrome URI-k egy részét támogatja
    - Safari:
      * Csak az *about:blank* URI-t ismeri fel.
    - Chromium alapú Edge:
      * *edge://about*

1. **Markdown**
   * **Jelölőnyelvek példákkal**
     + A jelölőnyelvek szövegek annotálására szolgáló számítógépes nyelvek. A szövegrészekhez metaadatokat adhatunk meg a szövegtől jól elkülöníthető módon.
     + példák:
       - AsciiDoc
       - MarkDown
       - TeX, LaTeX
       - troff
       - Wikitext
       - XML
   * **Mi a Markdown?**
     + Egy pehelysúlyú jelölőnyelv sima szöveges formázási szintaxissal
     + John Gruber fejlesztette Aaron Swartz közreműködésével 2004-ben.
   * **A Markdown jellemzői**
     + Könnyen olvasható, könnyen írható
     + Különféle formátumokba alakítható (pl. HTML)
     + Nem szükséges speciális ismeretek
     + Szintaxisát sima e-mail formázás ihlette
     + **Állomány kiterjesztés**: .md
     + **Média típus:** text/markdown
   * **Szabványosítás, Markdown változatok**
     + **CommonMark**
       - A markdown egy egyértelmű szintaxis specifikációja
     + **Változatok:**
       - GitHub Flavored Markdown (**GFM**)
       - Pandoc’s Markdown
   * **Gyakorlati felhasználások**

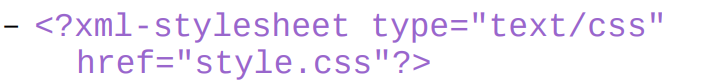
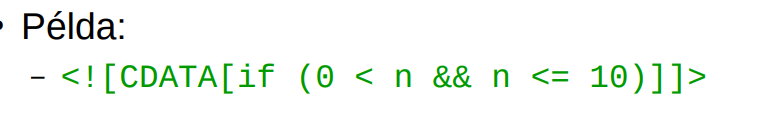
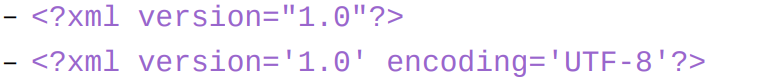
* A Markdown-t széles körben használják a **weben szövegbevitelre**.
* A fő alkalmazási területek közé tartoznak:
  + **Kollaborációs** eszközök és platformok (**pl.** GitHub, MS Teams, Trello)
  + **Blogolási** platformok és **tartalomkezelő** **rendszerek** (**pl.** Moodle, WordPress)
  + Online **közösségi** **platformok** (**pl.** reddit, Stack Overflow)
  + **Műszaki** és **tudományos** **publikálás** (**pl.** bookdown, GitBook, MS Docs, Daniel Stenberg. Everything curl)
  + **Mi az R Markdown?**
* A Markdown egy kiterjesztése, mely lehetővé teszi végrehajtható programkódok beágyazását a dokumentumokba.

1. **XML bevezető**
   * **Mi az XML?**
     + Az XML egy általános célú jelölőnyelv, mely az 1990-es évek második felében született.
     + Napjaink meghatározó, széles körben használt technológiája
       - Szűkebb értelemben: szintaxis strukturált dokumentumok ábrázolására, ezzel lehetővé téve automatikus feldolgozását
       - Tágabb értelemben:egy sereg közös tőről fakadó specifikációt jelent, melyet XML családnak neveznek
   * **Az XML és HTML összehasonlítása**
     + **XML:**
       - Nincs előre definiált címkészlet
       - Célja az adatok leírása
       - Adatcsere formátumként használjak
     + **HTML:**
       - Előre definiált címkészlet
       - Célja az információ megjelenítése
       - Prezentációs nyelv
       - Tekinthető az XML speciális alkalmazásának (XHTML)
   * **XML előnyei és hátrányai**
     + **Előnyök:**
       - Egyszerű szöveges állományok, bármilyen szövegszerkesztővel létrehozhatóak
       - Nyílt, gyártófüggetlen, platformfüggetlen
       - Univerzális adatcsere formátum
       - Kiterjedt infrastruktúra
       - Az iparban de-facto szabvány
     + **Hátrányok**
       - Bőbeszédű és nehezen használható szintaxis
       - Nagy tárigény
       - Bonyolultság
         * Sok XML-hez kötődő specifikáció
   * **Dokumentum-központú és adatközpontú XML**
     + **Dokumentum-központú**
       - A dokumentumokat jelölésekkel ellátott folyó szöveg alkotja
       - A szerkezete változatos
       - Elemek sorrendje lényeges
       - Tartalma emberi fogyasztásra szánt
         * például: XHTML
     + **Adatközpontú:**
       - A dokumentumokat nagyszámú adatelem alkotja
       - Nem annyira változatos szerkezet
       - Nem annyira lényeges az elemek sorrendje
       - Gépi feldolgozásra szánt
         * Például: SVG
2. **XML 1.0**
   * **XML dokumentumok, jólformáltság**
     + Olyan szöveges objektumok, melyek a szabvány előírása szerint jólformáltak
       - Fizikai és logikai szerkezetük van
         * **Fizikailag:** Egyedeknek nevezett tárolási egységekből állnak
         * **Logikailag:** deklarációkból, elemekből, megjegyzésekből, feldolgozási utasításokból és más részekből állnak
     + **Jólformáltság:**
       - Egyetlen felső szintű elem, a **gyökérelem** tartalmazza az összes többi elemet
       - Minden nyitó címkéhez tartozik záró
       - Az elemek egymásba ágyazottak, nem fedhetik egymást
       - Minden hivatkozott egyed is jólformált
   * **Elemek**
     + Minden elemet nyitó és zárócímke határol, vagy egyetlen üres elem címke alkotA képen szöveg látható

       Automatikusan generált leírás
     + A címkékben a nevet **elemtípus**nak nevezzük. A nyitó és zárócímkében a neveknek meg kell egyezni.
     + A nyitó és zárócímke fogja közre az elem tartalmát
     + Az elemekhez meg lehet adni attribútumspecifikációknak nevezett név-érték párokat
     + Üres elem: Tartalom nélküli elem
       - <elem></elem>
       - vagy: <elem/>
   * **Spec. karakterek ( & és <)**
     + A „&” és „<” karakterek ebben a formában kizárólag jelölő-határolóként, megjegyzésekben, feldolgozási utasításokban és CDATA-szakaszokban fordulhatnak elő.
     + Helyettük egyéb helyeken karakterhivatkozásokat, az &amp és &lt egyedhivatkozásokat kell használni
   * **Jelölők:** 
     + **Nyitó:** <title>, <title xml:lang=”en”>
     + **Záró:** </title>
     + **Üres elem:** <br/>, <hr />, vagy <img src=”logo.png” alt=”Logo”>
     + **Karakterhivatkozás:** 
       - * &#nnnn, ahol nnnn a kódpontot jelentő **decimális** számjegysorozat: &#169, &#9775
         * &#xhhhh, ahol hhhh a kódpontot jelölő **hexadecimális** számjegysorozat: &#xA9, &#x262F
     + **Egyedhivatkozás:**
       - Hivatkozás egy névvel azonosított egyed tartalmára

elemzett általános egyedre: *&név:* &amp

paraméteregyed-hivatkozás: *%név:* %inline

* + - **Megjegyzés**
      * A dokumentumban bárhol szerepelhetnek más jelölőkön kívül 
    - **Feldolgozási utasítások**
      * Az alkalmazások számára tartalmaznak utasításokat 
    - **CDATA-szakaszok**
      * Bárhol előfordulhatnak, ahol előfordulhat karakteres adat
        + Olyan karakterek levédését szolgálja, amelyek egyébként jelölőnek lennének tekintve 
    - **XML-deklaráció**
      * Az XML dokumentumoknak egy deklarációval ajánlott kezdődniük. Tartalmazza az XML verziót és karakterkódolást 
    - **Dokumentumtípus-deklaráció**
      * Dokumentumok egy osztályához egy nyelvtant meghatározó jelölő deklarációkat tartalmaz és/vagy ilyen deklarációkra mutat.
        + Ezt a nyelvtant dokumentumtípus-definíciónak, **DTD-**nek nevezik
      * Egy **külső DTD alkészlet**-nek nevezett, jelölő deklarációkat tartalmazó külső egyedre mutató dokumentumtípus-deklaráció: A képen szöveg, beltéri, képernyőkép látható

