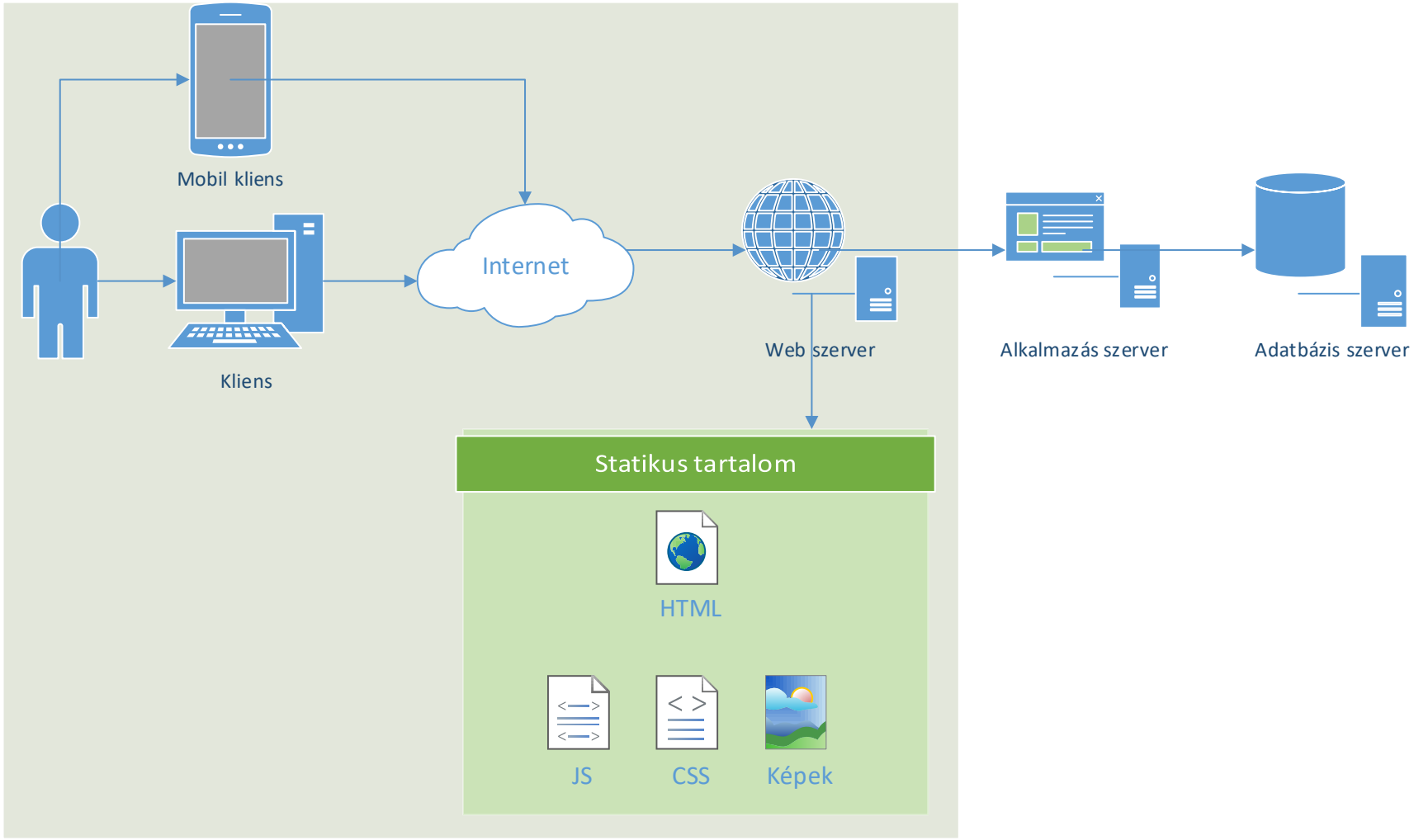
Webes architektúra áttekintése

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Webes architektúra áttekintése



Mi a web szerver?

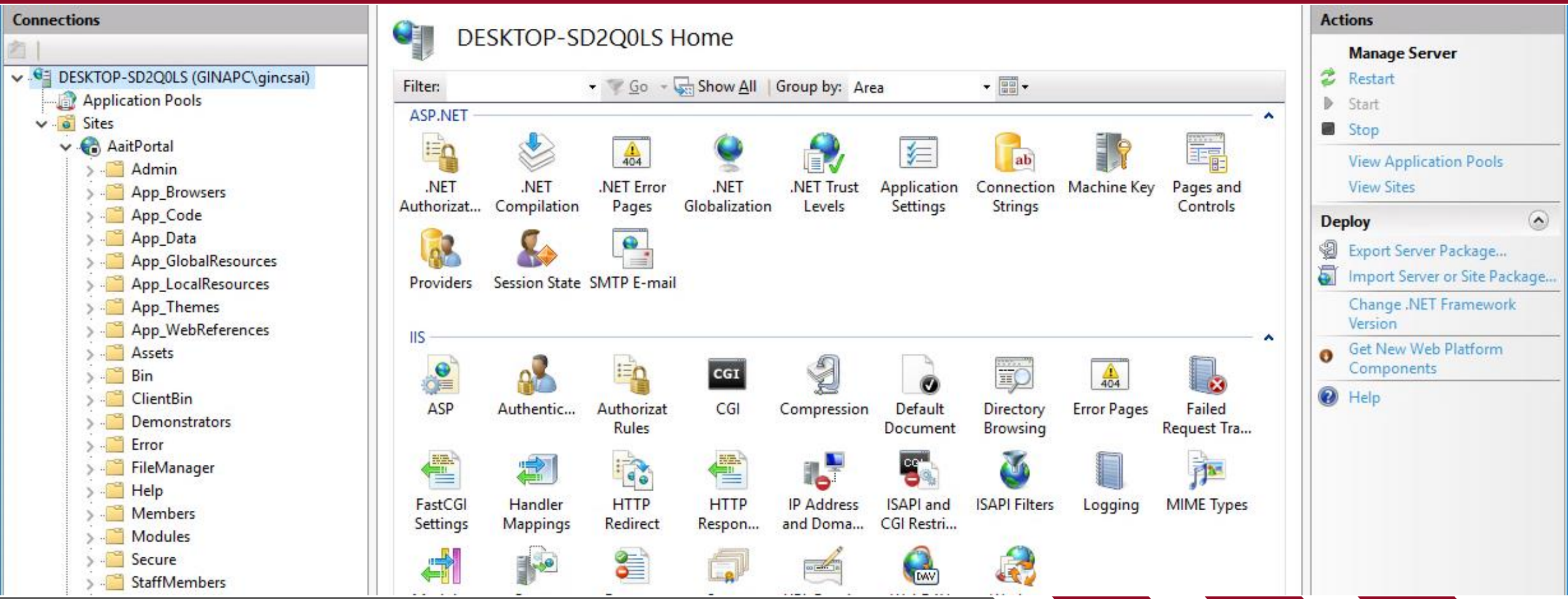
- Hardver szempontjából
  - egy internetre kapcsolt számítógép
  - ezen fut a web szerver szoftver
  - itt tároljuk a kiszolgálandó fájlokat

- Szoftver szempontból egy olyan alkalmazás, ami
  - egy adott porton figyeli a beérkező kéréseket.
  - fel tudja oldani az URL-eket és ez alapján statikus vagy dinamikus tartalmat tud szolgáltatni.
  - a beérkező HTTP kéréseket megérti és kiszolgája.
  - szabályozni tudja, hogyan és mit lehet elérni a weben keresztül.

URL és fizikai útvonal összerendelése

- Ahhoz, hogy bizonyos fájlokat elérhetővé tegyünk a weben a webszerveren létre kell hozni egy website-ot.
- A websitenak be kell állítani hogy
  - milyen fizikai útvonalon érhetők el a fájlok
  - milyen porton figyeljen a webszerver
  - kinek a nevében fusson az oldalt kiszolgáló processz
  - milyen bejelentkezés szükséges az oldal eléréséhez
  - szükséges-e titkosítás (HTTPS)
  - stb.

DEMO



BME AUT

Statikus webszerver

IIS

Kommunikáció: HTTP

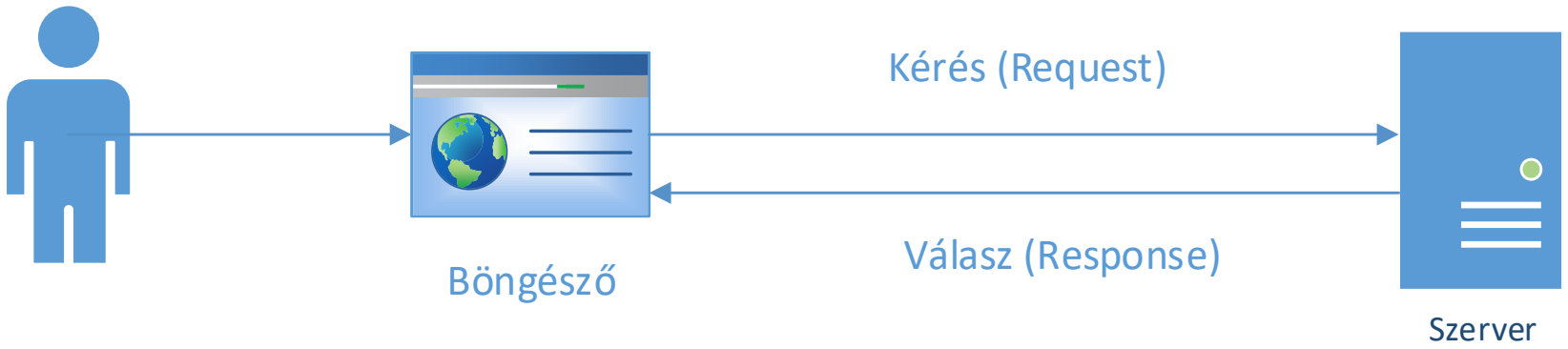
Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

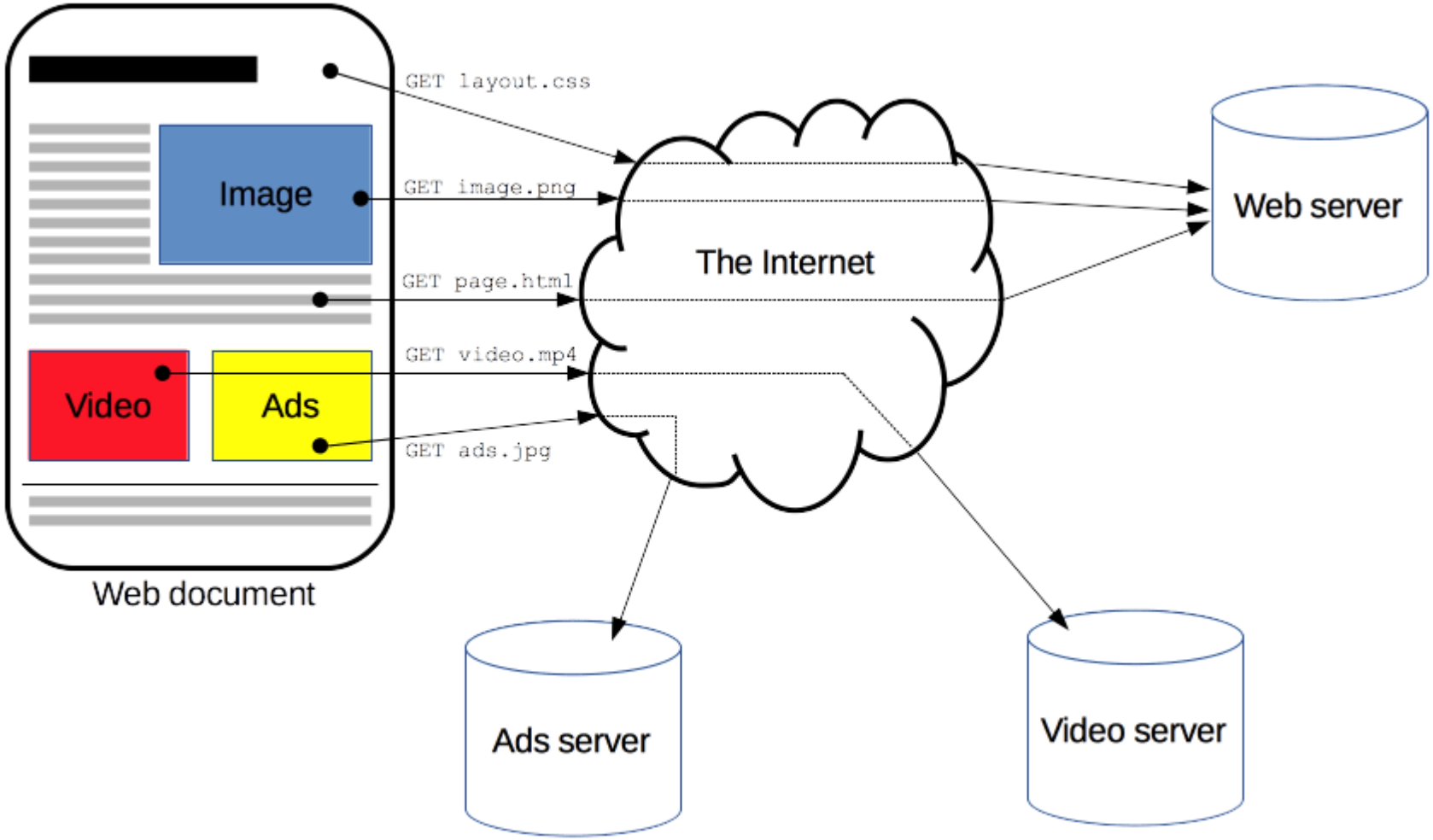
Gincsai Gábor
gincsai@aut.bme.hu

Kommunikáció: request-response

- Mindig a kliens kezdeményezi a kommunikációt, a szerver csak válaszol. Ez a pull model.
- A kérés egy adott URL-re küldött megfelelően formázott csomag.
- **User agent**: a kliens általános megnevezése, bármilyen alkalmazás, amely HTTP kérést tud küldeni.
  - pl. leggyakrabban web böngésző, mobil kliens, RSS olvasó



Weboldal letöltése több HTTP kéréssel



Hogyan válaszol a szerver?

- A beérkező kérést a webszerver feldolgozza és előállítja a szöveges HTTP válaszüzenetet.
  - **Statikus** tartalmat (fájlokat) szolgál ki jellemzően az *URL* → *fájlrendszer* megfeleltetés alapján.
  - **Dinamikus** tartalmat állít elő a kérés paraméterei és az alkalmazás állapota (memória, DB) alapján.

Statikus vs dinamikus kiszolgálás

Statikus kiszolgálás

- Egyszerű
- Olcsó
- Hatékony

A tartalom csak a szerveren található fájlok manipulációjával frissíthető.

Dinamikus kiszolgálás

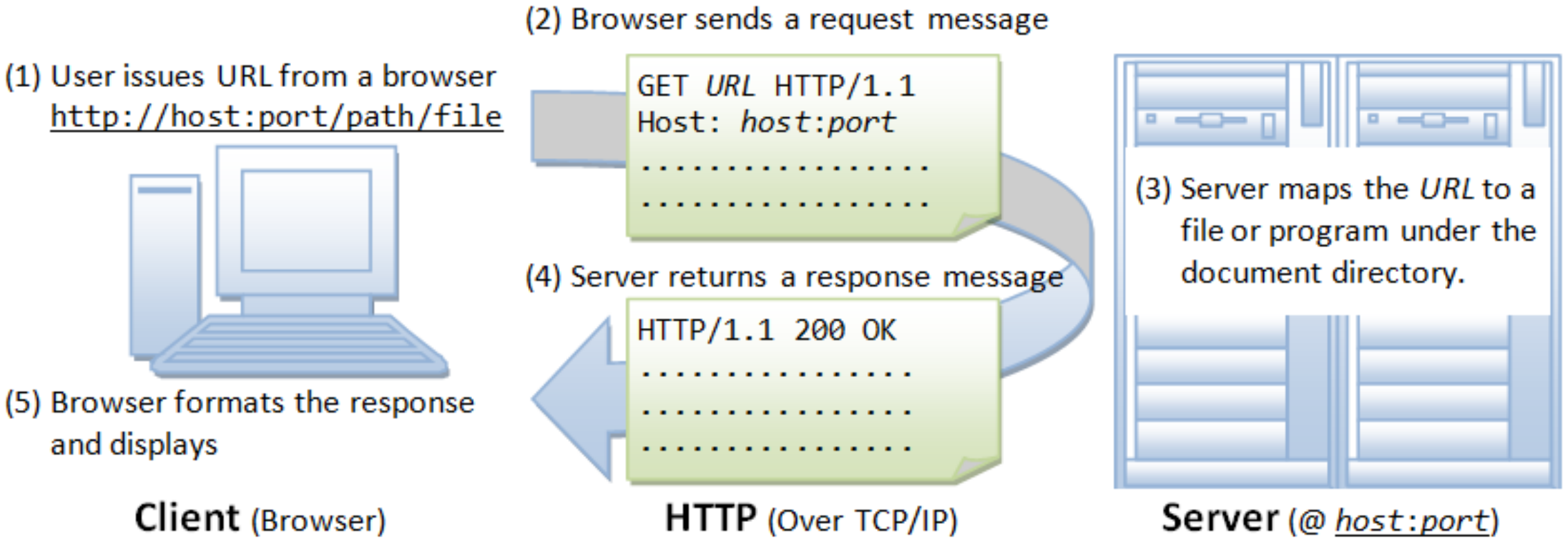
- Bonyolult
- Drága
- Lassabb

A tartalom újraindítás és telepítés nélkül frissíthető.

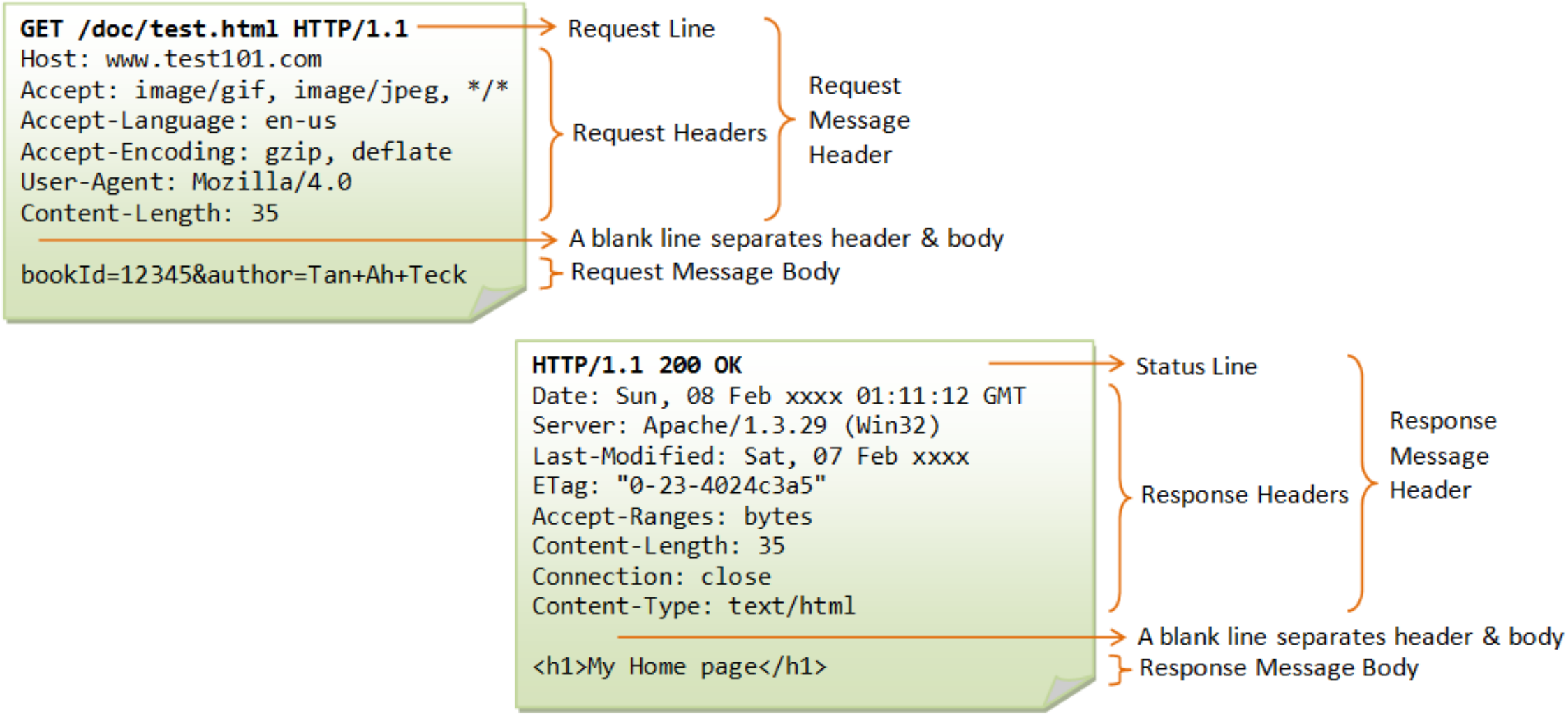
Statikus kiszolgálás

- A statikus kiszolgálás a kérések feldolgozásának egy lehetséges módja.
- Statikus kiszolgálás ≠ statikus weboldal!
  - Statikus JS kódból módosítjuk a tartalmat.
- Statikus weboldal = csak statikus kiszolgálás
  - Egyszerű HTML fájlok letöltése
- Dinamikus weboldal ≠ csak dinamikus kiszolgálás
  - Single page application: statikus HTML és JS fájlokat kell kiszolgálni, amik egy API-ról töltik le az adatokat.

HTTP kérés-válasz



Kérés és a válasz felépítése

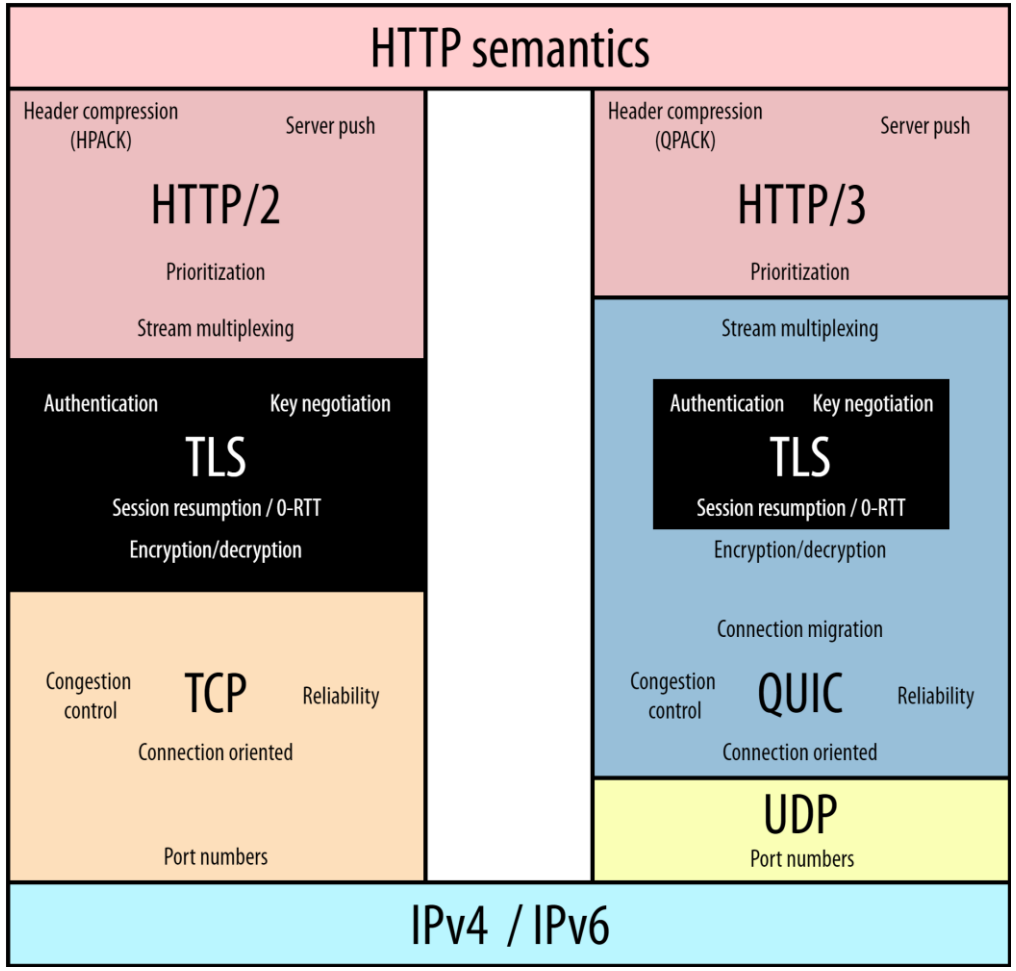


A HTTP állapotmentes

- Az egyes HTTP kérések között nincs állapotmegőrzés.
  - Ez probléma lehet olyan esetekben, amikor például egy webshopban tudni szeretnénk, hogy ki a belépett felhasználó aki a kosárba pakol.
- A HTTP protokoll a fejlécek segítségével testreszabható és például a cookie segítségével meg lehet őrizni az állapotot a kérések között.
  - **Session** (munkamenet): egy felhasználó első és utolsó kérése között lezajló kérés-válasz tranzakciók.
    - Időkorlátos, például 20 perces csúszó ablak (sliding timeout).
- Állapotmegőrzésre használható továbbá
  - rejtett mező,
  - URL parameter
  - HTTP fejléc. Pl: Authorization, amiben a Bearer token is utazik.

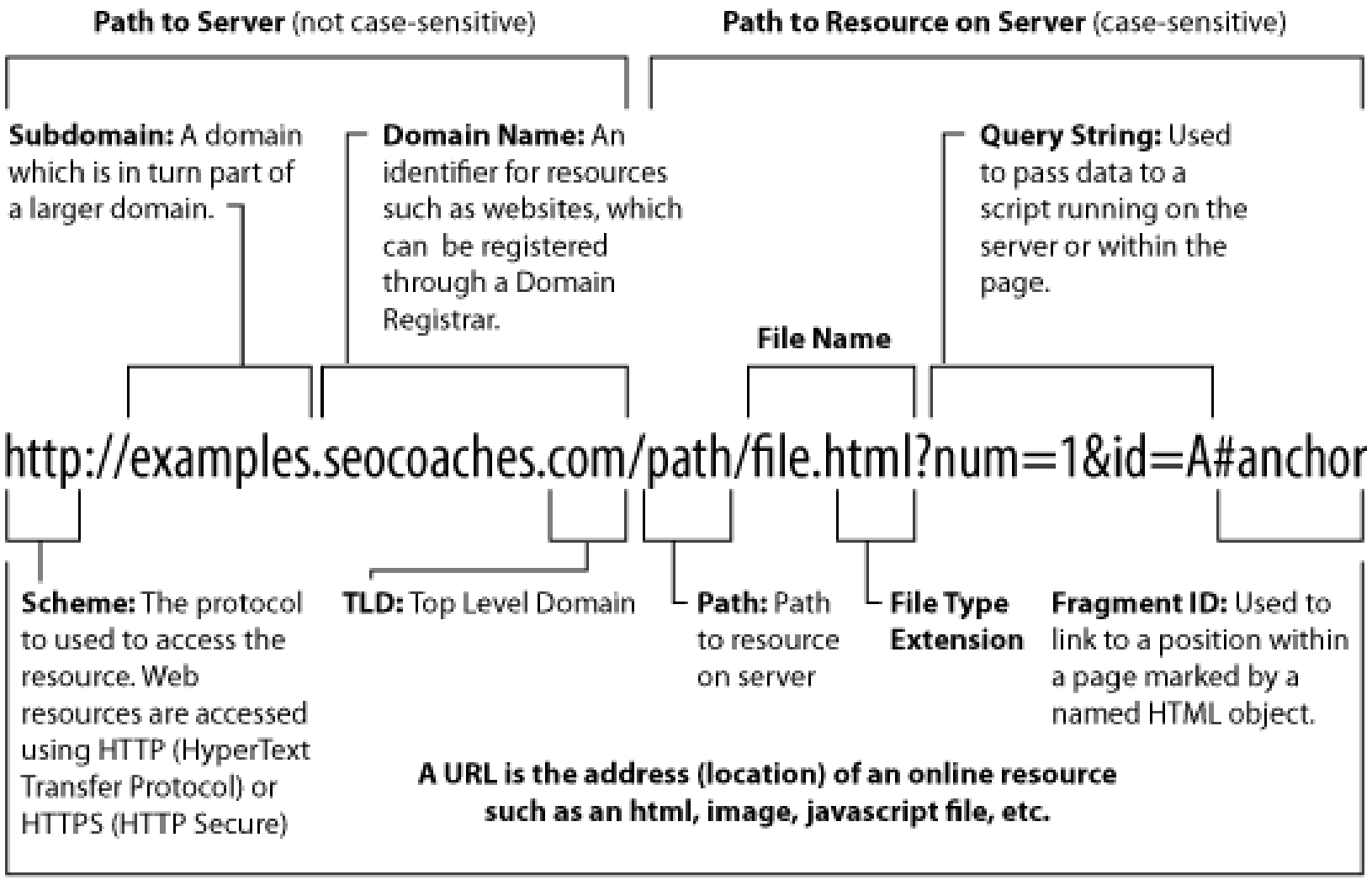
Kapcsolat kezelése

- Nem a HTTP protokoll kezeli a kapcsolatokat, hanem az alatta lévő TCP.
- Mielőtt a kliens és a szerver HTTP kérés/válasz formában kommunikálni tudna létre kell hozni egy TCP kapcsolatot.
  - **HTTP/1.0**: Minden kérés válaszhoz külön TCP kapcsolatot nyit
  - **HTTP/1.1**: Pipelining segítségével a kapcsolat nyitva marad, hacsak nem jön egy Connection: Close header.
  - **HTTP/2**: Egyetlen kapcsolaton keresztül multiplexeli az üzeneteket.
  - **HTTP/3**: QUIC,ami az UDP-re épít.