        Automatikusan generált leírás A képen szöveg látható

        Automatikusan generált leírás A képen szöveg látható

        Automatikusan generált leírás
  + **Dokumentumtípus-definíció, érvényesség**
    - Lehetővé teszi a megszorítások előírását a dokumentum logikai szerkezetére
    - Jelölő deklarációkból áll
    - A dokumentumtípus-deklarációban társítható egy dokumentumhoz
    - **Érvényesség:** Egy XML dokumentum akkor érvényes, ha tartozik hozzá dokumentumtípus-deklaráció és a dokumentum eleget tesz a DTD által kifejezett megszorításoknak *– Érvényességi megszorítások –*
      * + A dokumentumtípus-deklarációban adott név meg kell, hogy egyezzen a gyökérelem típusával
        + A dokumentum minden eleme deklarálva kell legyen a DTD-ben, és az elem tartalmának meg kell felelnie a deklarációnak
        + A dokumentum minden attribútumának deklarálva kell lennie a DTD-ben, és itt is, ezeknek az attribútumoknak az értékeinek meg kell felelnie a deklarációban leírtaknak
        + Ha egy attribútum kötelezőnek deklarált a DTD-ben, akkor az elemtípus minden előfordulásánál meg kell adni
  + **Elemtípus-deklarációk: üres elem, elemtartalom, vegyes tartalom**
    - **Elemtípus deklaráció:** ez ír elő megszorítást az elem tartalmára. Egy elemtípus nem deklarálható egynél többször
    - **Üres elemek:** Az így deklarált elemeknek **NEM LEHET** tartalma A képen szöveg látható

      Automatikusan generált leírás
    - **Elemtartalom:** Egy elemtípus akkor elemtartalmú, ha csak elemgyermekeket, azaz más elemeket tartalmazhatnak, karakteres adatot vagy szöveget nem. Ez elemgyermekeket opcionálisan whitespace karakterekkel el lehet választani. (Megjegyzések és feldolgozási utasítások lehetnek elemgyermekek között)
      * **A tartalommodell megadásánál használható konstrukciók**
        + Sorozat: (street, city, zip, country)
        + Lista: ( ul | ol | dl )
        + Az előfordulások számát szabályozó speciális karakterek

? 0 vagy 1 előfordulás

+ legalább 1

\* tetszőleges A képen asztal látható

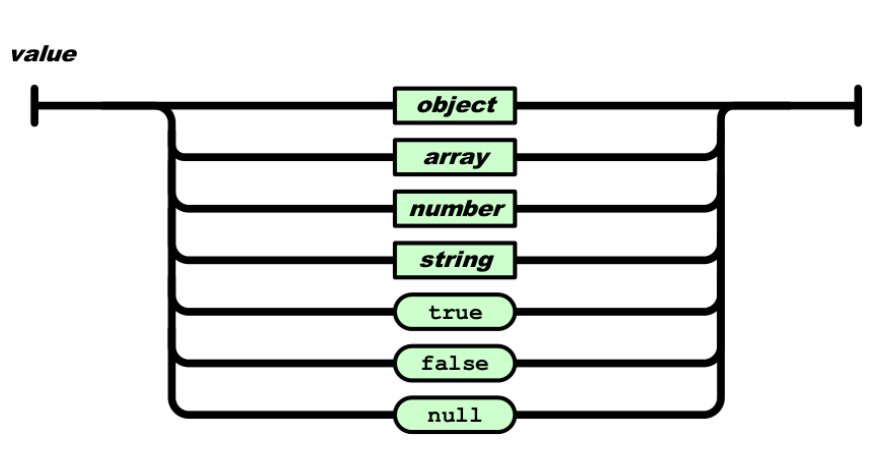
Automatikusan generált leírás

* + - **Vegyes tartalom:**
      * Egy elemtípus vegyes tartalmú, ha tartalmazhatnak karakteres adatot elemgyermekekkel vegyítve. Az elemgyermekek típusa korlátozható, de sorrendjük és számuk nem. A képen szöveg látható

        Automatikusan generált leírásA képen szöveg látható

        Automatikusan generált leírás

1. **JavaScript (JavaScript Object Notation)** **/ ECMAScript**
   * **Mi a JavaScript / ECMAScript?**
     + **Az ECMAScript** a JavaScropt nyelv szabványosítása. Jelenleg a 13-as kiadás aktuális.
     + **A JavaScript** kifejezést használják az ECMAScript különböző gyártók általi megvalósításaira.
   * **JS motor, fő JS motorok**
     + **Spidermonkey**: C/C++ programozási nyelv, Mozilla Public Lincense 2.0
       - A Mozilla projekt JS motorja.
     + **V8**: C++, New BSD License
       - A Chromium JS motorja.
     + **JavaScriptCore:** C++, LGPLv2 licensz.
       - A WebKit renderelő motorjához fejlesztett JS motor
     + Nashorn
       - Oracle JS motorja, JDK 8,9,10 tartalmazza.
     + GraalVM Community Edition
       - Linux, macOS, Windows OS-ek esetén áll rendelkezésre
     + Hermes
       - React Native alkalmazások optimalizált JS motorja
     + JerryScript
       - Könnyű JS motor IoT eszközökre (pehelysúlyú)
   * **Mi a Node.js?**
     + **A Node.js a V8 JS motorra épülő JS futtató környezet,** melyet skálázható hálózati alkalmazások létrehozására terveztek.
       - MIT License
       - C++, JavaScript programozási nyelvek
       - Linux, macOS, Windows platform
     + Lehetővé teszi a **böngészőn kívül futó** JS alkalmazások létrehozását
     + Kliens- és szerver oldali alkalmazások fejlesztésére
     + A csomag ökoszisztémája az **npm**, a világ legnagyobb nyílt forráskódú ökoszisztémája.
2. **JSON**
   * **Mi a JSON**
     + **JavaScript Object Notation**.
       - Könnyűsúlyú, szöveges, nyelvfüggetlen adatcsere formátum.
       - Strukturált adatokat ábrázol
       - Ember számára is könnyen olvasható és írható
       - Szoftverek által könnyen generálható és feldolgozható
       - Az ECMAScript programozási nyelvből származik.
     + Alkotója és népszerűsítője Douglas Crockford, 2001-ben alkotta. Eredetileg JS kliensek és Java szerverek közötti kommunikációra használta.
     + Állomány jellemzők:
       - .json
       - IANA média típus: *application/json*
   * **A JSON és az ECMAScript összehasonlítása**
     + Az ECMASript 2019-es verziójától a JSON már az ECMAScript szintaktikai részhalmaza.
     + A JSON adatcserére használt szöveges formátum.
     + Az ECMAScript egy szabvány, ami leírja a JavaScript nyelv működését.
   * **A JSON és XML összehasonlítás**
     + A JSON az XML alternatívájaként használható **adatcseréhez**
     + Ugyanazon előnyök, a hátrányok nélkül (JSON jobb)
     + Közös jellemzők:
       - Egyszerűség (JSON nyer)
       - Könnyen olvasható és írható
       - Könnyen generálható és feldolgozható (JSON nyer)
       - Interoperabilitás (A különböző infromatikai rendszerek együttműködésre való képessége.)
       - Nyíltság
       - Önleíró ábrázolás
       - Univerzális adatcsere formátumok
     + Fő különbség:
       - JSON **adat-orientált**
       - XML **dokumentum-orientált**
     + Adatszerkezetek ábrázolásához: **JSON**
     + Dokumentum-központú alkalmazásokhoz: **XML**A képen szöveg látható

       Automatikusan generált leírás 
   * **Primitív típusok: string, számok, logikai értékek, null**
     + **Tokenek**
       - A JSON szöveg tokenek olyan sorozata, mely megfelel a JSON érték nyelvtani szabályának
       - Szerkezeti tokenek: { } [ ] : , karakterek
       - Sztringek
       - Számok
       - Literális tokenek a ***true false null*** karakterláncok
     + **Számok**
       - Nincs tartományi vagy pontossági korlátozás
       - Interoperabilitás szem előtt tartása, érdemes dupla pontosságú lebegőpontos számokat használni.
     + **Stringek**
       - Unicode karakterek sorozatai, melyek idézőjelek határolnak.
       - Bármely karaktereket tartalmazhatják, de az alábbiakat csak levédve:

idézőjel, backslash, vezérlő karakterek

* + - * Speciális karakterek megadásához \", \\, \t, \n, \r.
      * A BMP-hez tartozó Unicode karakterek megadhatóak \unnnn módon, ahol nnnn a karakterkód **hexadecimálisan** ábrázolva.
  + **Strukturált típusok: tömbök, objektumok**
    - **Tömbök**
      * Tetszőleges számú érték rendezett sorozata (lehet üres)
      * Az elemek tetszőleges típusúak (akár tömbök is)A képen szöveg látható

        Automatikusan generált leírás
    - **Objektumok**
      * Tetszőleges számú név-érték párokból állnak
      * A név tetszőleges sztring, az érték tetszőleges JSON érték
      * Ezekre a párokra a *tag* nevet is használjukA képen szöveg látható

        Automatikusan generált leírás
  + **Az XML és JSON közötti konverzió** A képen szöveg látható

    Automatikusan generált leírásA képen szöveg látható

    Automatikusan generált leírásA képen szöveg látható

    Automatikusan generált leírás
  + **JSON Schema**
    - **JSON alapú sémanyelv a JSON dokumentumok érvényesítéséhez**
    - Legszélesebb körben támogatott verzió a 2018-as draft-07
    - **JSON dokumentum:** az application/json média típus által leírt információ erőforrás, azaz egy JSON érték
    - **Példány:** egy olyan JSON dokumentum, melyre egy séma vonatkozik.
    - **JSON séma:** egy olyan JSON dokumentum, mely példányokat ír le.
      * Egy objektum vagy egy logikai érték
      * Sémák egymásba ágyazhatóak
        + A legkülső a gyökér séma, a többi alséma
      * Média típus: application/schema+json
    - **Tulajdonság:** egy objektum példány egy tagja (név-érték pár)
    - **Kulcsszó:** egy séma objektum egy példányra vonatkozó tulajdonsága
      * A kulcsszavak JSON példányokra vonatkozó megszorításokat fejeznek ki
        + pl.: „properties”, „type”, „$ref”
    - **Szótár:** adott célra szolgáló kulcsszavak egy halmaza, szintaxisukkal és jelentésükkel együtt.
    - **Meta-séma:** egy sémát leíró séma A képen szöveg látható