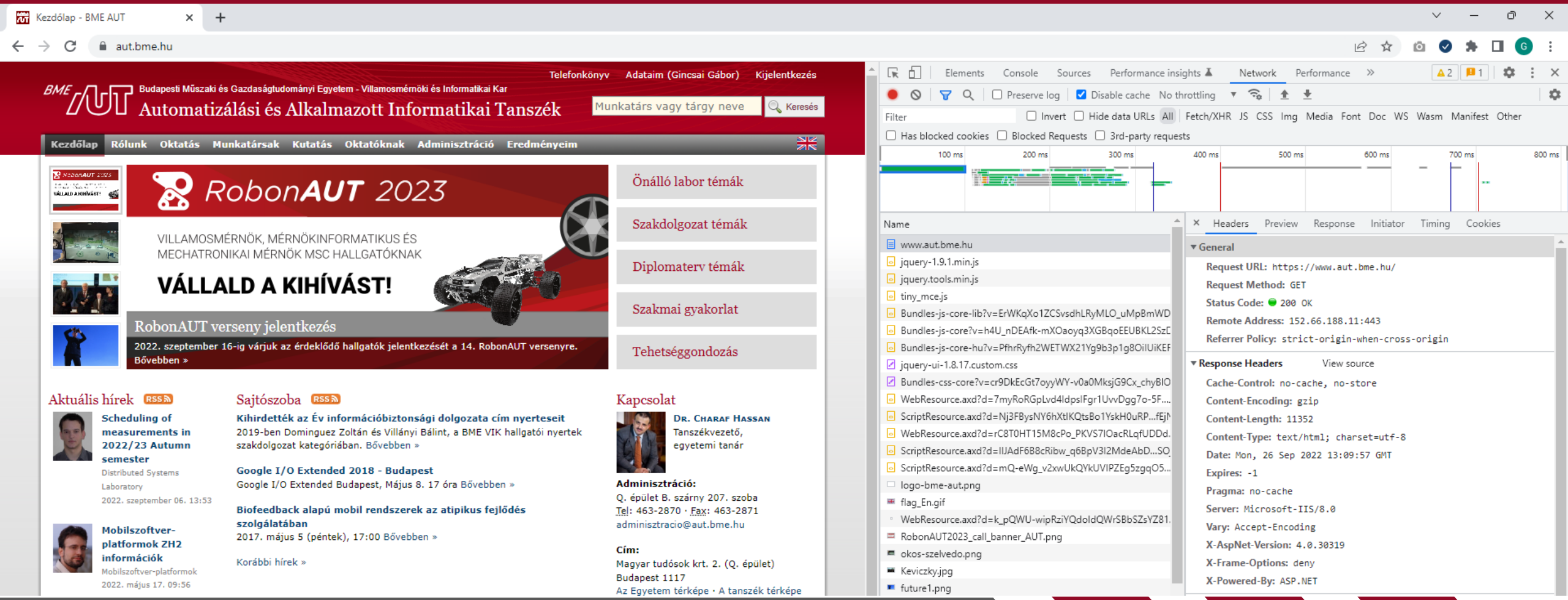


https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/

Hogyan épül fel az URL?



DEMO



Hálózat monitorozása

Böngésző Dev Toolbar – Network tab

A HTTP kérés és válasz elemei

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

A kérés és a válasz elemei

- Metódusok (methods, verbs)
  - GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS, TRACE
- A kért erőforrás (resource)
  - pl.: http://www.aut.bme.hu
- Fejléc mezők
  - Szerverre, tartalomra, biztonságra és gyorsítótárazásra vonatkozó extra információk
- Hibakódok (Status-Code) + Hibaüzenetek (Reason-Phrase)
  - Pl.: 404 – Not Found

Metódusok

- **GET**: a kért erőforrás letöltése a szerverről.
- **POST**: adatot (pl űrlap tartalmát) küld fel a szerverre a kérés body részében feldolgozásra, vagy új erőforrás létrehozásához, beszúrásához.
- **PUT / PATCH**: frissíti a megadott erőforrást a szerveren.
- **DELETE**: törli a megadott erőforrást.
- **HEAD**: HTTP fejléc lekérdezése a megadott erőforrásról.
  - pl. méret, típus, utolsó módosítás dátuma. Nincs benne a BODY!
- **OPTIONS**: visszaadja a szerver által támogatott HTTP metódusok listáját.
- **TRACE**: visszaküldi a kapott kérést (diagnosztika).

Metódusok tulajdonságai

- **Biztonságos (safe) metódus**
  - Csak információ letöltésére szolgál, nincs mellékhatása, nem változtat állapotot a szerveren
    - pl. GET, HEAD, OPTIONS, TRACE.
  - A kliens újra próbálkozhat.
- **Idempotens (idempotent) metódus**
  - Többszöri végrehajtása ugyanazt a hatást váltja ki, mint az egyszeri.
  - Minden biztonságos metódus idempotens is + PUT, DELETE
    - DELETE nem dobhat hibát, hogy az erőforrás nem található.
- POST tipikusan nem idempotens (pl. fórum hozzászólás beszúrása).
  - POST-Redirect-GET (PRG) pattern.
- PATCH általában nem idempotens, mert a részleges frissítés megvalósítása lehet olyan, hogy mellékhatása van.

A kért erőforrás (resource)

- **Uniform Resource Identifier (URI)** azonosítja
- *„A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource.”* (RFC 3986, 61.old)
- **[uri_scheme]:[scheme specific part]**
  - tel:+36 1 4633714
  - mailto:John.Doe@example.com
  - http://www.bme.hu

Uniform Resource Locator (URL) RFC 3986

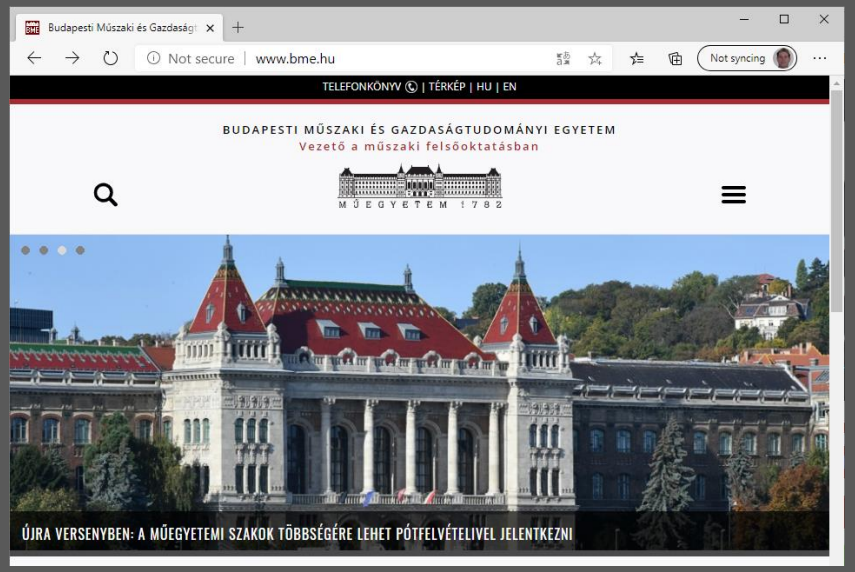
- Speciális URI weboldalak címzésére.
- Meghatározza az erőforrás elérését (location) is.

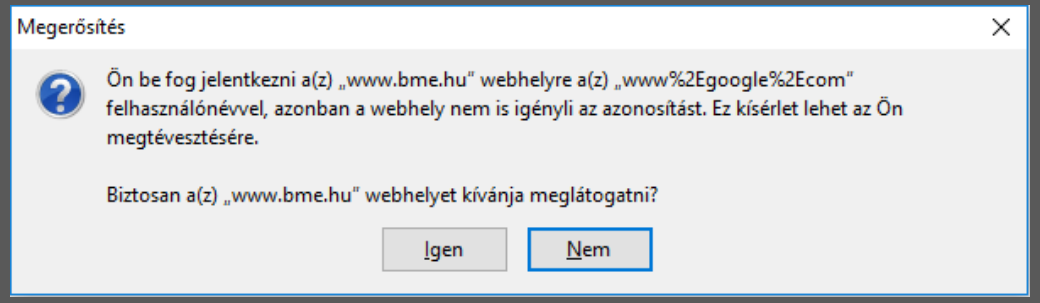
- A fragment nem jut el a szerverre, a kliens dolgozik vele.
- Gyakorlatban használt általános forma:

protocol://username:password@FQDN:port/path/file
?variable1=value1&variable2=value2#fragement

Mi töltődik be az alábbi URL-nél?

http://www.google.com@www.bme.hu





- **Chorme** és **Edge**
  > www.bme.hu-t
- **FireFox**
  > jelzi, hogy mi fog történni.
- Adathalász (phishing) támadások sokszor ezt használják ki.

URL fajtái

- **Absolute URL**: mindentől függetlenül egyértelműen meghatározza az erőforrást
  - http://www.bme.hu/hirek
- **Relative URL**: az aktuális dokumentumhoz vagy a szerver gyökeréhez (root relative) képest relatív út
  - /Oktatas/Lists/Szakiranyok
  - Image%20Library/BulletinImage.jpg
- Van amikor case-sensitive
  - szerver beállítás és kódolás kérdése

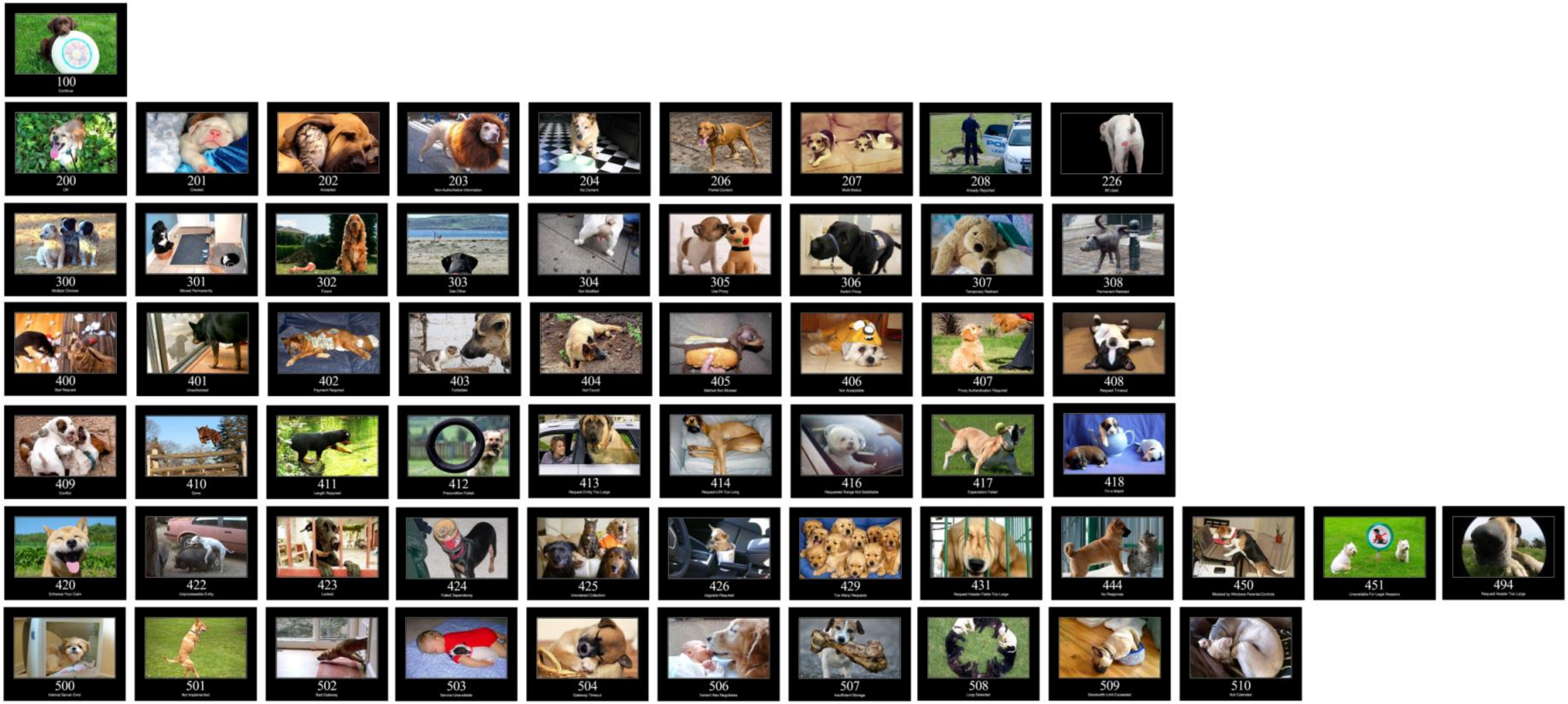
Fejléc mezők (RFC 2616 Section 14)

- **Szerverrel** kapcsolatos mezők
  - `Date: Wed, 21 Aug 2022 08:41:30 GMT`
  - `Server: Apache`
- **Tartalommal** kapcsolatos mezők (például):
  - `Accept: text/html, image/jpeg`
  - `Accept-Encoding: gzip, deflate`
  - `Accept-Language: en-US,hu-HU;q=0.5`
  - `Content-Length: 3495`
  - `Content-Type: text/html`
  - `Content-Disposition: mentendő fájl neve`
  - `Content-Encoding: gzip`

Fejléc mezők (RFC 2616 Section 14)

- **Gyorsítótárral** (cache) kapcsolatos mezők (például):
  - `Cache-Control: no-cache`
  - `Expires: dátum`
  - `If-Modified-Since: dátum`
  - `Last-Modified: dátum`
  - `ETag: verzió`
- **Biztonsággal** kapcsolatos mezők (például):
  - `Authorization: BasicTXlEb21haW5cTXlDb21wdXRlcjpTdXBlclNlY3JldFBhc3N3b3Jk`
  - `WWW-Authenticate: Basic realm="MyComputer"`
  - `X-Frame-Options: SAMEORIGIN`
  - `DNT: 1`

HTTP státusz kódok



Hibakódok (Status-Code) RFC 2616 Sec. 10

- **1xx: Information**
  - 100 Continue
  - 101 Switching protocols (pl. WebSocket)
- **2xx: Successful**
  - 200 OK
  - 201 Created (REST API)
  - 204 No content
- **3xx: Redirect**
  - 301 Moved permanently
  - 302 Found (temporary move)
  - 304 Not modified
- **4xx: Client Error**
  - 400 Bad request
  - 401 Unauthorized
  - 403 Forbidden
    - 403.5: SSL required
    - 403.6: Forbidden: IP address rejected
  - 404 Not found
  - 405 Method not allowed
  - 410 Gone
  - 413 Request entity too large
  - 414 Request URI too long
- **5xx: Server Error**
  - 500 Internal server error
  - 503 Service unavailable

Representational State Transfer (REST)

- Webes API, amin keresztül adatok kérdezhet le / módosíthat a kliens.
  - Erőforrás alapú (URI)
  - Az adat JSON-ban vagy XML-ben utazik.
  - HTTP method-okat (verb-eket) használ
- Példa: https://example.hu/api/person/1
  - Erőforrás: person
  - Szolgáltatás: elérhetőségi adatok (GET)
  - Representation: név, cím, telefonszám (JSON-ban)
- Mi hiányzik ebből?
  - CRUD-on túli műveletek
  - Pl: Ki akarjuk törölni egy könyv címének első n karakterét! Milyen végpontot definiáljunk?
    - URL: /books/<isbn>/delete-from-title
    - GET: jó lehet, bár félrevezető.
    - POST: jobb is, mert parametéret is kell küldeni (n)

Backend as a service

- Felhő alapú szolgáltatási modell
- A frontend fejlesztő outsource-olja a backend fejlesztést
- Megadunk egy adatstruktúrát és ahhoz automatikusan kapunk egy adatbázist és CRUD műveleteket REST végpontokon keresztül.
- Például a *Firebase* egy ilyen Backend as a service szolgáltatás.

REST API

- **GET: Lekérdezi** az adott URL-en található erőforrást. A válasz üzenet törzse tartalmazza a kért erőforrás részleteit.
- **POST: Létrehoz** egy új erőforrást a megadott URL-en. Az üzenet törzsében kell megadni az új erőforrás adatait. Figyeljünk rá, hogy a POST olyan műveletek indítására is használható, amelyek valójában nem hoznak létre erőforrásokat.
- **PUT: Létrehozza, vagy lecseréli** az adott URL-en lévő erőforrást. A kérés törzsében megadott erőforrás kerül létrehozásra vagy frissítésre.
- **PATCH: Részlegesen frissíti** az erőforrást. A kérés törzsében meghatározott módosítások kerülnek végrehajtásra az erőforráson.
- **DELETE: Törli** az adott URL-en lévő erőforrást.

Miben tér el a POST, PUT és a PATCH

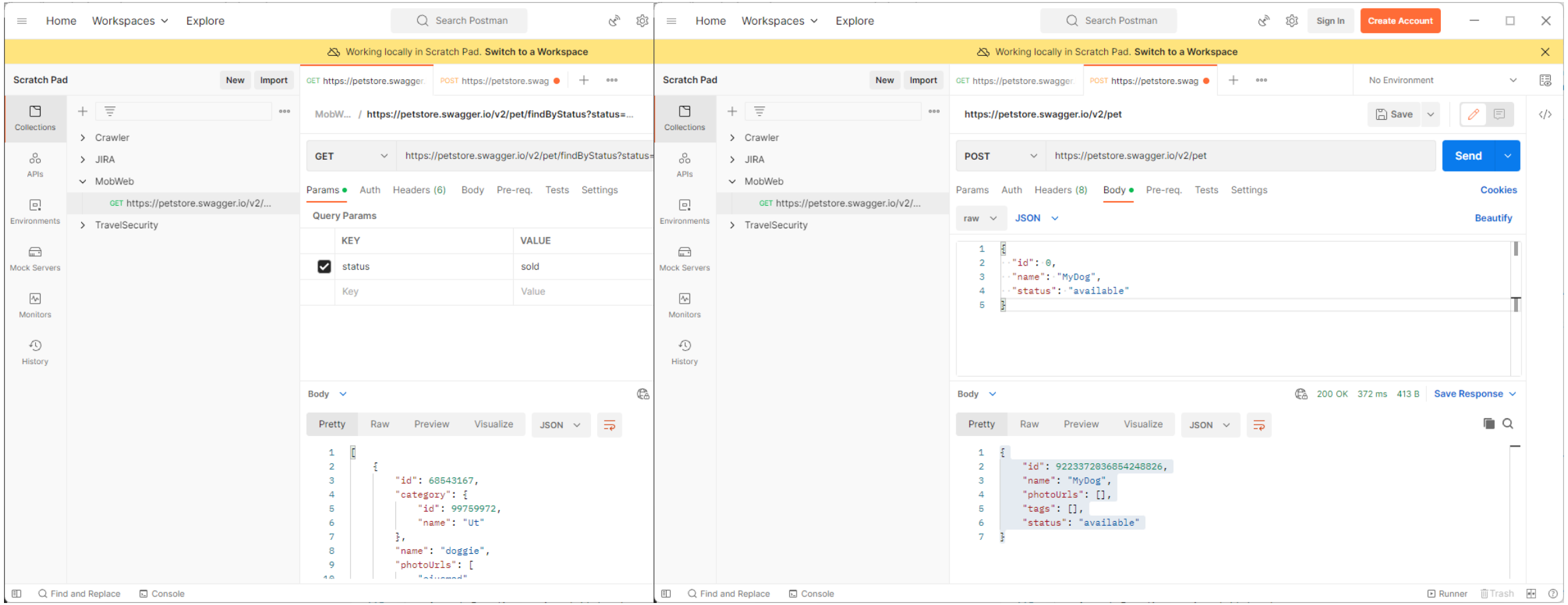
- A POST kérés létrehoz egy új erőforrást. A szerver hozzárendel egy URI-t, amit visszaad a kliensnek.
  - Általában gyűjteményekhez tartozó URI-n használjuk Ilyenkor az új erőforrás hozzáadása került a gyűjteményhez.
  - Létező erőforrások esetén használhatjuk adatok feldolgozásra küldésére is a nélkül, hogy új erőforrást hoznánk létre.
- A PUT kérés létrehoz egy új vagy frissít egy létező erőforrást. A kliens adja meg az erőforrás URI-ját és a body-ban az erőforrás összes adatát. Ha az URI-n már létezik erőforrás, akkor azt lecserélni, egyébként új erőforrást hoz létre.
  - Általában olyan egyedi erőforrásokon használjuk, mint például egy konkrét vevő.
  - Lehet, hogy a szerver támogathatja PUT-on keresztül a frissítést, de a létrehozást nem.
  - A PUT-on keresztüli létrehozás támogatása attól függ, hogy a kliens értelmesen hozzá tud-e rendelni egy URI-t az erőforráshoz, mielőtt az létezne. Ha nem, akkor használjuk a POST-ot az erőforrások létrehozásához, a PUT-ot vagy PATCH-et pedig a frissítéshez.
- A PATCH kérés részlegesen frissíti a létező erőforrást. A kliens adja meg az erőforrás URI-ját és a body-ban hogy milyen adatokat mire kell módosítani.
  - Hatékonyabb, mint a PUT, mivel a kliens csak a változtatásokat küldi, nem a teljes erőforrást.
  - Technikailag a PATCH új erőforrást is létrehozhat ("null" erőforrás frissítéseinek megadásával), ha a szerver támogatja.

Példa egy vevőket kezelő REST API-ra

Resource	POST	GET	PUT	DELETE
/customers	Létrehoz egy új vevőt.	Visszaadja az összes vevőt.	Tömegesen frissíti a vevők adatait.	Törli az összes vevőt.
/customers/**1**	Hiba	Visszaadja az 1-es azonosítójú vevő adatait.	Frissíti az 1-es azonosítójú vevő adatait, ha létezik.	Törli az 1-es azonosítójú vevőt.
/customers/**1/orders**	Új megrendelés létrehozása az 1-es azonosítójú vevőhöz	Visszaadja az 1-es azonosítójú vevő összes megrendelését.	Tömegesen frissíti az 1-es azonosítójú vevő megrendeléseit.	Törli az 1-es azonosítójú vevő összes megrendelését.

Postman

- REST API tesztelésre használt webes / asztali kliens



A HTML szabvány

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Egy kis történelem

Tim Berners-Lee

1991 HTML első megjelenése

1995 HTML 2.0

1997 HTML 3.2

1999 HTML 4.01

2000 XHTML 1.0 – Ez lesz a jövő!

2002 XHTML 2.0 - A nem szabványos oldalakat nem szabad

2009 megjeleníteni. 2009-re rájöttek, hogy ez zsákutca.

38

2004 Elkezdik fejleszteni a HTML 5-öt

2012 HTML5 megjelenése - Sokkal megengedőbb, visszafelé kompatibilis.

WHATWG

HTML Living Standard

https://html.spec.whatwg.org/multipage/

W3C

2016 HTML 5.1

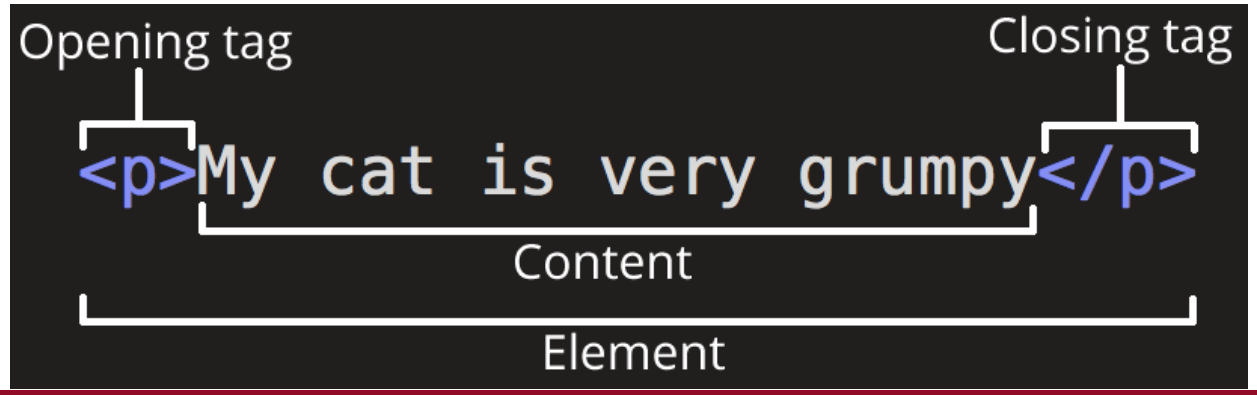
2017 HTML 5.2

Mi a HTML?

- Hypertext Markup Language
- Jelölőnyelv ami leírja a böngészőnek, hogyan épül fel a weboldal **struktúrája.**
- HTML elemekből épül fel
  - Ez határozza meg hogy mi jelenjen, illetve hogyan viselkedjen az adott rész.
  - Pl. a kép az <img> elem, a táblázat pedig a <table>

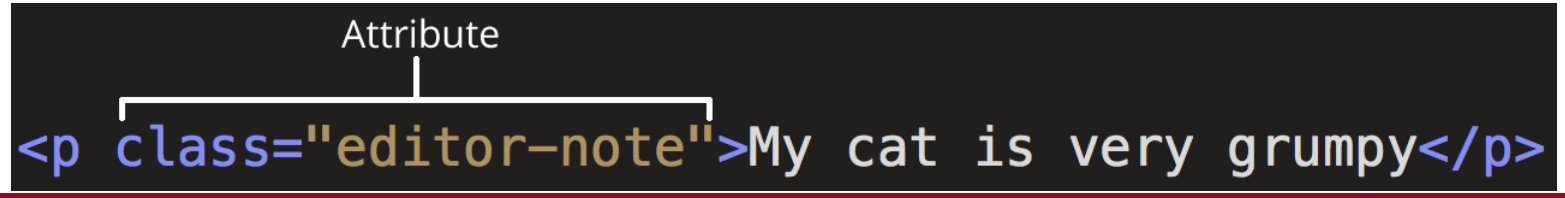
Hogyan épül fel egy HTML elem?

- Nyitó tag: Az elem neve < és > között.
  - A tag nem kis-nagybetű érzékeny, de ajánlott kisbetűkkel írni.
- Lezáró tag: azonos a nyitó taggel, de </-el kezdődik.
- Tartalom: A két tag közötti rész, ami jelenleg szöveg.
  - Ha a tag nem tartalmazhat semmit (üres), akkor lezáró tagje sincs.
  - Pl: <img src="..."> vagy
 DE <script src="..."></script>



Attribútumok

- Extra információt adnak az elemhez
  - Pl.: Elem egyedi azonosítója, neve, CSS osztályok…
- A nyitó tagbe kell megadni, szóközökkel elválasztva egymástól.
- Az attribútum neve után = jel következik
  - De a bool értékű attribútumoknál elhagyható az érték
  - Pl.: disabled="true", disabled="disabled" vagy disabled
- Az attribútum értékét idézőjelek közé tesszük
  - Használhatjuk a " vagy a ' idézőjelet is.