      Automatikusan generált leírásA képen szöveg látható

      Automatikusan generált leírásA képen szöveg látható

      Automatikusan generált leírás

1. **CSS**
   * **Mi a CSS?**
     + Strukturált dokumentumok megjelenítésének leírására szolgáló stíluslap nyelv.
     + Szétválasztja a dokumentum megjelenítési stílusát a dokumentum tartalmától.
     + Állomány: .css
     + IANA média típus: text/css
   * **A CSS fejlesztése, CSS szintek**
     + W3C CSS Munkacsoportja fejleszti
     + **Szintek:**
       - CSS level 1
       - CSS level 2
       - CSS level 3
     + Minden egyes szint az előzőn alapul.
     + **CSS level 1:**
       - Elavultnak tekintett.
     + **CSS level 2:**
       - A CSS 2.1 specifikáció (egyetlen dokumentum) w3c ajánlás, 2011
       - Javítása jelenleg fejlesztés alatt 2016 óta
     + **CSS level 3:**
       - Jelenleg is fejlesztés alatt áll
       - Moduláris
         * Modulokra van bontva, minden modul a CSS egy részét definiálja
         * CSS 2.1 speciális lehetőségeit bővíti, cseréli le
       - A moduloknak is szintjei vannak
         * Az 1. szintről indulnak az olyan modulok, melyeknek nincs megfelelője a CSS Level 2-ben.
         * A 3. szintről indulnak a CSS Level 2 létező lehetőségeit frissítő modulok
       - A CSS modulok eltérő stabilitási szintűek
       - Néhány modul:

Selectors Level 3

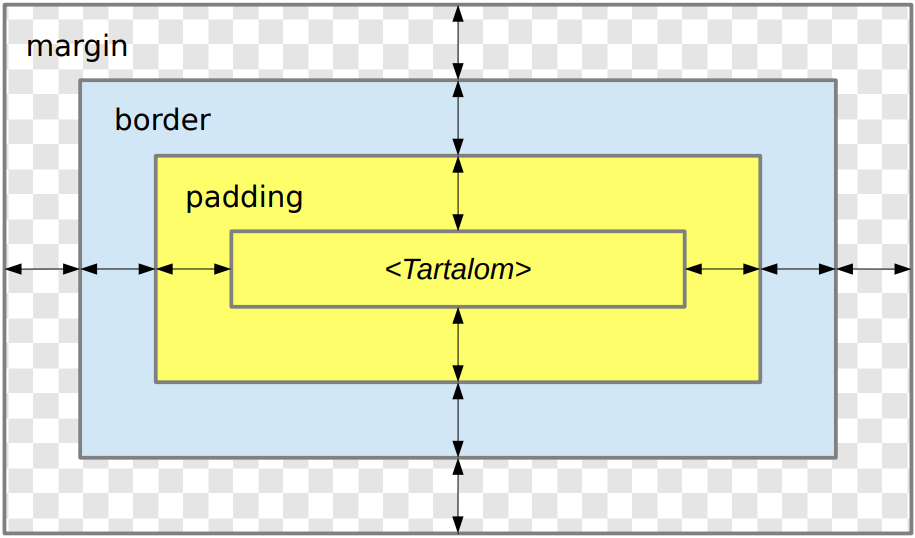
CSS Values and Units Module Level 3

CSS Transforms Module Level 1

CSS Transforms Module Level 2

CSS Snapshot 2021

A CSS jelenlegi állapotát alkotó specifikációkat gyűjti össze az implementálók számára

* + - **CSS Level 4 és azon túl**
      * Nem létezik CSS level 4
      * Önálló modulok elérhetnek 4. vagy azon túli szintre, de a CSS nyelvnek nincsenek 3. túli szintjei
  + **CSS dobozmodell**
    - A CSS egy fa-struktúrájú dokumentumot kap, melyet egy rajzvásznon jelenít meg, egy **dobozfát** (**box-tree)** előállítva, mely a megjelenített dokumentum formázási szerkezetét ábrázolja.
    - Minden egyes doboz a fában a dokumentum egy megfelelő elemét ábrázolja a térben, és/vagy időben a rajzvásznon
    - Minden elemhez nulla vagy több dobozt generál
      * *display* tulajdonsága által meghatározott módon
  + **Szintaktikai elemek:** karakterek, vezérlősorozatok, megjegyzések, deklarációs blokk, at-szabályok, stílus szabályok
    - **Karakterek:** Unicode karakterkészletet használ
      * Unicode karakterek megadásához \hhhhhh ahol hhhhhh a kódpontot ábrázoló, legalább 1, max 6 karakter hosszúságú **hexadecimális szám**
    - **Vezérlősorozatok:**
      * A speciális karakterek jelentésének elnyomásához használjuk: \
    - **Megjegyzések:**
      * Határolói: /\* és \*/
      * Tokeneken kívül bárhol megengedettek
      * Nem egymásba ágyazhatóak
    - **Deklarációs blokk**
      * **{**  és **}**  karakterek határolják, melyek között egy deklarációs lista kötelező
      * A deklarációk *tulajdonságnév:érték;*  formátumúak
      * A deklarációkat ';' karakterrel kell elválasztani.
    - **Stílus szabályok**
      * Egy kiválasztóból, vagy vesszővel elválasztott kiválasztókból, és egy azokat követő deklarációs blokkból állnak
        + pl: div { color: black; } // fekete színű karakterek lesznek a divekben
    - **At-szabályok**
      * A stíluslap feldolgozását vezérlő speciális szabályok
      * @ karakterrel kezdődnek, melyet egy azonosító követ, és ’;’ karakterrel, vagy deklarációs blokkal végződnek.
        + pl.: @charset, @import, @namespace, @media
  + **Tulajdonságok**
    - A CSS által definiált paraméterek, melye révén a dokumentumok megjelenítése vezérelhető.
    - A tulajdonságoknak neve és értéke van.
    - Összesen 548 tulajdonságnév
    - Összevont tulajdonság (shorthand property)
      * Olyan tulajdonság, mely több CSS tulajdonság értékének egyidejű beállítására szolgál.
      * Például a margin tulajdonság a margin-top, margin-bottom, margin-right, margin-left tulajdonságok értékét állítja be
  + **Kiválasztók:** típus kiválasztó, általános kiv., attribútum kiv. ([att] és [att=val]), osztály kiv, ID-kiv, pszeudo-osztályok (dinamikus, :lang(C) és szerkezeti pszeudo-osztályok), pszeudo-elemek
    - Mintaillesztésre szolgálnak, meghatározzák, hogy egy tulajdonság melyik elemekre vonatkozik
    - Kisbetű-nagybetű érzéketlenség az ASCII tartományban, kivéve ami nem a CSS hatályába esik (elemnevek, attribútumnevek és értékek)
    - **Egyszerű kiválasztó:** egyetlen típus kiválasztó, általános kiválasztó, attribútum kiválasztó, osztály kiválasztó, ID-kiválasztó vagy pszeudo-osztály
    - **Egyszerű kiválasztók sorozata:** 
      * Egymást követő egyszerű kiválasztók, melyek között nincsenek kombinátorok.
      * Típus kiválasztóval vagy általános kiválasztóval kezdődnek és nem tartalmaznak további típus kiválasztókat, vagy általános kiválasztókat.

*div*

*h2#status*

*link[rel=stylesheet][type=text/css]*

*p.copyright*

* + - **Kiválasztó:**
      * Egyszerű kiválasztók sorozatainak olyan sorozata, melyeket kombinátorok választanak el.
      * Pszeudo-elem az utolsó egyszerű kiválasztó sorozat végéhez adható, és max 1.