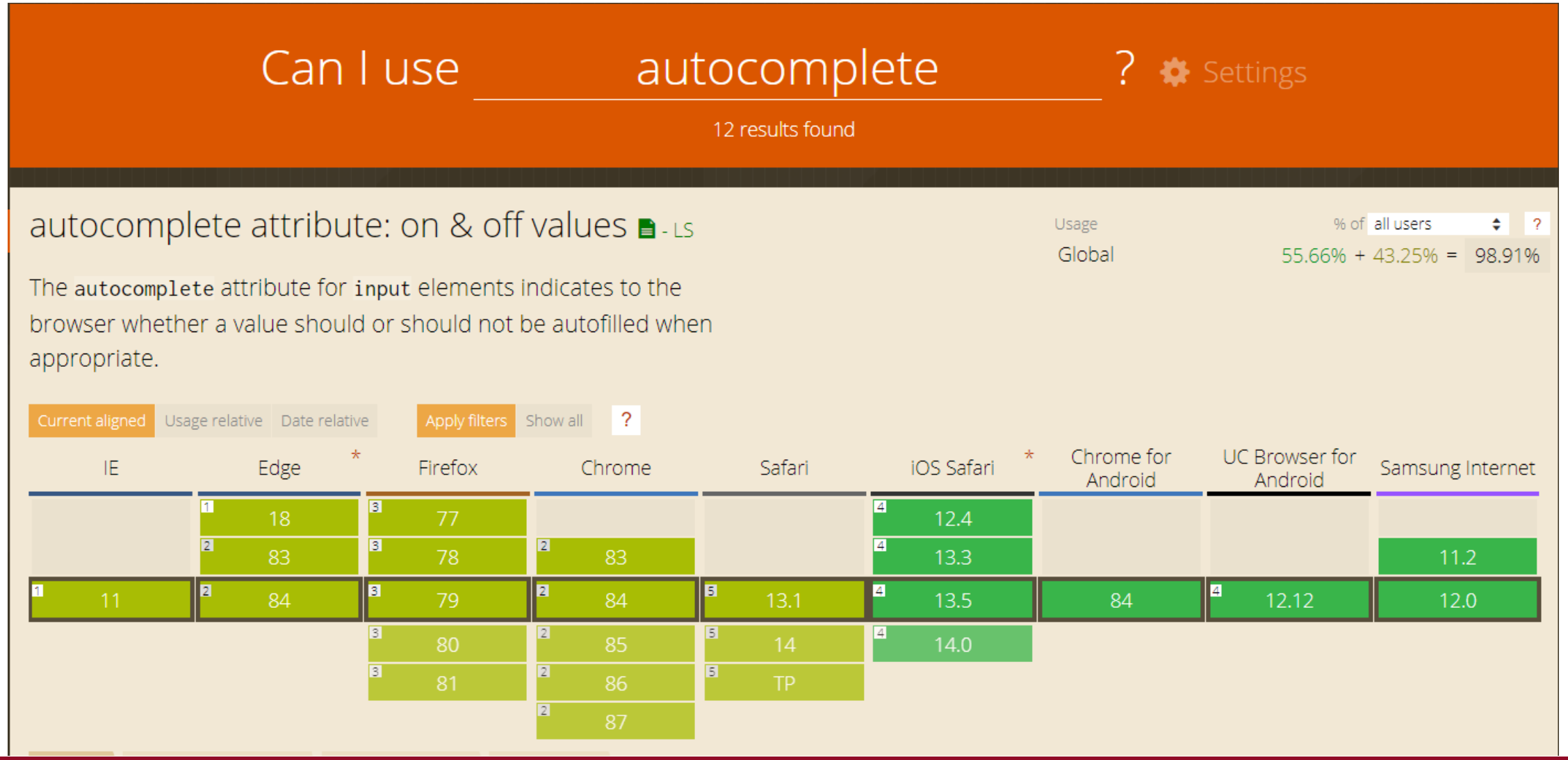
Leggyakoribb attribútumok

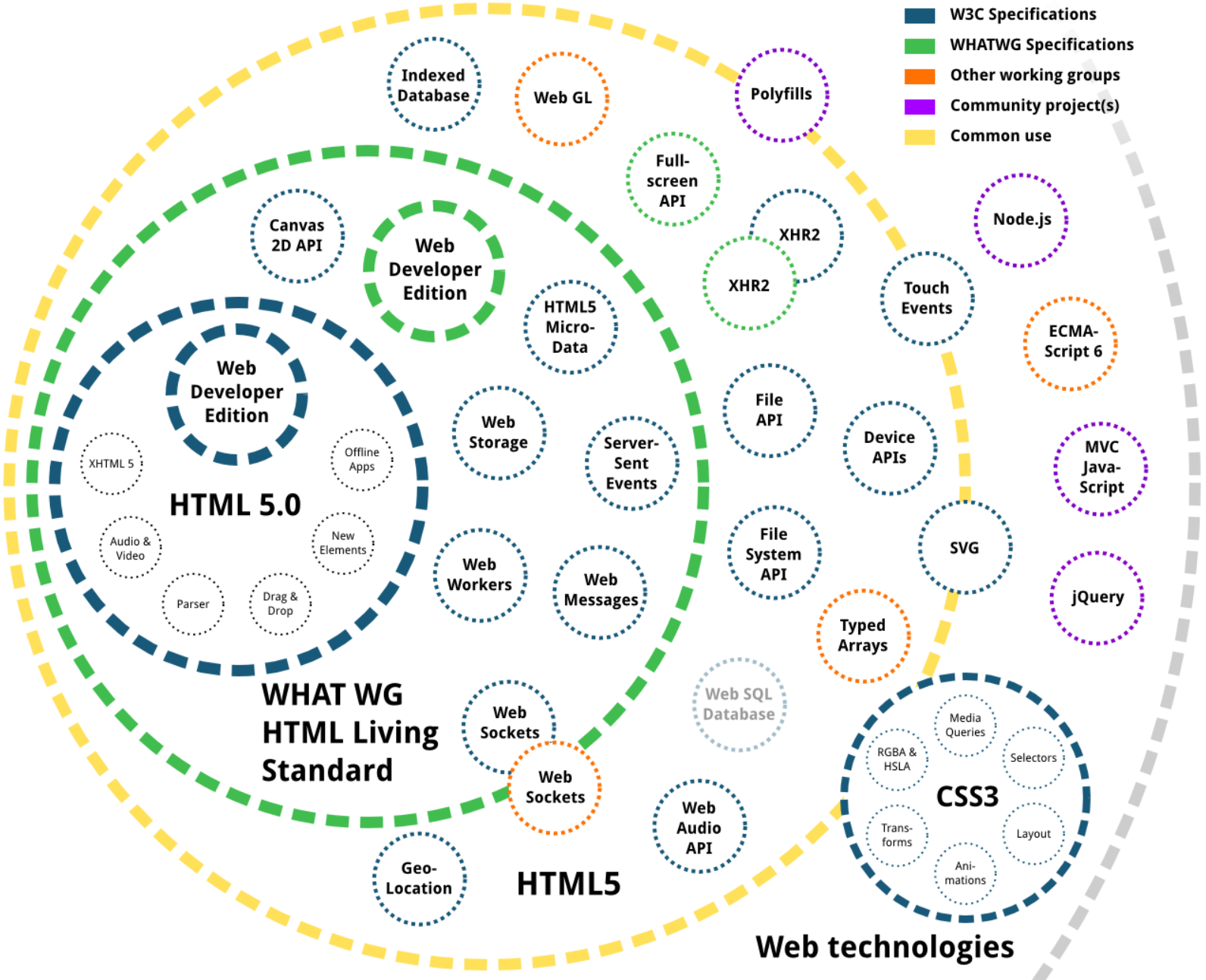
- **id**: Egyedi azonosítója egy elemnek
  - Később ez alapján tudjuk majd kiválasztani az elemet
  - Ha több elemnek is azonos az id-ja az oldalon, nem kapunk hibát, de a jQuery hibásan fog működni.
- **title**: Az elem címe, általában tooltipként jelenik meg
- **class**: A megadott CSS osztály ráteszi az elemre
  - Ehhez a <head>-ben linkelni kell a CSS fájlt is.
- **style**: CSS szabályt lehet közvetlenül az elemre tenni
  - Ha lehet kerüljük az inline style-okat.

Böngésző támogatás

- Nincs 100%-ban „HTML kompatibilis” böngésző
  - Némelyik ezt támogatja belőle, némelyik azt…
- A HTML mozgó célpont: bekerülnek, kikerülnek, változnak dolgok, folyamatosan frissül
- Egyes elemek a különböző böngészőkben másképpen viselkednek vagy jelennek meg.
- Megoldás: külön-külön ellenőrizzük, hogy a támogatni szánt böngészők mit tudnak, és mit nem.
  - Webes projekt legfontosabb kérdése, hogy mik a minimális böngészőverziók

Támogatja-e a böngésző a ... taget?




W3C Specifications
WHATWG Specifications
Other working groups
Community project(s)
Common use
Indexed Database
Web GL
Polyfills
Full-screen API
Canvas 2D API
Web Developer Edition
XHR2
XHR2
Node.js
Touch Events
ECMA-Script 6
HTML5 Micro-Data
Web Developer Edition
File API
Device APIs
MVC Java-Script
Web Storage
Server-Sent Events
XHTML 5
Offline Apps
HTML 5.0
File System API
SVG
jQuery
Audio & Video
New Elements
Web Workers
Web Messages
Parser
Drag & Drop
Typed Arrays
Web SQL Database
WHAT WG HTML Living Standard
Web Sockets
Web Sockets
Web Audio API
Media Queries
RGBA & HSLA
Selectors
CSS3
Trans-forms
Layout
Ani-mations
Geo-Location
HTML5
Web technologies

Hogyan készítsünk HTML oldalt?

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Helló világ HTML-ben

- <!doctype>
  - html verzióját és típusát adja meg.
- <html>
  - az oldal gyökéreleme, ezen belül található minden.
- <head>
  - Az oldal címét állítjuk be és egyéb metaadatokat.
- <body>
  - Az oldal tartalma, ami megjelenik a böngészőben.

HelloVilag.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Oldal címe</title>
5      </head>
6      <body>
7          Helló világ!
8      </body>
9  </html>
```

<head>

- A head szekcióban adhatunk meg az oldalra vonatkozó meta adatokat.
  - Karakter kódolás, oldal címe, cachelés beállítása…
- Itt hivatkozhatunk CSS fájlokra
  - JavaScript fájlokat a body lezáró tag előtt érdemes.

```
<head>
      <meta charset="utf-8">
      <title>Első HTML5 oldalam</title>
      <link rel="stylesheet" href="style.css">
      <script src="script.js"></script>
</head>
```

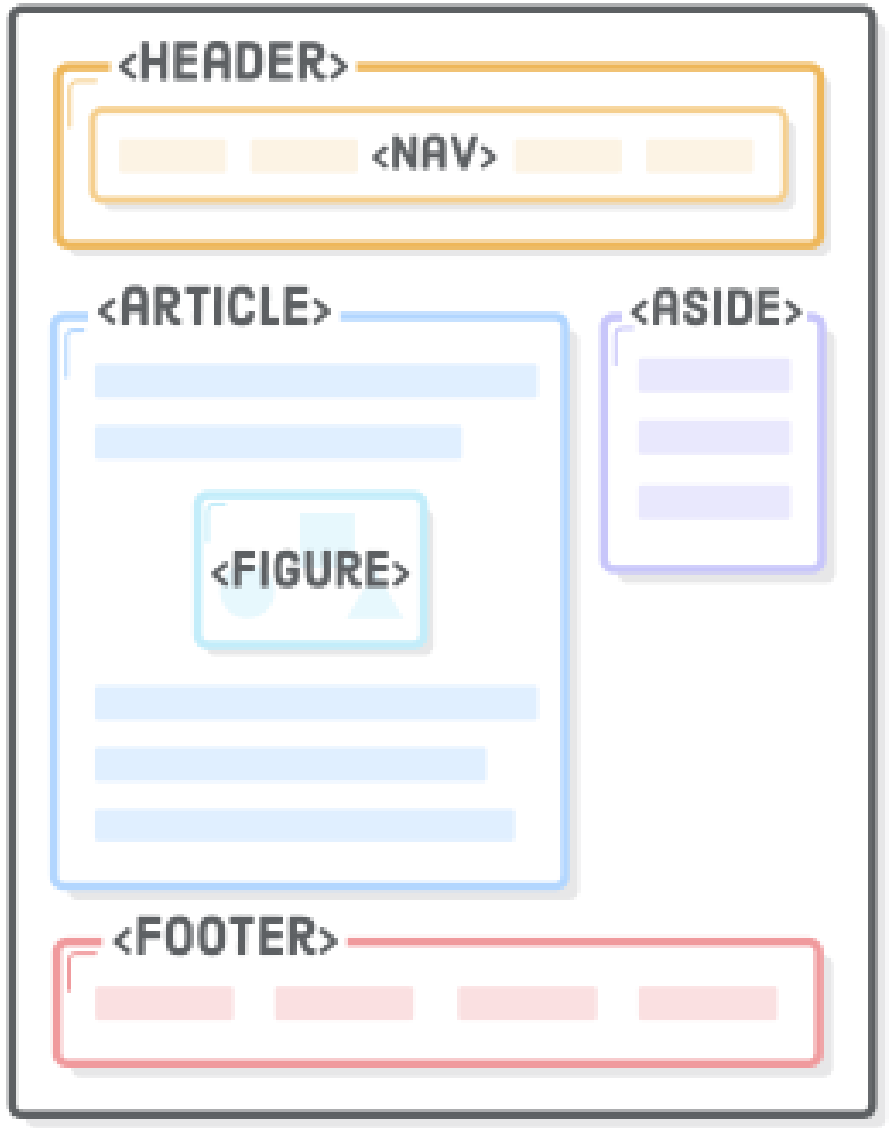
Metaadatokkal bővített Helló Világ

HelloVilagMeta.html

```html
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Első HTML5 oldalam</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <p>Helló világ!</p>
    <!-- TODO: Oldal kódja -->

    <script src="script.js"></script>
</body>
</html>
```

Oldalváz

- **<header>**
  - Az oldal vagy egy section fejléce.
- **<nav>**
  - Navigációs lineket fogja össze
- **<aside>**
  - Oldaljegyzet, ami a mellette lévő tartalomhoz nem
- **<section>**
  - Az oldalt logikai egységre bontására használjuk.
- **<article>**
  - Egy önállóan értelmezhető része az oldalnak.
  - Pl.: cikk vagy hozzászólás…
- **<footer>**
  - Az oldal, vagy akár egy article lábléce.



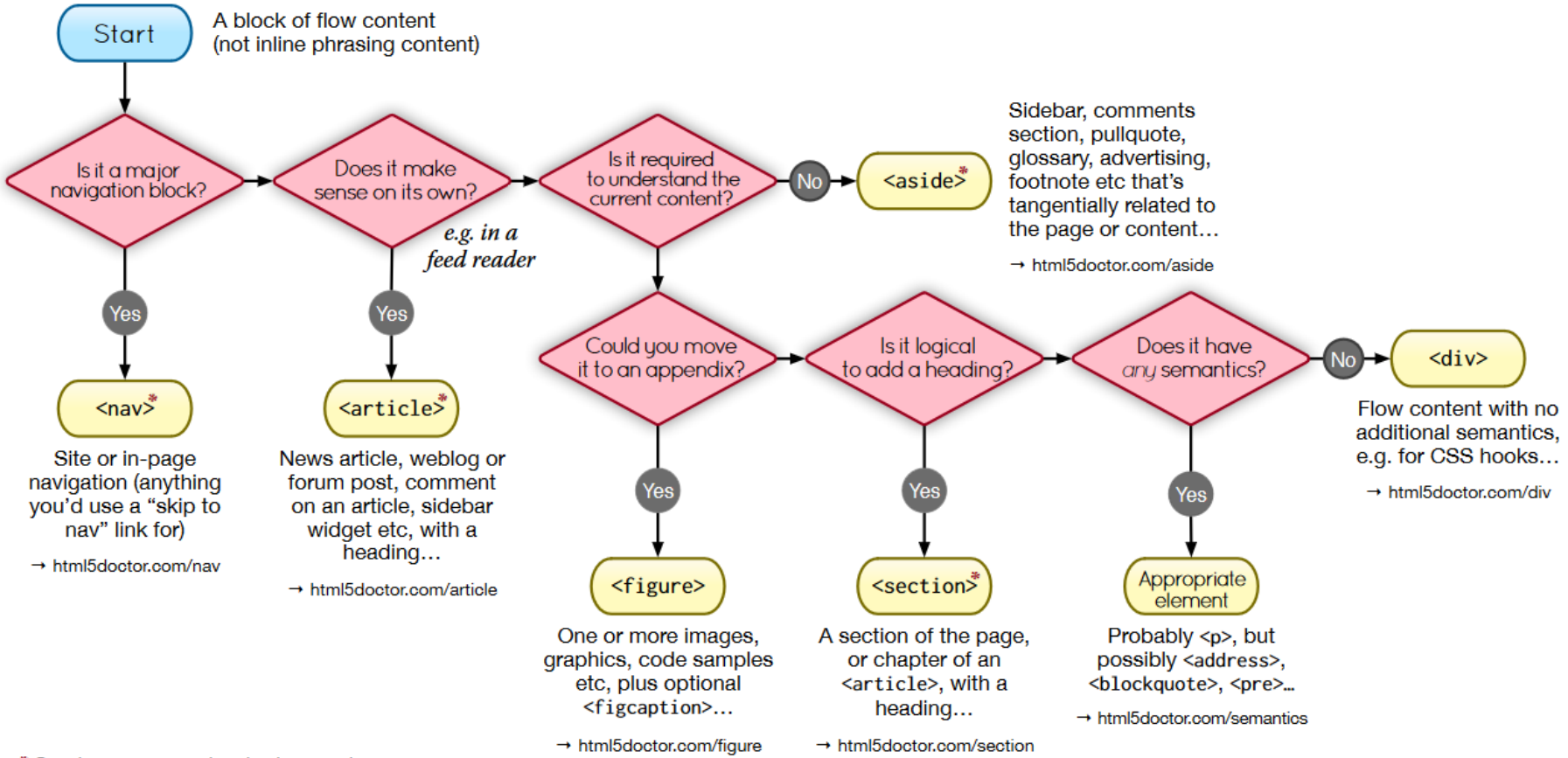
Miért jobb a szemantikus oldalváz?

- Az egyes elemek jelentéssel bírnak
  - A <div id="..." –nak nincs semmi extra jelentése
  - Több szemantikai információ az új tagekben.
  - A böngészők, keresőmotorok, felolvasó szoftverek értelmez*hetik* → értelmezhető az oldal felépítése.
- A Google a header-ökben megadott értéket fontosabbnak tartja mint ami a footerben van.
- A <nav>-ban megadott elemekről tudjuk, hogy hivatkozások. Egy felolvasó szoftvernek ez segíthet.
- Megadja az oldal struktúráját.

HTML5 Element Flowchart

Sectioning content elements and friends

By @riddle & @boblet
www.html5doctor.com



* Sectioning content element

These four elements (and their headings) are used by HTML5's outlining algorithm to make the document's outline
→ html5doctor.com/outline

2011-07-22 v1.5
For more information:
www.html5doctor.com/semantics

Blokk és inline elemek

Blokk elemek

- Mindig új sorban jelennek meg.
- Blokk elemet csak blokk szintűbe lehet beágyazni.
- Pl.: , paragrafus, lista, navigációs menü, lábléc

Inline elemek

- A böngésző ugyanabban a sorban az előtte lévő elem mögött jeleníti meg.
- Blokk szintű elem tartalmának egy része
- Pl.: paragrafuson belüli hivatkozások, vastag betűs szöveg

Bekezdés - <p>

- Szövegblokkon belül új bekezdést a <p> taggel adhatunk meg.
  - Tagek használatával is befolyásolhatjuk a megjelenést
    - <i> - dőlt
    - <b> - vastag
    - <u> - aláhúzott

```
<p>
   Sima szöveg, lehet <b>vastag</b> vagy <i>dőlt</i>.
</p>
```

- Ha szemantikailag is ki szeretnénk emelni a tartalmat
  - - fontos tartalom, alapértelmezés szerint vastag betűvel szedve.
  - - kiemelt tartalom, alapértelmezés szerint dőlt betűvel szedve.