*h1 ~ table*

*div#main > h1*

*div > h1 + p*

*q::before*

*a img*

* + - **Típus kiválasztó:**
      * Egy CSS minősített név, a gyakorlatban egy azonosító.
      * A megfelelő nevű elemek illeszkednek rá. A képen szöveg látható

        Automatikusan generált leírás
    - **Általános kiválasztó**
      * A \* formájú kiválasztó
      * Minden elem illeszkedik rá.
      * Elhagyható olyan egyszerű kiválasztókból, mely további komponenseket is tartalmaz.
        + *\*.important* ugyan az mint a *.important*
      * Az elhagyás nem ajánlott a jobb olvashatóság végett.
    - **Attribútum kiválasztók:**
      * **[att]:** Az att attribútummal rendelkező elemek illeszkednek rá.
        + A képen szöveg látható

          Automatikusan generált leírás**[att = érték]:** Olyan elemek illeszkednek rá, melyek az *att* attribútumának értéke pontosan *érték.*
    - **Osztály kiválasztó:**
      * HTML dokumentumoknál:
    - **ID kiválasztó**
      * **#azonosító** formájú kiválasztó, az az elem illeszkedik rá, melynek *id* attribútuma *id=”azonosító”*
    - **Pszeudo-osztályok**
      * **:azonosító**  vagy **:azonosító(érték)** formájú kiválasztók.
      * Olyan kiválasztást tesznek lehetővé, mely a dokumentumon kívüli információkon alapul vagy nem fejezhető ki a többi egyszerű kiválasztóval.
      * Bizonyos pszeudo-osztályok egymást kölcsönösen kizáróak
      * **Dinamikus pszeudo osztályok:**

Olyan pszeudo-osztály, melyet egy elem megszerezhet vagy elveszíthet, miközben a felhasználó interakcióba lép a dokumentummal

pl.: *: hover*

* + - * + **Link pszeudo-osztályok:**

**:link:** a felhasználó által még nem meglátogatott linkek

**:visited:** a már meglátogatott linkek.

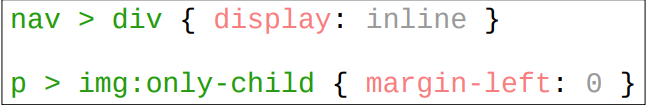
* + - * **A :lang(C)** pszeudo-osztály
        + C nyelvű szövegeket tartalmazó elemek illeszkednek rá (C egy CSS azonosító nyelvkód)
        + pl.: :lang(en) :lang(hu)
      * **Szerkezeti pszeudo-osztályok:**
        + Amikor megállapításra kerül, egy elem helye a testvéreinek listájában, akkor csak az elemeket kell a listában figyelembe venni
        + A listában az elemek számozása 1-től
        + **:root** a dokumentum gyökérelemét választja ki
        + **:only-child** olyan elemeket választ ki, melyek szülőjének nincs más elemgyermeke
        + **:only-of-type** melyek szülőjének nincs más, ugyan olyan típusú eleme
        + **:empty** A szöveget és elemeket nem tartalmazó elemek
        + **:nth-child(an+b)** ahol a és b egészek: Az olyan elemeket választja ki, melyeknek an+b-1 megelőző elemtestvére van valamely nemnegatív n-re.
        + **:nth-last-child(an+b)** ahol a és b egésze: Az olyan elemeket választja ki, melyeknek an+b-1 következő elemtestvére van valamely nemnegatív n-re.
        + **:nth-of-type(an+b)** ahol a és b egészek: Az olyan elemeket választja ki, melyeknek valamely nemnegatív n-re an+b-1 olyan megelőző elemtestvére van, melyek kifejtett neve megegyezik az elem kifejtett nevével.
        + **:nth-last-of-type(an+b)**  ahol a és b egészek: Az olyan elemeket választja ki, melyeknek valamely nemnegatív n-re an+b-1 olyan következő elemtestvére van, melyek kifejtett neve megegyezik az elem kifejtett nevével. A képen szöveg látható

          Automatikusan generált leírás
        + **:first-child** azt jelenti, mint :nth-child(1)
        + **:last-child** azt jelenti, mint :nth-lastchild(1)
        + **:first-of-type** azt jelenti, mint :nth-oftype(1)
        + **:last-of-type** azt jelenti, mint :nth-lastof-type(1)
    - **Pszeudo elemek:**
      * Lehetővé teszik a dokumentumok olyan részeinek kiválasztását, melyek más módon nem hozzáférhetők.
      * Minden kiválasztókban legfeljebb egy pszeudo-elem megengedett.
      * Az alábbi pszeudo-elemek állnak rendelkezésre:

::first-line

::first-letter

::before, ::after

* + - * A pszeudo-osztályoktól való megkülönböztetés miatt ':' helyett '::' karaktereket használ a pszeudo-elemekhez a CSS3.
  + **Kiválasztók:** kombinátorok (leszármazott kombinátor, gyermek kombinátor, szomszéd/testvér kombinátor)
    - **Kombinátor:** Az alábbi karakterek valamelyike:
      * *whitespace*
      * *>*
      * *+*
      * *~*
    - **Leszármazott kombinátor:** Egyszerű kiválasztók két sorozatát elválasztó *whitespace*.
      * Ha **P** és **Q** egyszerű kiválasztók sorozata, akkor azok az elemek illeszkednek a **„P Q”** kiválasztóra, melyek **Q**-ra illeszkednek, és **P**-re illeszkedő elemek leszármazottjai. (NEM CSAK GYERMEK, HANEM BÁRMILYEN LESZÁRMAZOTT)
      * **Például:** Ha a **thead** minden **th** elemére akarunk hivatkozni, azt úgy érjük el, hogy **thead***whitespace***th, azaz thead th {***….***}**
    - **Gyermek kombinátor:** Egyszerű kiválasztók két sorozatát elválasztó > karakter.
      * Ha **P** és **Q** egyszerű kiválasztók sorozata, akkor azok az elemek illeszkednek a **„P>Q”** kiválasztóra, melyek **Q**-ra illeszkednek, és a **P**-re illeszkedő elemek gyermekei (azaz közvetlen leszármazottjai.) 
    - **Szomszéd testvér kombinátor:** Egyszerű kiválasztók két sorozatát elválasztó + karakter.
      * Ha **P** és **Q** egyszerű kiválasztók sorozata, akkor azok az elemek illeszkednek a **„P+Q”** kiválasztóra, melyek **Q-**ra illeszkednek, és **P**-re illeszkedő elemet követ közvetlenül a dokumentumban.
      * Az illeszkedő elemeknek közös szülőjének kell lennie, tehát szomszéd is, és testvér is. 
  + **Kiválasztók:** specifikusság, specifikusság meghatározása
    - **Specifikusság:**
      * Kiválasztókhoz és deklarációkhoz specifikusság meghatározása
      * A specifikusság egy háromelemű (a, b, c) vektor, ahol abc nemnegatív egészek.
      * A vektorok rendezése lexikografikusan
      * **Az (a,b,c) vektorban:**

Az ’**a**’, a kiválasztóban előforduló **ID-kiválasztók** száma

A ’**b**’,a kiválasztóban előforduló **attribútum kiválasztók, és pszeudo osztályok** száma

A ’**c**’, a kiválasztóban előforduló **típus kiválasztók és pszeudo-elemek** száma

* + - **Deklarációk specifikussága:**
      * + Egy deklaráció annyira specifikus, mint a kiválasztó, ami tartalmazza.
        + Szabályhoz nem tartozó, azaz a HTML dokumentumban, az elem *style* attribútumához megadott deklaráció specifikussága minden kiválasztónál nagyobbA képen szöveg látható

          Automatikusan generált leírás
  + **Stíluslap eredet:**
    - Különböző eredetű stíluslapok állhatnak rendelkezésre dokumentumok megjelenítéséhez. (Fentről lefele erősödik.)
      * **felhasználói ágenstől (böngészőtől)**
      * **felhasználótól**
      * **a dokumentum szerzőjétől származó**
    - Felhasználói ágenstől származó:
      * Például olyan stílus szabály tartalmazása, mely az *em* HTML elem megjelenítéséhez kurzív betűtípust ír elő.
      * **Firefox:** az alábbi módon érjük el az alapértelmezett stíluslapot: resource://gre-resources
    - **Felhasználótól származó:**
      * A felhasználó megadhat saját stíluslapot a dokumentum megjelenítéséhez
      * Például a fogyatékkal élő felhasználók számára fontos, akadálymentesítést biztosít
    - **Szerzőtől származó stíluslap**
      * + **HTML** esetén a *link* fejléc elemmel adható meg a dokumentumhoz külső stíluslap: <link rel = „stylesheet” href = „style.css”>
      * Használhatjuk a *style* fejléc elemet is, amivel közvetlenül ágyazhatunk be szabályokat
      * **XML** esetén az xml-stylesheet feldolgozási utasítással adható meg külső stíluslap
        + A dokumentum gyökéreleme előtt kell szerepeljen
  + **A kaszkád**
    - **Kaszkádolás:** konfliktusfeloldási mechanizmus arra az esetekre, amikor egy elem/tulajdonság kombinációhoz különböző deklarációk állítanak be értéket
    - Az alábbi módon határozza meg egy elem tulajdonságának kaszkádolt értékét:
      * A vonatkozó deklarációkat eredetük szerint csökkenő sorrendbe állítja
      * Az azonos eredetű deklarációkat specifikusság szerint állítja csökkenő sorrendbe
      * A tulajdonság értékét az így kapott sor első deklarációja határozza meg.
  + **Szabályok sorrendje**
    - Akkor számíthat a sorrend, ha egy elemre egynél több azonos specifikusságú stílus szabály vonatkozik
    - Mindig az utolsó a legerősebbA képen szöveg látható

      Automatikusan generált leírás
  + **Öröklés**
    - **Öröklés, kezdőértékadás:** abban az esetben alkalmazásra kerülő mechanizmusok, amikor egy elem/tulajdonság kombinációhoz egy deklaráció sem állít be értéket.
    - Az öröklés tulajdonságértékek továbbadását jelenti. (Szülőtől gyermeknek)
    - A specifikáció minden tulajdonsághoz meghatározza, hogy öröklött-e
      * Öröklött például: *font-family, font-style, font-variant*
      * Nem öröklött: *margin*
    - Egy tulajdonság kaszkádolt értékeként az *inherit* kulcsszó öröklést kényszerít ki
  + **Kezdőértékadás**
    - Minden tulajdonságnak van egy kezdőértéke, melyet a CSS specifikációk határoznak meg. Előírható úgy, hogy a böngészőtől függjön.
      * pl.: a *background-color* kezdőértéke a *transparent*
      * A *color* tulajdonság kezdőértéke a böngészőtől függ.
      * Egy tulajdonság kaszkádolt értékeként az *initial* kulcsszó azt eredményezi, hogy a kezdőérték lesz beállítva

1. **CSS előfeldolgozók**
   * **CSS előfeldolgozó fogalma**
     + A CSS előfeldolgozók CSS-t generálnak a CSS írására szolgáló, saját szintaxisukból
     + A CSS-t új lehetőségekkel egészítik ki, pl. **változók, egymásba ágyazás, függvények, mixinek.**
       - Less
       - PostCSS
       - Sass
       - Stylus
   * **CSS előfeldolgozók közös jellemzői**
     + Egysoros megjegyzések: // kezdetű és a sor végéig tart
     + **@import:**
       - **A sima CSS** importokhoz a böngésző **oldal** **renderelésekor** kell HTTP kéréseket végrehajtania.
       - **A CSS előfeldolgozók** az importokat **fordítás során** kezelik.
   * **Sass: az SCSS szintaxis, főbb lehetőségek (változók, egymásba ágyazás, mixinek, @extend, @if, @for)**
     + **Mixinek:** A mixinek lehetővé teszik CSS deklarációk újra felhasználás céljából történő csoportosításátA képen szöveg látható

       Automatikusan generált leírás
       - Akár **argumentumokat** is kaphatnak: A képen szöveg látható

         Automatikusan generált leírás
     + **SASS:** .sass/.scss
       - **Lehetőségek:**

**Változók**

**Mixinek**

**Egymásba ágyazás**

Aritmetikai műveletek

Beépített függvények

Asszociatív tömbök

Vezérlési szerkezetek (**@if, @for,** @each, @while)

**Kiterjesztés/öröklés**

Import

* + - * **Szintaxis:**

**SCSS**: a CSS szintaxis kiterjesztése .scss

**Behúzott szintaxis (indented syntax):** A Sass eredeti szintaxisa, mely behúzást használ kapcsos zárójelek és pontosvesszők helyett. .sassA képen szöveg látható

Automatikusan generált leírás

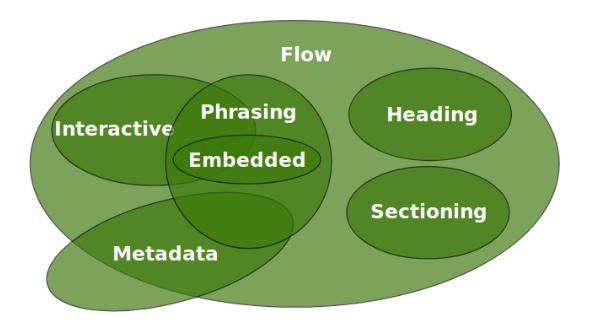
Behúzott szintaxis előnyei:

Tömörebb

Könnyen írható

Kényszerít tiszta kód írására

1. **A Web jelölőnyelvei**
   * **Mi a HTML?**
     + A HTML a Web elsődleges leírónyelve.
     + **XHTML:** Az XML alkalmazásként definiált HTML szigorúbb szabályokat ír elő a dokumentumok számára, így azok feldolgozása egyszerűbb.
     + **HTML5:** A HTML legfrissebb verziója.
       - Eredetileg a **WHATWG** fejlesztette ki a specifikációt.
       - A **W3C** 2007-ben kapcsolódott a fejlesztésébe.
   * **HTML elemek**
     + Az elemeknek, attribútumoknak és attribútumértékeknek meghatározott jelentése, **szemantikája** van.
       - pl: az *ol* elem egy rendezett listát ábrázol, a *lang* attribútum pedig a tartalom nyelvét ábrázolja.
     + A szerzők számára tilos az elemek, attribútumok és attribútumértékek megfelelő rendeltetésüktől eltérő jelentésbeli céllal történő használata.
     + A HTML előző verzióiban rendelkezésre álló prezentációs lehetőségek többsége már nem megengedett.
     + Prezentációs jelölők problémái:
       - Prezentációs elemek használata rontja a hozzáférhetőséget
       - Nehezebb karbantartás
       - Nagyobb dokumentumméret
     + Csak a *style* attribútum és a *style* elem maradt meg, mint prezentációs jelölési lehetőség.A képen asztal látható

       Automatikusan generált leírás
     + A specifikációban definiált minden egyes elemnek van egy tartalommodellje, az elem szükséges tartalmának egy leírása. Egy HTML elem tartalma ennek meg kell feleljen.
     + Minden elem 0 vagy több kategóriába sorolható az alábbiak közül
       - Idegen elemek: A MathML és SVG névterekbe tartozó elemek
       - A HTML5-ben a jobb tagoláshoz bevezetett elemek:

*article*

*aside*

*figure*

*footer*

*header*

*nav*

*section*

* + - * További új elemek:

*audio*

*canvas*

*dialog*

*meter*

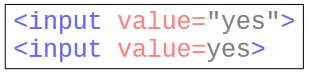
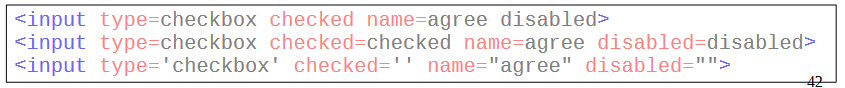
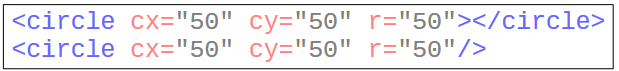
*progress*

*time*

*video*

* + **Document Object Model (DOM)**
    - Egy DOM fa egy dokumentum memóriabeli ábrázolása
    - A DOM egy alkalmazásprogramozási interfész (API), dokumentumok eléréséhez, és manipulálásához.
    - Minden ilyen dokumentumot egy fa ábrázol, mely az alábbi fajta csomópontokból áll:
      * *Document*
      * *DocumentType*
      * *DocumentFragment*
      * *Element*
      * *Text*
      * *ProcessingInstruction*
      * *Comment*
    - A képen szöveg látható

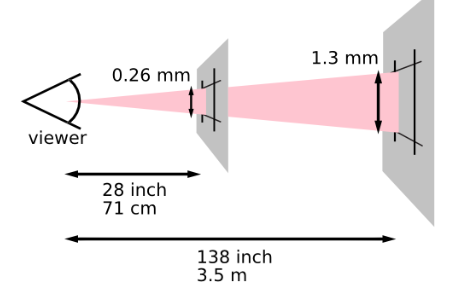
      Automatikusan generált leírásMinden egyes csomópontot egy API-val rendelkező objektum ábrázol, így manipulálható.
    - A DOM interfészek a Web IDL-ben kerülnek leírásra.
      * A Web IDL egy interfészleíró nyelv, mely a böngészőkben implementálható interfészek leírására szolgál.
    - A HTML specifikáció a HTML elemek ábrázolásához a DOM interfészeket kiterjesztő, további interfészeket határoz meg.
    - A DOM nem csupán egy API, a HTML implementációk megfelelési kritériuma is DOM műveletekkel vannak meghatározva. A specifikációk nagyrészt DOM fákkal vannak megfogalmazva.
    - Egy DOM fa szkriptekből manipulálható az oldalon. A képen asztal látható

      Automatikusan generált leírás
  + **HTML szintaxisok: HTML, XML szintaxis**
    - **HTML szintaxis:**
      * Bár nagyon hasonlít az SGML-hez és az XML-hez, egy külön nyelv saját feldolgozási szabályokkal.
      * Kompatibilis a legtöbb ősi böngészővel.
      * Fájlkiterjesztés: .html, .htm
      * Média típus: text/html
    - **XML szintaxis:**
      * Az XML 1.0 és a Namespaces in XML 1.0 szabványokon alapuló szintaxis.
      * Nem határoz meg az további szintaktikai követelményeket az XML-hez előírtakon túl.
      * XHTML szintaxisnak is nevezik.
      * Fájlkiterjesztés: .xhtml, .xht
      * Média típus: application/xhtml+xml
    - Kötelező a dokumentumtípus-deklaráció
    - Speciális karakterek: Elem szövege **nem tartalmazhat <,** vagy félreérthető **&** karaktert.
    - Félreérthető & karakter:
      * Olyan & karakter, melyet egy vagy több ASCII alfanumerikus karakter és egy ; karakter követ, melyek nem felelnek meg a szabvány által nevesített karakterhivatkozások egyikének sem. (Tehát az Unicode karakterhivatkozásokkal össze lehet keverni, de nem illeszkedik egyik karakterhivatkozásra sem.)
    - Az elem és attribútumnevek nem érzékenyek betűméretre.
    - Nem idézett attribútumérték szintaxis: 
    - Logikai attribútumok:
      * Egy logikai attribútum jelenléte egy elemen az igaz értéket ábrázolja, hiánya pedig a hamis értéket.
      * Ha megjelenik az attribútum, akkor értéke az üres karakterláncnak kell lenni, vagy egy olyan érték, mely kisbetű-nagybetű érzéketlenül megegyezik az attribútum nevével
    - Üres (void) elemek: Csak nyitó címkéjük van, tilos zárócímkét adni.
      * + pl.: *br, img, input, link, meta*
    - Az idegen elemeknek vagy egy nyitó és egy záró címkéjük van, vagy egy önlezáró címkéjük.
    - Opcionális címkék:
      * + Bizonyos elemek nyitó és zárócímkéi elhagyhatóak.
        + Egy elem nyitó címkéjének elhagyása ebben az esetben nem azt jelenti, hogy az elem nincs ott.

pl. egy HTML dokumentumnak mindig van egy gyökéreleme, még akkor is, ha a <html> karakterlánc egyáltalán nem jelenik meg benne.