```
<p>
   Sima szöveg, de <strong>fontos</strong> vagy
   <em>kiemelet</em> is.
</p>
```

Oldalak közti navigáció (hyperlink)

- Az oldalak közötti navigáció: <a> tag

```
<a href="http://www.aut.bme.hu">AUT portál</a>
```

- Link megnyitása új tabon

```
<a href="http://www.aut.bme.hu"
   target="_blank">AUT portál</a>
```

- Levelező program megnyitása a kliensen

```
<a href="mailto:xy@example.com?Subject=Hello">
                Levél küldése!</a>
```

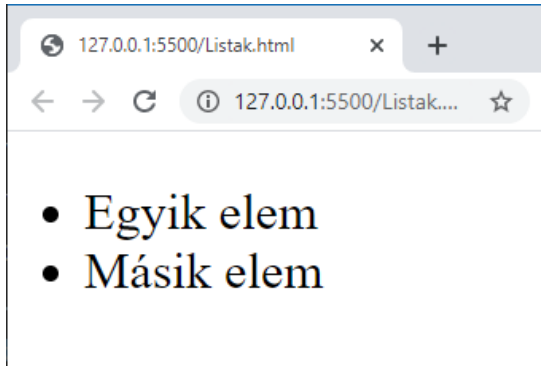
Kép - <img>

- HTML-be képet az <img> tag segítségével ágyazhatunk be
- Az src="..." attribútumban adhatjuk meg, hogy mit jelenítsen meg.
  - URL:
    - src="http://www.aut.bme.hu/Static/img/logo-bme-aut.png"
  - Base64 enkódolt adat:
    - src=" iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU  Jggg=="
- Helyettesítő szöveget az alt="..." attribútummal adhatunk meg.
- Megadhatjuk a magasságát (height) és szélességét (width)
  - Ha megadjunk mindkettőt, akkor torzíthatja a képet.
  - Érdemes csak az egyiket megadni.

Listák - és

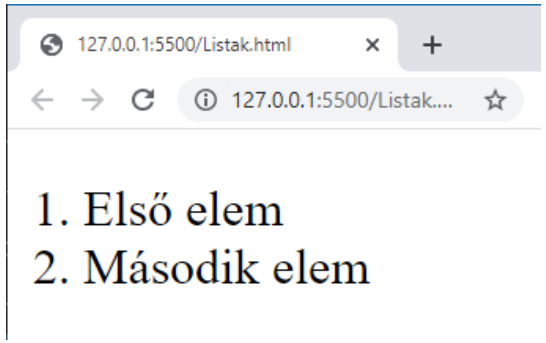
- Egyszerű felsorolást az <ul> tag segítségével hozhatunk létre.
  - Az egyes listaelemeket <li> tagbe kell tenni

```
<ul>
    <li>Egyik elem</li>
    <li>Másik elem</li>
</ul>
```



- Sorszámozott felsorolást az <ol> tag segítségével készíthetünk.
  - Az egyes listaelemeket <li> tagbe kell tenni

```
<ol>
    <li>Első elem</li>
    <li>Második elem</li>
</ol>
```



Táblázatok - <table>

- Egy táblázat <table>
  - sorokból <tr> és
  - cellákból <td> épülnek fel.
- Megadható
  - fejléc <thead> és
  - tartalmi része <tbody> is

```
<table>
    <tr>
        <td>1. sor 1. cella</td>
        <td>1. sor 2. cella</td>
    </tr>
    <tr>
        <td>2. sor 1. cella</td>
        <td>2. sor 2. cella</td>
    </tr>
</table>
```

```
<table>
    <caption>Csoport lista</caption>
    <thead>
        <tr>
            <th scope="col">Név</th>
            <th scope="col">Életkor</th>
            <th scope="col">Jel</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th scope="row">Gábor</th>
            <td>39</td>
            <td>zászló</td>
        </tr>
    </tbody>
</table>
```

Név	Életkor	Jel
Gábor	39	zászló

Csoport lista

HTML űrlapok

Kliensalkalmazások

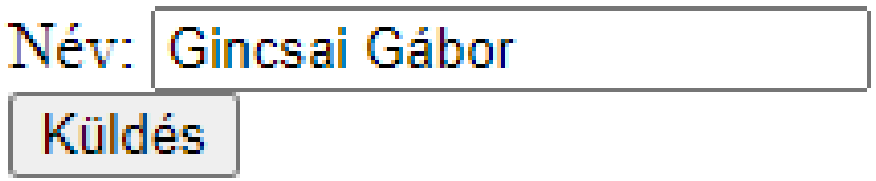
BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

HTML űrlap <form>

- Adatok bekérése a felhasználótól és továbbítása a szerverre.
  - címkék
    - <label>
  - beviteli mezők
    - <input type="..."
  - gombok
    - <button>
    - <input type="submit">

```
<form action="register.aspx" method="get">
  <label for="name">Név:</label>
  <input type="text" id="name"
       name="name" value="Gincsai Gábor">
  <br>
  <input type="submit" value="Küldés">
</form>
```



<input> tag típusok

- Egyszerű beviteli mező
  - text / password / number (A többsoros más tag: <textarea>)
- Választós mezők
  - radio / checkbox
- Gombok
  - button / submit / reset (használható a <button> tag is)
- Fájl
  - File
- Dátum típusok
  - date / datetime/ datetime-local / month / time / week
- Egyéb gyakori típusok
  - email / range / search / tel / url / color

Beviteli mező előtti címke: <label>

- Cél: a megjelenített szöveg és a beviteli mező összetartozzon.
  - A címkére kattintva a fókusz a beviteli mezőre ugrik
- Egyszerűen a beviteli mezőt a label-be ágyazzuk.
- A *for* attribútummal megadjuk, hogy melyik inputhoz tartozik.

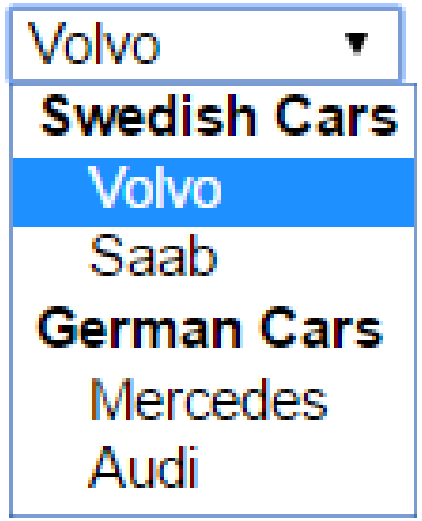
```
<label>Név:
  <input type="text" id="fullName">
</label>

<label for="fullName">Név:</label>
<input type="text" id="fullName">
```

Fontosabb input attribútumok

- Helyőrző szöveg megadása
  - `<input type="text" placeholder="keresés">`
- Csak olvasható input
  - `<input type="text" readonly>`
- Letiltott elem
  - `<input type="text" disabled>`
- Korábbi értékeket ne kínáljon fel
  - `<input type="text" autocomplete="off">`
- Automatikusan kapjon fókuszt
  - `<input type="text" autofocus>`

Legördülő lista: <select>

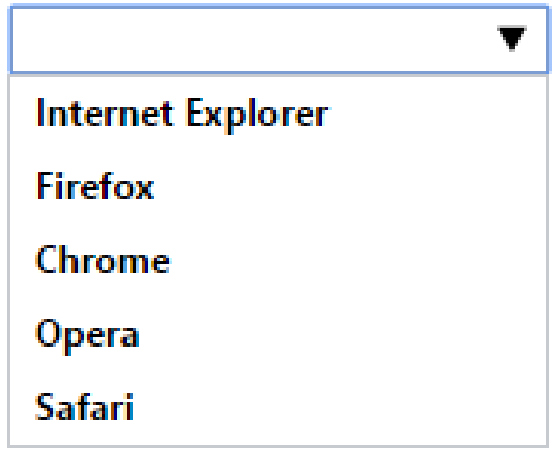
- Elemek megadása
  - option
- Elemek csoportosítása
  - optgroup

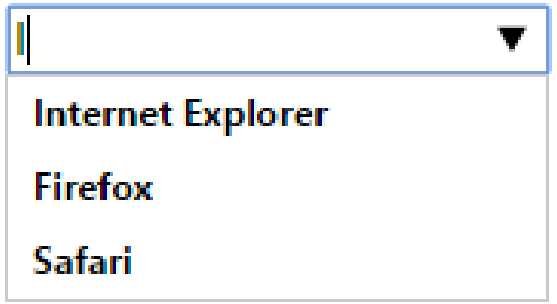



```
<select>
  <optgroup label="Swedish Cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </optgroup>

  <optgroup label="German Cars">
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
```

Szűrhető lista: <datalist>

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```





Fájl feltöltés

- Olyan input amivel fájlt lehet kiválasztani feltöltéshez.

```
<input type="file">
```

- Formon (enctype) / inputon a formenctype megadása

```
<input formencType="multipart/form-data">
```

```
<form … encType="multipart/form-data">
```

- Több fájl feltöltésére a multiple attribútum használható

```
<input type="file" multiple>
```

- Kiválasztott fájlok átadása egy JavaScript eseménykezelőnek

```
<input type="file" multiple onchange="handleFile(this.files)">
```

Zene lejátszása HTML-ből

- Támogatott formátumok
  - mp3
  - Wav
  - Ogg (Safari kivételével)

Element					
<video>	4.0	9.0	3.5	4.0	10.5

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  A böngésző nem támogatja az audio taget.
</audio>
```

Video lejátszása HTML-ből

- Támogatott formátumok
  - mp4
  - webM (Chrome, FF, Opera)
  - Ogg (Chrome, FF, Opera)

Element					
<video>	4.0	9.0	3.5	4.0	10.5

```
<video width="320" height="240" controls autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  A böngésző nem támogatja a video lejátszását.
</video>
```

HTML validáció

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Validáció célja

- A felhasználónál érjük el, hogy kitöltse az űrlapot
  - Kötelező mezők definiálása
  - Adat típusok definiálása (szám / email ...)
  - Tetszőleges formátum reguláris kifejezéssel
- Mindezt HTML-ből JavaScript nélkül
- Ha egy fomot nem kell validálni: **novalidate**

Validációs attribútumok

- Kötelezően kitöltendő mező

```
<input type="text" required>
```

- Validáció reguláris kifejezéssel

```
<input type="text" name="code"
       pattern="[A-Za-z]{2}"
       title="Két betűs országkód">
```

- Min/max hossz (karakterszám)

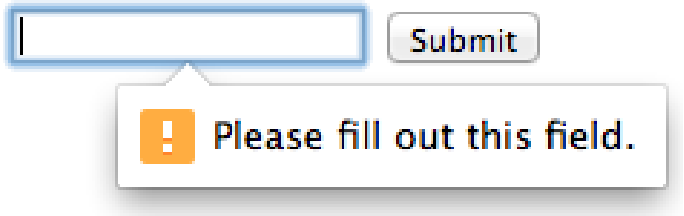
```
<input type="text" maxlength="20">
```

- Min/max érték és a lépésköz megadása

```
<input type="number" min="1" max="10" step="1">
```

Hogyan jelenik meg a böngészőben?

Chrome 30



Internet Explorer 10

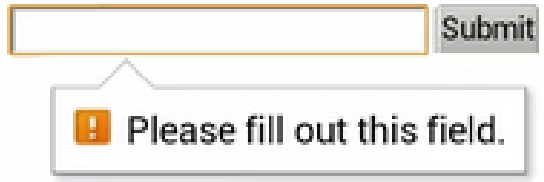
Firefox 23

Firefox for Android

Opera 12

Opera Mobile

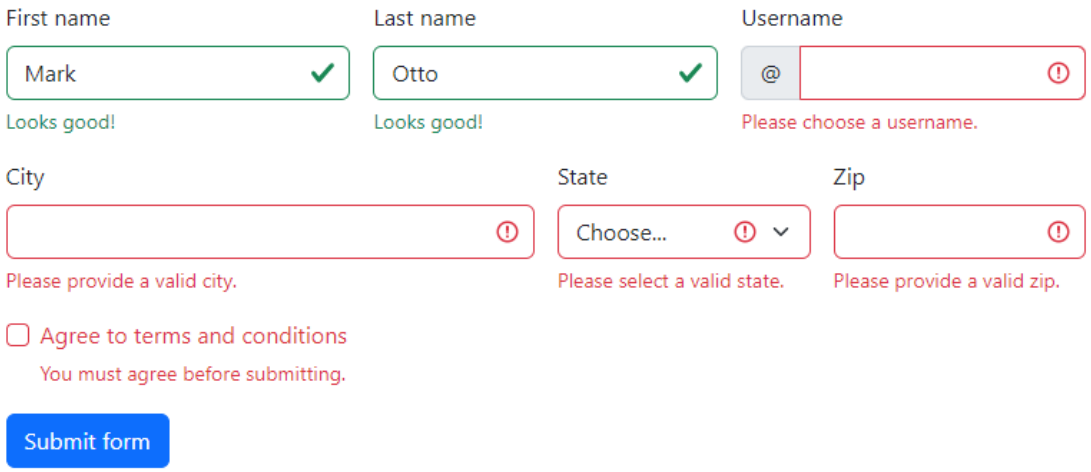
Chrome for Android



Forrás: https://www.html5rocks.com/en/tutorials/forms/constraintvalidation/

Ha szépet szeretnénk...