* + - * + A képen szöveg látható

          Automatikusan generált leírásElhagyható egy *li* elem záró címkéje, ha az elemet közvetlenül egy másik *li* elem követi, vagy nincs több tartalom a szülő elemben
      * Nem támogatottak a névtér-deklarációk, még idegen elemekhez sem.
  + **A HTML dokumentumtípus-deklaráció**
    - A HTML szintaxisban a <!DOCTYPE html> dokumentumtípus-deklaráció szükséges, melynek célja csak annyi, hogy a dokumentum megjelenítése szabványos módban történjen.
      * A fenti rövid dokumentumtípus-deklaráció előállítására nem képes szoftverek használhatják helyette a <!DOCTYPE html SYSTEM "about:legacy-compat"> dokumentumtípus-deklarációt.
    - Az XML szintaxisban tetszőleges dokumentumtípus-deklaráció használható, nem is kötelező megadni.

1. **Responsive webdesign**
   * **Mi a reszponzív webdesign?**
     + A reszponsive webdesign (RWD) egy webfejlesztési megközelítés olyan weblapok készítéséhez, melyek optimálisan néznek ki, és viselkednek egy egész sor eszközön, a mobiloktól kezdve az asztali monitorokig, függetlenül a képernyőmérettől és tájolástól.
     + A responsive weboldalak dinamikusan ahhoz az eszközhöz igazodnak, melyen megjelennek.
     + A kifejezést Ethan Marcotte találta ki 2010-ben.
   * **Weboldal elrendezések: statikus/rögzített, folyékony, adaptív, reszponzív**
     + **Statikus:** Abszolút mértékegységekben (többnyire pixelben) kifejezett rögzített szélességet használó elrendezés.
     + **Fluid:** relatív mértékegységekben kifejezett szélességet használó elrendezés
     + **Adaptív:** statikus elrendezések egy sorozataként tekinthető.
       - Média lekérdezések révén különböző statikus elrendezéseket határoz meg különböző nézetablak szélességekhez.
     + **Reszponzív:** folyékony elrendezések egy sorozataként tekinthető
       - Média lekérdezések révén különböző folyékony elrendezéseket határoz meg különböző nézetablak szélességekhez.
   * **Referencia pixel, CSS pixelek és fizikai pixelek**
     + **Referencia pixel:**
       - * A referencia pixel az a látószög, mely alatt egy pixel látszik egy 96dpi pixelsűrűségű eszközön egy kartávolságról nézve.
         * 28 inch névleges karhossz esetén a látószög nagyjából 0,0213 fok
         * Egy karhossz távolságról nézve 1px nagyjából 0,26mm-nek felel meg
       - A *window* interfész *devicePixelRatio* attribútuma a CSS pixel méretének az eszközpixel méretéhez viszonyított arányát adja vissza az aktuális megjelenítő eszközhöz.
   * **A CSS abszolút és relatív hosszúság mértékegységei, nézetablak mértékegységek**
     + A képen asztal látható

       Automatikusan generált leírás**Abszolút hosszúság mértékegységei:**
       - Az abszolút hossz mértékegységek kötése az alábbiak szerint történik:

A fizikai mértékegységek a megfelelő fizikai mérésekhez való kötésével

a px mértékegység a referencia pixelhez való kötésével

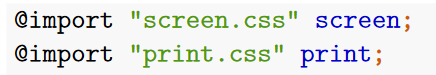
* + - * A px mértékegységet **látószög mértékegységnek** is nevezik. (**visual angle unit**)
    - **CSS relatív hosszúság mértékegységei:**
      * **em:** a *font-size* tulajdonság az elemhez számított értékét jelenti.
      * **rem:**  az *em* egység gyökérelemhez tartozó számított értékét jelenti.
    - **A CSS nézetablak mértékegységei:**
      * **vw:** a nézetablak szélességének 1%-ával egyenlő.
      * **vh:** a nézetablak magasságának 1%-ával egyenlő.
      * **vmin:** a vw és vh közül a kisebbel egyenlő.
      * **vmax:** a vw és vh közül a nagyobbal egyenlő.
  + **Média lekérdezések, CSS-ben és HTML-ben, szintaxis, kiértékelés, média-típusok, média jellemzők (csak *width* és *height* és *orientation*)*, média jellemzők kombinálása, (not, and, or)*, a min- és max- előtagok (min/max width, min/max height)**
    - **Responsive webdesign fő alkotó elemei:**
      * Média lekérdezések
      * Nézetablak
      * Folyékony rácsok
      * Rugalmas helyettesített elemek
      * Adaptív/reszponzív képek
    - **Média lekérdezések:**
      * Egy média lekérdezés egy módszer a felhasználói ágens vagy eszköz bizonyos jellemzőinek vizsgálatára, melyen a dokumentum megjelenítésre kerül.
      * Stíluslapok feltételes alkalmazására szolgálnak.
      * A média lekérdezések majdnem mindig függetlenek a megjelenített dokumentumtól
      * Csak külső információktól függenek, hacsak nem írja elő egy másik lehetőség azt, hogy hatással van a feloldásukra
      * Alábbi nyelvekben használhatók:

CSS

XML

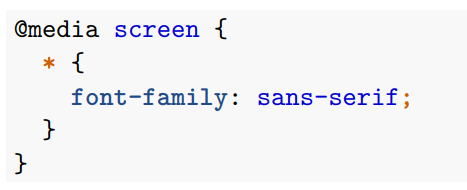
HTML

JavaScript

* + - **Média lekérdezések CSS-ben:**
      * **@import**

Az opcionális média lekérdezés lista a feltételeket fejezi ki, melyek teljesülése esetén alkalmazható az @import

Ha a média lekérdezés nem illeszkedik, akkor az importált stíluslap szabályai nem lesznek alkalmazva.

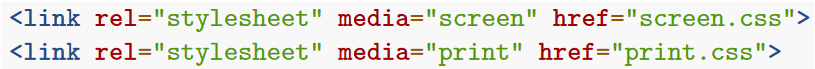
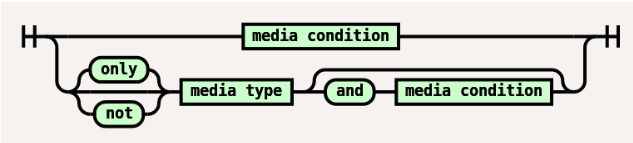
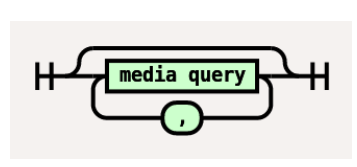
* + - * **@media**

Egy olyan feltételes csoportszabály, melynek feltétele egy média lekérdezés.

A feltételes csoportszabályok egy feltételt társítanak más CSS szabályok csoportjához

Minden egyes feltételes csoportszabálynak van egy feltétele.

Amikor a feltétel igaz, a CSS feldolgozó a csoportszabályon belüli szabályok alapján jeleníti meg a dokumentumot. Amikor a feltétel hamis, a CSS feldolgozók számára tilos a csoportszabályon belüli szabályok alkalmazása.

* + - **Média lekérdezések HTML-ben:**
      * *media* attribútum a következő elemeken adható meg:
        + *link*
        + *meta*
        + *source*
        + *style*
    - **Média lekérdezések szintaxis:**
      * Egy média lekérdezés egy opcionális módosítóból, egy opcionális média típusból, és egy média feltételből áll:
      * Több média lekérdezés egy vesszővel elválasztott média lekérdezés listává kombinálható
    - **Kiértékelés:**
      * Egy média lekérdezés egy logikai kifejezés, igaz vagy hamis.
      * Igaz, ha:

meg van adva a média típus, akkor az illeszkedik annak az eszköznek a média típusára, ahol a felhasználói ágens fut.

* + - * Egy média lekérdezés lista akkor igaz, ha bármely komponense igaz, és csak akkor hamis, ha minden komponens hamis.
      * A felhasználói környezet változásaira válaszul a felhasználói ágensnek újra ki kell értékelniük a média lekérdezéseket.
    - **Média típusok:**
      * Egy média típus felhasználói ágens eszközök egy széles csoportja, melyeken megjeleníthető egy dokumentum
      * Nem összetévesztendő az IANA média típusokkal!!!
      * A következő média típusok definiáltak a média lekérdezésekben való használatra:

*all:* az összes eszköz

*print:* a nyomtató eszközökre illeszkedik, és az olyan eszközökre, melyek célja egy nyomtatott megjelenítés visszaadása, pl. print review.

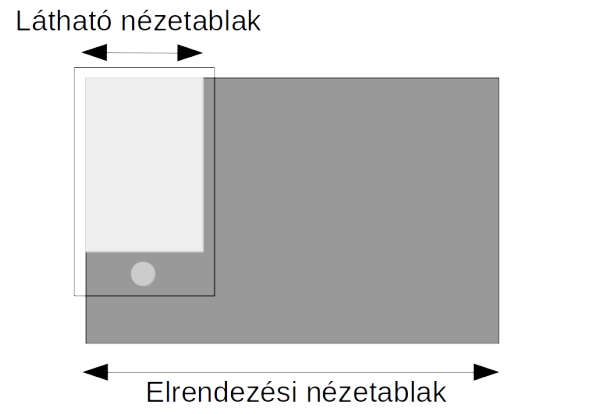
*screen:* *print-re* vagy *speech-re* nem illeszkedő összes eszköz.

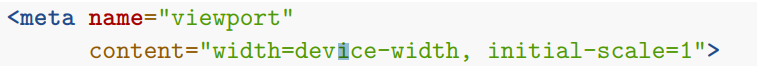
*speech*: a képernyőolvasókra és hasonló eszközökre illeszkedik.

* + - **Média jellemzők (width, height, orientation):**
      * Egy média jellemző egy, a média típusoknál finomabb teszt, mely a felhasználó ágens vagy megjelenítő eszköz egy bizonyos jellemzőjét vizsgálja.
      * Szintaktikailag CSS tulajdonságokra hasonlítanak
      * **width: <length>**
      * **height: <length>**
      * **orientation: portrait | landscape**
    - **A min- és max- előtagok használata tartomány típusú jellemzőkön (min/max width, min/max height)**
      * Egy tartomány típusú média jellemző a tartomány kontextusban történő kiértékelés helyett írható közönséges média jellemzőként, de a neve előtt a min- vagy max- előtaggal.
      * **A min- előtag** használata egy jellemző néven ekvivalens a >= operátor használatával: (min-height: 600px) ugyanaz, mint (height >= 600px).
      * **A max- előtag** használata egy jellemző néven ekvivalens a <= operátor használatával: (max-width: 40em) ugyanaz, mint (width <= 40em).
      * Diszkrét típusú média jellemzőkhöz nem adható meg min- vagy max- előtag.
    - **Média jellemzők kombinálása (not, and, or)**
      * Több média jellemező egy média feltétellé kombinálható össze a not/and/or logikai műveletekkel.
      * Bármely média jellemző negálható elé not helyezésével: not(color).
      * Két vagy több média jellemző összefűzhető közéjük and/or helyezésével: (width < 600px) and (height < 600px), update: slow) or (hover: none)
      * Média feltételek csoportosíthatók zárójelek közé zárással, mint például not((color) or (hover)).
      * Nem megengedett egy média lekérdezés ugyanazon “szintjén” keverni az and/or/not logikai műveleteket: (color) and (pointer) or (hover) érvénytelen.
  + **A látható és elrendezési nézetablak, a *viewport meta* címke.**
    - **Nézetablak**
      * A mobil/kézi eszközök böngészőinek nézetablaka ált. sokkal keskenyebb egy, a CSS által ajánlott CSS pixelméretet adó nagyítási szinten, mint egy böngészőablaké.
      * A keskeny nézetablak gondot jelent olyan dokumentumoknál, melyeket asztali böngészőre terveztek.
      * Mobil böngészők két nézetablaka:

**Látható nézetablak:** az oldal a kijelzőn látható része

A képen szöveg látható

Automatikusan generált leírás**Elrendezési nézetablak:** a nézetablak, melybe a böngésző kirajzol egy oldalt

* + - * **A *viewport meta* címke:** Abból a célból vezették be, hogy lehetővé tegye a szerző számára a nézetablak és a kezdeti nagyítási arány közvetlen megadását

1. **A HTTP haladó lehetőségei**
   * **Űrlap adatok továbbítása a *GET* és *POST* metódusokkal, különbségeik**
     + **Biztonságos metódusok:**
       - Ha alkalmazása a cél erőforrásra várhatólag nem eredményez semmiféle állapotváltozást az eredet szerveren
         * Biztonságos: GET, HEAD, OPTIONS, TRACE
         * Nem biztonságos: CONNECT, DELETE, POST, PUT
       - A biztonságos metódusok információ lekérésére szolgálnak.
       - A gyakorlatban egy biztonságos metódusnak is lehet mellékhatása a szerveren
   * **Sütik: definíció, felhasználások, *Set-Cookie, Cookie* fejlécmezők, süti attribútumok (*expires, max-age, domain, path, secure, HttpOnly*)*,* sütik kezelése, harmadik féltől származó sütik**
     + **Süti-cookie:**
       - Egy név-érték pár és kapcsolódó metaadatok, melyeket egy eredet szerver egy válasz Set-Cookie fejlécmezőjében küld a felhasználói ágensnek
         * Az attribútumok segítségével az eredet szerver egy hatáskört határozhat meg a sütihez.
       - A felhasználói ágens a további kérésekben a név-érték párt a Cookie fejlécmezőben küldi vissza az eredet szervernek.
       - Az eredet szerver akár több sütit is küldhet egy válaszban.
     + **Felhasználások:**
       - Munkamenet kezelés
       - Testreszabás
       - Felhasználó követés
     + **Set-cookie**
       - Amikor a felhasználói ágens egy *Set-Cookie* fejlécmezőt kap, eltárolja az attribútumaival együtt
       - A továbbiakban, amikor a felhasználói ágens egy HTTP kérést hajt végre, a Cookie fejlécmezőbe illeszti az alkalmazható, nem lejárt sütiket.
         * Csak a név-érték párokat, nem attribútumokat.
       - Ha a felhasználói ágens egy olyan új sütit kap, melynek neve, valamint *Domain és Path* attribútuma megegyezik egy már tárolt sütiével, akkor az új sütire cseréli le a régit.
     + **Süti attribútumok:**
       - ***expires:*** a süti lejáratának dátumát és idejét adja meg
       - ***max-age:*** azt adja meg, hogy hány másodperc múlva jár le a süti.
       - ***domain:***

Meghatározza, hogy a süti mely szervereknek lesz elküldve

Ha nincs, akkor csak az eredet szervernek küldi vissza a sütit.

* + - * ***path:***

A süti hatáskörét az adott útvonalakra korlátozza

Ha nincs, akkor a kért URI útvonal komponensének könyvtárát használja alapértelmezett értékként

* + - * ***secure:***

A süti hatáskörének biztonságos csatornákra korlátozása

Egy Secure attribútummal rendelkező sütit a felhasználói ágens csak akkor tesz bele egy kérésbe, ha annak átvitele biztonságos csatornán történik

* + - * ***HttpOnly:***

HTTP kérésekre korlátozza a süti hatáskörét

Arra utasítja a böngészőt, hogy ne tegye a sütit elérhetővé kliens oldali API-k (pl. JavaScript) számára

* + - **Sütik kezelése**
      * **A böngészőknek:**

Törölni kell a lejárt sütiket

Az aktuális munkamenet végén törölniük kell az összes nem perzisztens sütit.

Ajánlott a felhasználók számára lehetővé tenni a tárolt sütik kezelését

Ajánlott a felhasználók számára lehetővé tenni a sütik tiltását

Korlátot szabhatnak a tárolható sütik számára és méretére

Specifikáció ajánlása (mit tudjanak a böngészők):