- Boostrap segítségével tehetjük meg.




```
<form class="row g-3 needs-validation" novalidate>
  <div class="col-md-4">
    <label for="validationCustom01" class="form-label">First name</label>
    <input type="text" class="form-control" id="validationCustom01" value="Mark" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationCustom02" class="form-label">Last name</label>
    <input type="text" class="form-control" id="validationCustom02" value="Otto" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationCustomUsername" class="form-label">Username</label>
    <div class="input-group has-validation">
      <span class="input-group-text" id="inputGroupPrepend">@</span>
      <input type="text" class="form-control" id="validationCustomUsername"
             aria-describedby="inputGroupPrepend" required>
      <div class="invalid-feedback">
        Please choose a username.
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <label for="validationCustom03" class="form-label">City</label>
    <input type="text" class="form-control" id="validationCustom03" required>
    <div class="invalid-feedback">
      Please provide a valid city.
    </div>
  </div>
```

Validációs API

- Validációt JS kódból is kiválthatjuk
  - **willValidatie**: Fogja-e validálni a mezőt az űrlap elküldésekor.
  - **validity**: Egy ValidityState objectet ad vissza, amiben minden típusú validációhoz megkajuk, hogy érvényes vagy nem.
    - Pl: customError, tooLong, valueMissiong, valid.
  - **validationMessage**: Hibaüzenet amit a böngésző megjelenít ha az adott elem nem érvényes.
  - **checkValidity**(): Validálja az adott mezőt, vagy ha az űrlapon hívjuk, akkor a teljes űrlapot validálja.
  - **setCustomValidity**(): A validációs hibaüzenetet lehet vele lecserélni.

Állapotmegőrzés

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

Probléma

- HTTP állapotmentes (stateless)
  - Az egyes kérés-válasz párok között a protokoll nem biztosít állapotmegőrzést.
  - Nem alakul ki munkamenet (session).
- Miért van szükség állapotkezelésre?
  - „memory for websites”
  - elég egyszer bejelentkezni egy webalkalmazásba.
  - webáruházban megmarad a kosár tartalma.
  - testreszabási beállítások megmaradnak. Pl: Neptun skin ☺

Megoldási lehetőségek (kliens -> szerver)

A munkamenethez (session) tartozó információk minden kérésnél és válasznál utaznak a böngésző és a szerver között.

- Előny: Nem igényel szerver oldali erőforrást
  - sok felhasználóra jól skálázódik.
- Hátrányok:
  - A tárolható adatok mérete korlátozott
    - adatmennyiségre nem jól skálázódik.
  - Az adatok mindig utaznak a hálózaton
    - sávszélesség pazarló.
  - Az adatok láthatóak egy MITM támadó számára
    - nem biztonságos.

Állapot tárolási lehetőségek (kliens -> szerver)

- URL paraméterben.

  `http://www.aut.bme.hu?page=2`

- Rejtett mezőben.

  `<input type="hidden" name="id" value="2">`

- **Cookie**-ban.
- Local storage és session storage.
- IndexedDB.
- File system.
  - Csak egy virtuális fájlrendszert használhatunk, nem a ténylegest.

Mire kell figyelni? (kliens oldalon)

- Támadó látja az adatokat (eavesdropping)
  - Megoldás: HTTPS.
- Támadó megváltoztathatja az adatokat (tampering)
  - Megoldás: digitális aláírás.
- Az adatok elveszhetnek (pl. elszáll a böngésző vagy a felhasználó manuálisan törli az adatokat)
  - Megoldás: fallback mechanizmus.
- Korlátozott méretek.
- Számos esetben kliens oldalon külön gondoskodni kell arról, hogy eljusson a szerverre a szükséges információ. (Pl: storage-ben tárolt adatok)

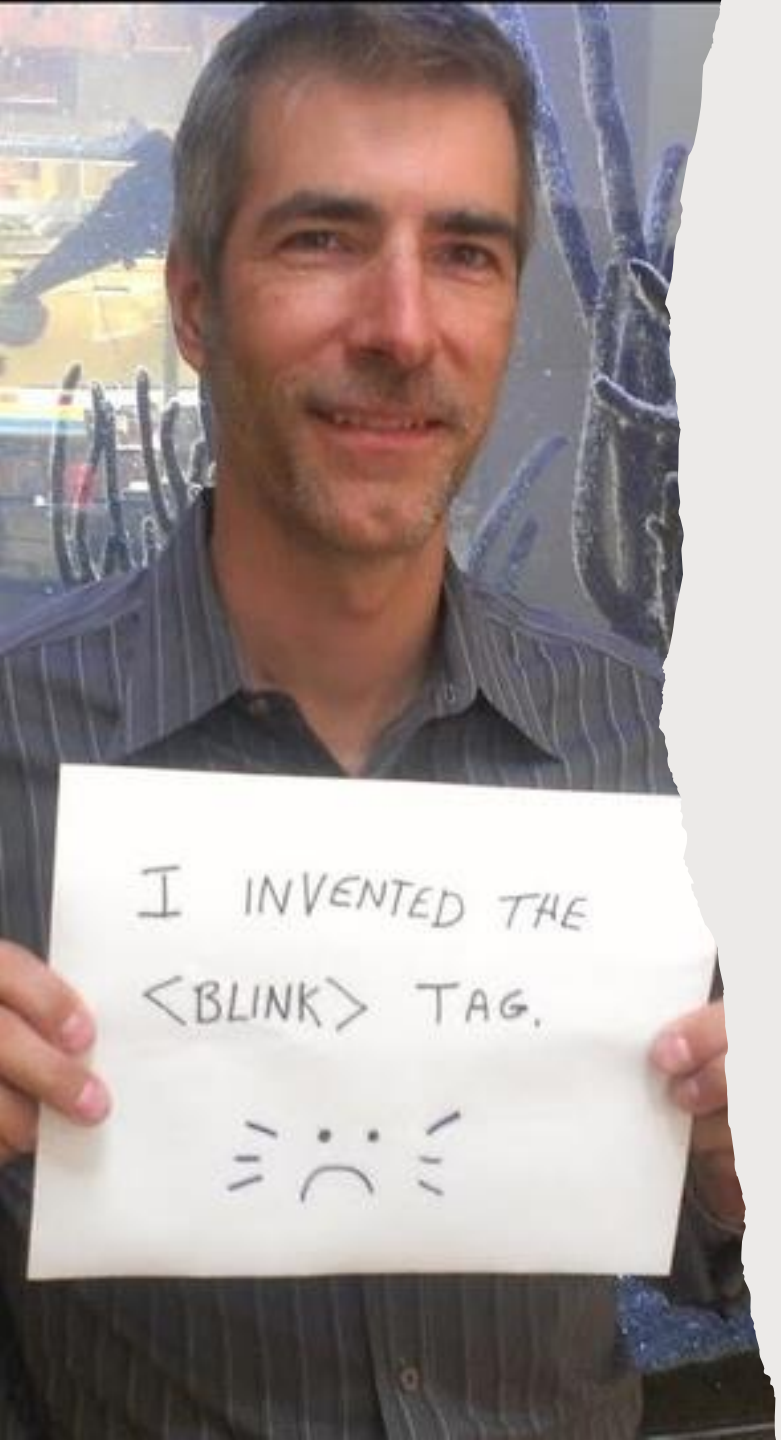
Megoldási lehetőségek: kliens + szerver

A munkamenethez tartozó **információk a szerveren tárolódnak**, csak a munkamenet azonosítója (session ID) utazik a böngésző és a szerver között.

- Előny: ami a kliensoldali megoldásnál hátrány volt.
- Hátrány:
  - Memória igény
    - sok felhasználóra nem jól skálázódik.
  - Szerver farm esetén
    - vagy intelligens terheléselosztás (server affinity) kell,
    - vagy state server → single point of failure.

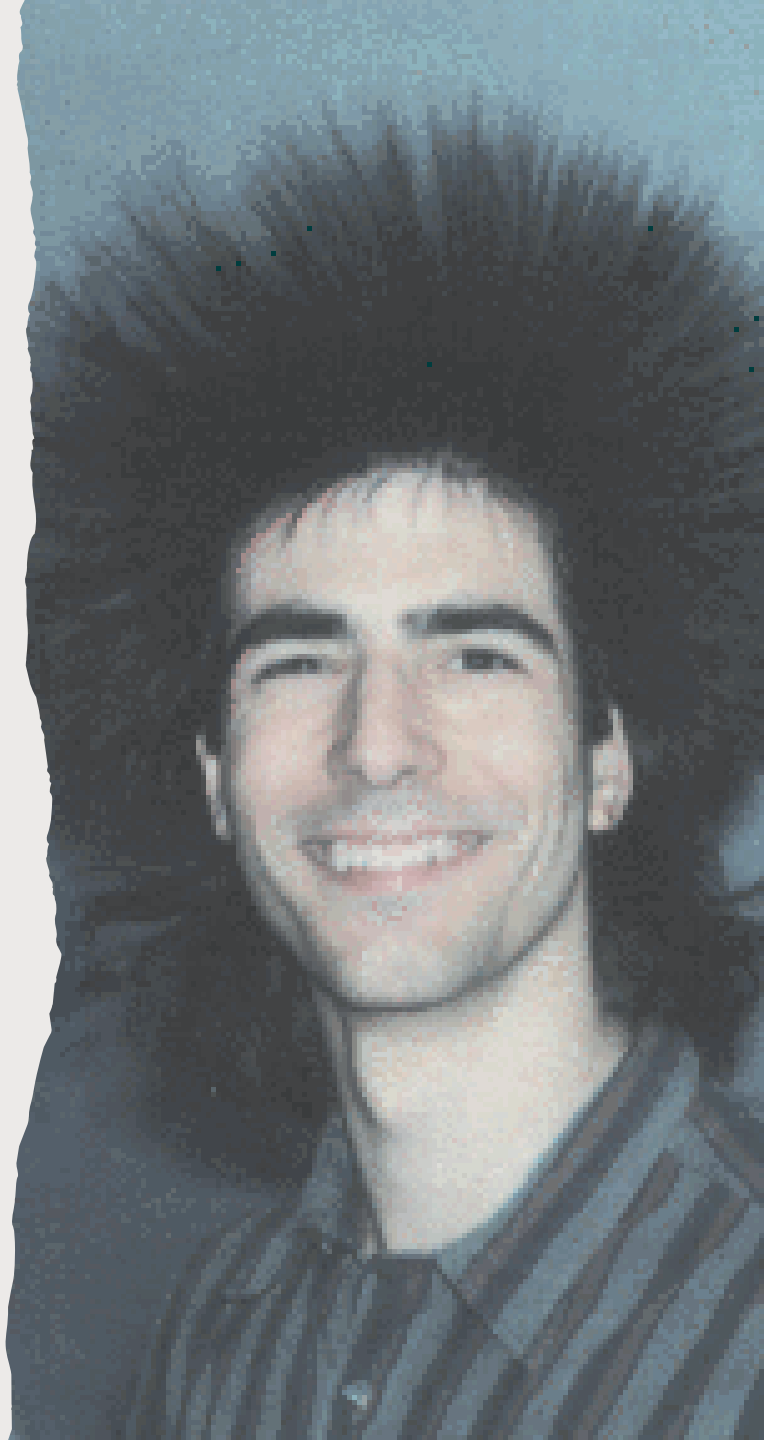
Mire kell figyelni? (szerver oldal)

- Skálázódási problémák sok felhasználó esetén
  - nehéz tesztelni.
- A webalkalmazás vagy a webszerver bármikor újraindulhat
  - pl. process crash, OS upgrade
- Szerver farm esetén a terheléselosztás problémái
  - server affinity vagy state server.



Lou Montulli

- 1991-ben Lynx böngésző
- BLINK tag
- animáló GIF
- Második webkamera.



http://www.montulli-blog.com/2013/05/the-reasoning-behind-web-cookies.html

Cookie

RFC 6265: **HTTP State Management Mechanism**
(2011. április)

- Cél: memória a HTTP-hez.
- Ötlet volt, hogy a böngészőknek legyen egyedi azonosítója
  - de akkor nyomonkövethetők a felhasználók → elvetették.
- SessionId készítése, leküldése a böngészőnek, amit az visszaküld mindig a szervernek.
- Úgy készüljön, hogy cross site tracking-et ne engedje.
- A mai süti az akkori koncepció kb 95%-át megtartotta.
- A sütibe kerülhet egy random sessionID, username, és bármi, csak ne legyen nagy!
- Törlődjenek
  - ha bezárják a böngészőt.

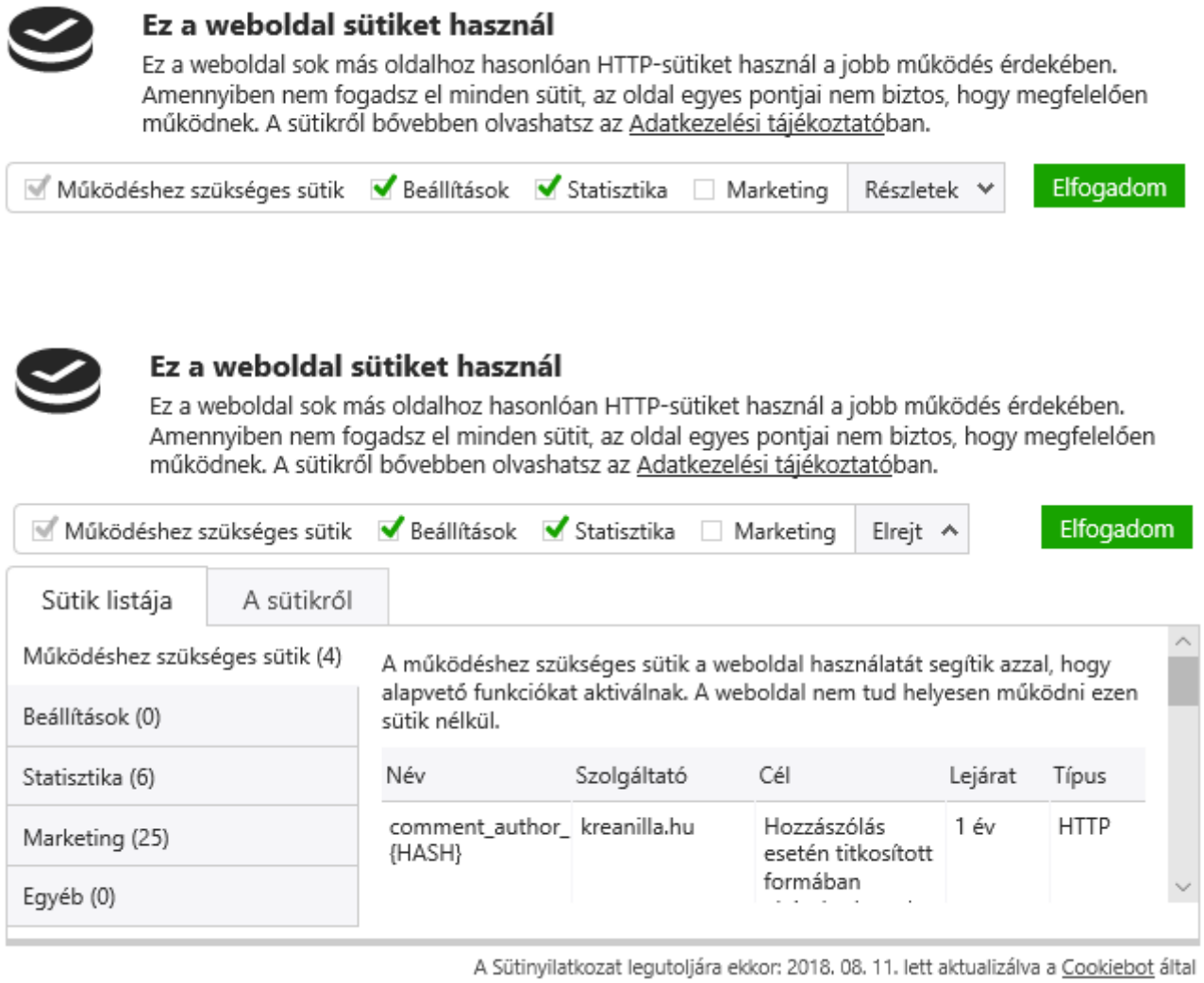
3rd party cookie

- Probléma lett 1996 körül
  - alapvetően a sütit nem arra tervezték, hogy nyomon kövesse a felhasználót oldalakon keresztül
- Két megoldás merült fel megoldási lehetőség
  - Továbbra is engedélyezni a 3rd party sütiket
    - így láthatók maradnak a cégek, akik nyomon követik a felhasználói szokásokat
    - kormányzat az adatgyűjtést tudja szabályozni (ha akarja).
  - Vagy letiltani az egészet
    - akkor majd kitalálnak valami mást, amit lehet, hogy észre sem veszünk.