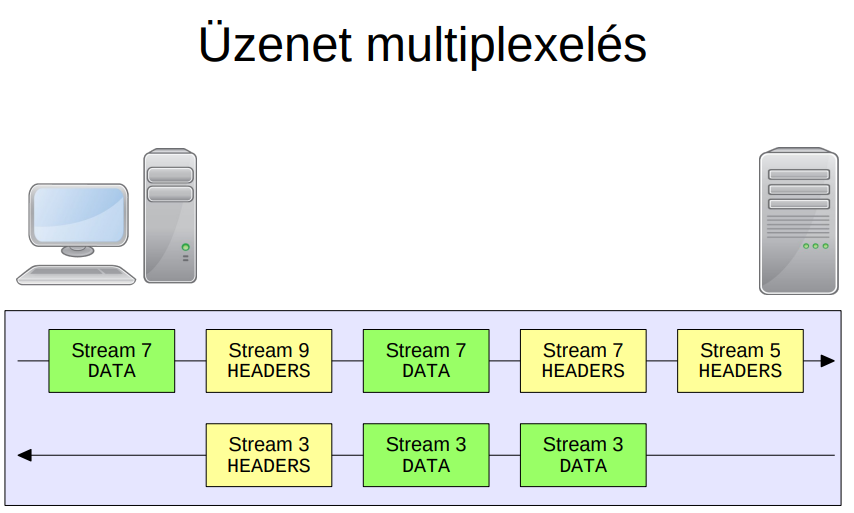
Sütinként legalább 4096 byte

Tartományonként legalább 50 süti

Összesen legalább 50 süti

* + - **Harmadik féltől származó sütik:**
      * Az úgynevezett third-party cookies különösen problémásak
        + Egy HTML oldal megjelenítése során egy felhasználói ágens gyakran kér le erőforrásokat más szerverekről
        + Ezek a harmadik félnek számító szerverek sütiket használhatnak a felhasználó követésére még akkor is, ha a felhasználó közvetlenül soha nem látogatja meg őket.
  + **Felhasználó követés: definíció, az alapjául szolgáló információk, a *Referer* fejlécmező, védekezés a követés ellen**
    - **Felhasználó követés:**
      * Egy adott felhasználó több különböző kontextuson keresztüli tevékenységével kapcsolatos adatgyűjtés, és az ebből a tevékenységből származtatott adatok megőrzése, felhasználása vagy megosztása azon a kontextuson kívül, amelyben ez a tevékenység megtörtént.
      * Egy kontextus olyan erőforrások egy összessége, melyek ugyanazon fél ellenőrzése vagy több fél közös ellenőrzése alatt állnak
      * **Alábbiakon alapulhat:**
        + IP-cím
        + sütik
        + Az ETag fejlécmező
        + Eszköz ujjlenyomat
    - **Referer fejlécmező**
      * Lehetővé teszi a felhasználói ágens számára, hogy megadja azt az erőforrást azonosító URI hivatkozást, melyből a cél URI származik (tehát melyik URI oldalán találta az URI-t amire átjött, tehát ki ajánlotta, ki referálta)
      * Potenciálisan információkat nyújthat a felhasználó böngészési előzményeiről, ami biztonsági kockázat.
      * Nem ajánlott a Referer fejlécmező küldése akkor, ha a hivatkozó erőforrás elérése biztonságos protokollon keresztül történt és a kérés céljának eredete eltér a hivatkozó erőforrásétól, hacsak nem engedi meg kifejezetten a Referer mező küldését a hivatkozó erőforrás
      * Tilos a Referer fejlécmező küldése nem biztonságos HTTP kérésben, ha a mezőértékben jelzett oldal biztonságos protokollon érkezett.
    - **Védekezés a követés ellen:**
      * A Referer fejlécmező küldésének tiltása
      * Harmadik féltől származó sütik tiltása
      * Privát böngészés – inkognító mód
      * Beépített követés elleni védelem

1. **HTTP/2**
   * **Bevált gyakorlatok a késleltetési idő csökkentésére: beágyazás, spriting, sharding, összefűzés, kicsinyítés**
     + **Beágyazás – inlining:**A képen szöveg látható

       Automatikusan generált leírásképek közvetlen beágyazása CSS stíluslapokba
     + **Spriting:** Több kép kombinálása egyetlen képállományban
     + **Sharding:** A tartalom elosztása több szerveren
       - Érdemes a képeket olyan webszerverről szolgáltatni, amely nem használ sütiket
     + **Összefűzés:** Több CSS stíluslap és JavaScript állomány összefűzése
     + **Kicsinyítés:** felesleges karakterek eltávolítása CSS stíluslapokból és JavaScript állományokból anélkül, hogy az erőforrás a böngésző általi feldolgozása módosulna
   * **Mi a HTTP/2**
     + A HTTP szemantikájának egy optimalizált kifejezésmódja.
       - Cél a hálózati erőforrások hatékonyabb használata, és a késleltetés csökkentése.
       - Fő cél, hogy a klienseknek csak egy kapcsolatot kelljen fenntartani egy szerverhez
       - Ugyanazokat a metódusokat, állapotkódokat, fejlécmezőket és URI sémákat használja, mint a HTTP/1.1. (Az üzenetek formálása és átvitele történik eltérő módon.)
   * **Üzenet multiplexelés, keretek**
     + **Multiplexelés:** több adatfolyam egyetlen fizikai csatornán való küldése. Lehetővé teszi, hogy több külön HTTP kérés/válasz párost küldjünk egyszerre, darabokba bontva, felváltva, egy csatornán.
       - **Kis magyarázat:** A HTTP/2-ben, az HTTP multiplexelést a HPACK használatával oldják meg, ami lehetővé teszi, hogy a kliens és a szerver meghatározzák, hogy melyik kérés/válasz pár tartozik egymáshoz.
     + **Keretek:**
       - Minden keret egy 9 byte méretű fejléccel kezdődik, melyet egy változatos hosszúságú adatrész (**payload)** követ.
       - Keret fejléc mezői:

**Hossz:** payload hosszát adó, 24 bit-es egész

**Típus:** a keret 8 bit-en ábrázolt típusa

**Jelzők:** 8 bit-en ábrázolt logikai jelzők, jelentése a keret típusától függ.

**R:** fenntartott célú nulla értékű bit. Jelenleg nincs definiált jelentése.

**Adatfolyam azonosító:** egy adatfolyamot azonosító, 31 bit-es egész.

* + **Adatfolyamok, jellemzőik, azonosításuk**
    - **Adatfolyamok jellemzői:** 
      * Egyetlen HTTP/2 kapcsolat több egyidejűleg nyitott adatfolyamot tartalmazhat (multiplexelés)
      * Egyoldalúan, egyeztetés nélkül nyithat adatfolyamot a kliens vagy a szerver, de megosztva használják.
      * Bármelyik végpont lezárhatja
      * Keretek küldésének sorrendje lényeges
      * A fogadó abban a sorrendben dolgozza fel a kéréseket, ahogy kapta
      * Az adatfolyamokat egy egész szám azonosítja
    - **Adatfolyamok azonosítása:**
      * 31 bit-es előjel nélküli egész, a létrehozó rendeli az adatfolyamhoz.
      * **kliens által nyitott:** páratlan számok
      * **szerver által nyitott:** páros számok
      * **0x00 azonosító:** vezérlő üzenetekhez használt
      * Az új adatfolyamok azonosítójának mindig nagyobbnak kell lenni, mint a létrehozó fél által létrehozott bármely másik azonosító
      * Az azonosítók nem újra használhatóak
      * Ha kimerül a tartomány, akkor új kapcsolatot kell nyitni.
  + **HTTP/2 kérés/válasz váltás**
    - Egy kliens minden egyes HTTP kérést egy új adatfolyamon küld el a szervernek, melyhez egy előzőleg nem használt adatfolyam azonosítót használ.
    - A szerver ugyanezen az adatfolyamon küldi vissza a választ.
    - A kérés/válasz váltás során az adatfolyam teljesen „elhasználódik”.
    - A választ záró keret lezárja az adatfolyamot.
  + **Fejlécmezők, pszeudo-fejlécmezők**
    - A fejlécmezők nevét kisbetűssé kell alakítani a HTTP/2 szerinti kódolásuk előtt.
    - **Pszeudo-fejlécmezők:**
      * A HTTP/1.x üzenetek kezdősorában adott információk ábrázolásához a HTTP/2 ’:’ karakterrel kezdődő, speciális pszeudo-fejlécmezőket használ.
      * A pszeudo-fejlécmezők nem HTTP fejlécmezők.
      * **Kérés pszeudo-fejlécmezők:**

**:method**: a HTTP metódust tartalmazza.

**:scheme**: a cél URI séma részét tartalmazza.

**:authority:** a cél URI autoritás részét tartalmazza, a *host* fejlécmező megfelelője.

**:path:** a cél URI útvonal és lekérdezés részét tartalmazza.

* + - * **Válasz pszeudo-fejlcmezők:**

**:status** a HTTP állapotkódot hordozza. Minden válasznak tartalmaznia kell.

* + - * A pszeudo-fejlécmezőnek a rendes fejlécmező előtt kell állnia.
      * *CONNECT* kérések kivételével minden HTTP/2 kérésnek pontosan 1 érvényes értéket kell tartalmaznia a *:method, :scheme, :path* mezőkhöz.

1. **HTTP/3**
   * **Mi a HTTP/3**
     + A HTTP legfrissebb fő verziója, mely a QUIC átviteli protokollon alapul.
     + HTTP szemantika leképezése a QUIC átviteli protokollra.
   * **Mi a QUIC**
     + Az UDP-re épülő biztonságos, általános célú átviteli protokoll.
     + A HTTP/3-at, és a QUIC-et az IETF QUIC munkacsoportja fejleszti.
   * **A HTTP/2 és HTTP/3 összehasonlítása**
     + **Hasonlóságok:**
       - A két protokoll gyakorlatilag ugyan azokat a lehetőségeket biztosítja a kliens számára.

Adatfolyamok

Szerver push

Mező tömörítés

* + - **Eltérések:**
      * A HTTP/2 elvileg implementálható TLS nélkül, a HTTP/3 nem.
      * A HTTP/3 több adatfolyam használatát teszi lehetővé, mint a HTTP/2 (262-1, a 231-1 helyett)
      * A HTTP/2 keretek néhány fajtája nem szükséges a HTTP/3-ban, mert a QUIC nyújt ekvivalens funkcionalitást.
      * A HTTP/3 a HPACK egy módosított változatát, QPACK-et használ.

1. **Kódolvasás**
   * + **Követelmény az alábbiak olvasása és megértése:**
       - XML, DTD, JSON, JSON Schema, HTML, CSS SCSS (Sass)
       - A curl program kimenete