GDPR cookie szabályozás EU-ban

- A weboldal látogatóit figyelmeztetni kell, hogy
  - az oldal sütiket használ
  - milyen típusú sütiket használ
    - Oldal működéséhez szükséges
    - Statisztika gyűjtésre
    - Beállítások mentésére
    - Marketing célokra
    - ...
- A weboldal látogatóinak lehetőséget kell adni arra, hogy kikapcsolhassák az egyes típusú sütik használatát.

Figyelmeztetés süti használatra



Cookie típusai

- **Session (in-memory/transient) cookie**
  - Csak a munkamenet idejére létezik, a böngésző bezárásával törlődik.
  - Több böngésző ablak / tabfül osztozik rajta.
- **Permanent (persistent) cookie**
  - diszkre mentődik.
  - „Remember me” checkbox a bejelentkező oldalakon.

Cookie tartalma

Szöveges tartalom, nem futtatható, de privacy problémát jelenthet.

- **Name**: süti neve, ezzel tudunk rá hivatkozni.
- **Value**: az eltárolt érték string formátumban.
- **Expiration date**: Süti lejárai ideje
- **Path**: URL-ben minek kell szerepelnie, hogy elküldje a sütit a böngésző.
  - Alapértelmezés szerint: "/"
- **Domain**: Melyik hostokra kell elküldeni.
  - Ha nincs megadva, akkor ahonnan letöltöttük az oldalt (subdomainek nélkül)
- **Secure**: Csak HTTPS-en keresztül használható.
- **HttpOnly**: Kapcsoló, hogy ne lehessen JS-ből módosítani.

Sütihez kapcsolódó HTTP fejlécek

- **Set-Cookie**: Süti létrehozása
- **Cookie**: Szervertől kapott sütik listája

- Cookie törlésére nincs külön fejléc
  - felülírás üres tartalommal és elmúlt lejárati dátummal.
- A böngésző minden alkalommal visszaküldi a szerverre ha a domain és path egyezik.
  - Akkor is, ha az adott HTTP kéréshez nem kellene
    - pl. CSS → cookieless domain.

Biztonság

- Nyílt szövegként utazik
  - tartalom titkosítása, HTTPS + **Secure** flag beállítása
- Nyílt szövegként tárolhatja a kliens (privacy)
  - tartalom titkosítása
- Változtatható a tartalma
  - integritás ellenőrzés, digitális aláírás, HMAC
- Nem garantálható az eredete
  - Nem csak a szerver hozhatja létre
    - __Host- cookie prefix használata, digitális aláírás.
  - Máshonnan is visszaküldhető (session hijacking)

Biztonság

- Script hozzáférhet és módosíthatja
  - XSS (Cross-site scripting) támadás → **HttpOnly** flag
- A perzisztens süti a böngésző bezárása után, a felhasználó akarata ellenére is eljuthat a szerverre.
  - XSRF - Cross-site request forgery támadás.
    - SameSite attribútum használata. (Strict, Lax, ~~None~~)
- Cookie store:
  - Több böngésző esetén több cookie store.
  - Több operációs rendszer felhasználó esetén több store.
  - „Private browsing”

Biztonság: HTTPS

Kliensalkalmazások

BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Gincsai Gábor
gincsai@aut.bme.hu

HTTPS

- A HTTP kapcsolat titkosítatlan
  - HTTP + SSL (Secure Sockets Layer protocol)
  - HTTP + TLS (Transport Layer Security)
- Nem önálló protokoll („HTTP over SSL”)
- Port: 443
- **https://** URI séma
- SSL/TLS felett más protokoll is mehet
  - Mivel alsóbb rétegben működik, mint a HTTP, ezért a hostname alapú virtual hosting megoldásokat önmagában nem támogatja.

Funkciók

- **Szerver azonosítása** (server authentication): pontosan kivel áll kapcsolatban a kliens.
  - Képes a kliens azonosítására (mutual authentication) is, de az általában nem használatos.
- **Kommunikáció titkosítása** (encryption): harmadik fél nem tudja lehallgatni (eavesdropping).
- **Tartalom integritása** (integrity): harmadik fél nem tudja megváltoztatni (tampering).
- Oldalon összes hivatkozásnak **https://**-nek kell lennie, különben: mixed content warning.

Tanúsítvány (certificate)

Alapelv: egy megbízható harmadik fél (trusted 3rd party) igazolja a szerver hitelességét.

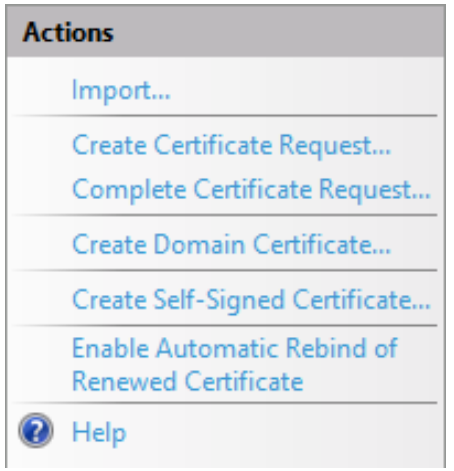
- **Tanúsítvány (X.509 certificate)**
- **Tanúsítvány lánc (certificate chain, chain of trust)**
- **Certification Authority** (CA): tanúsítvány kiadó
  - Subordinate CA (intermediate CA)
  - Root Certification Authority (Root CA)

A tanúsítvány részei

- **Signature algorithm**: Aláíráshoz használt algoritmus
- **Issuer**: Kiállító
- **Valid from, Valid to**: Mettől meddig érvényes.
- **Subject**: Kinek állították ki.
- **Public key**: Maga a publikus kulcs. Pl a 4096 bites RSA kulcs.
- **Thumbprint**: A tanúsítvány lenyomta, ez alapján is kereshető
- **Subject Alternative Name (SAN):** További FQDN melyekre a tanúsítvány érvényes lesz.
  - Pl. www.digicert.com és digicert.com

Tanúsítvány készítés folyamata

1. Kérelem (Certificate Signing Request) összeállítása
   - Kulcspár generálás a szerveren (privát és publikus) openssl-lel
     - A kérelemben csak a publikus kulcs szerepel!
   - Szervezeti egység adatainak megadása
   - CN beállítása
   - Egyéb adatok beállítása (pl: signiture algorithm, SAN....)
   - IIS Manager → Server Certificates → Create Certificate Request...
2. Generált kérelem feltöltése a CA-hoz
3. CA-tól visszakapott adatokkal befejezni a Certificate igénylési folyamatot a szerveren.
   - Ott lehet befejezni, ahol a kérelmet indítottuk. Csak ott van meg a privát kulcs.
4. Tanúsítvány website-hoz adása és HTTPS binding létrehozása.

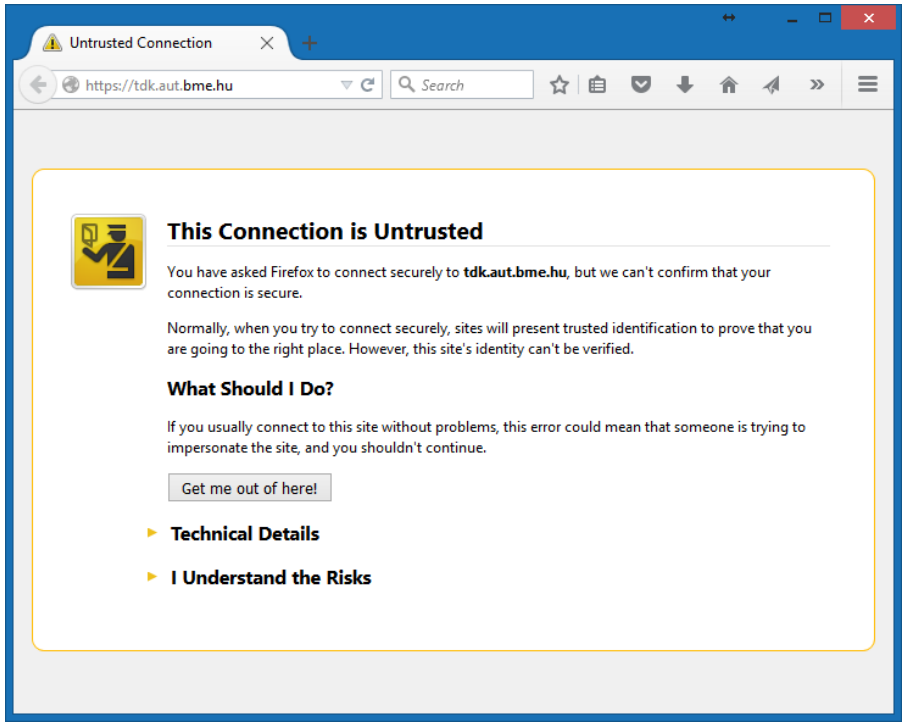


A tanúsítvány privát kulcsa

- A privát kulcs
  - nem része a tanúsítványnak.
  - lehet jelszóval védett
  - lehet exportálható vagy nem exportálható.
- A webszerveren keletkezik és nem jut el a CA-hoz.
- A CA csak azt igazolja, hogy a nyilvános kulcs az adott tulajdonosé.
- Fájl formátumok
  - **.pem, .cer, .crt, .der, .p7b, .p7c, .p12, .pfx**

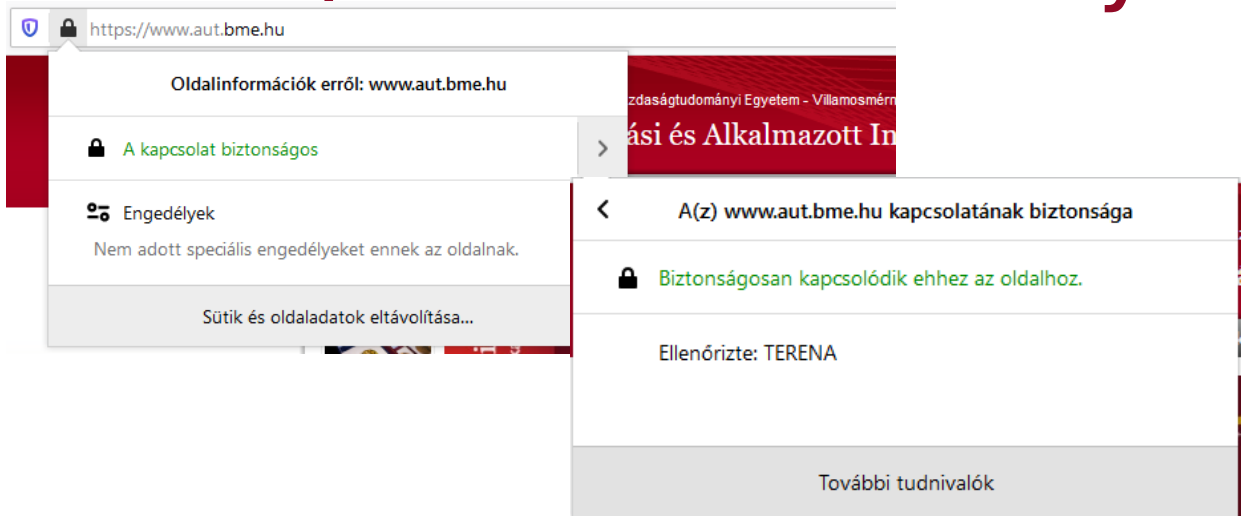
Önaláírt tanúsítvány (self-signed certificate)

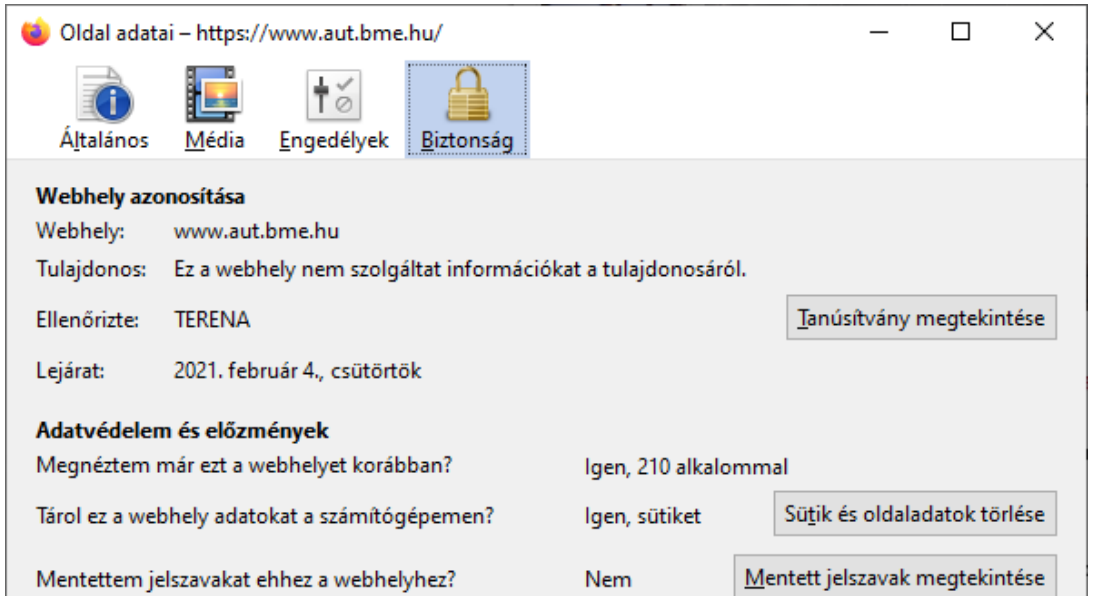
- Előnyök
  - Olcsó
  - Gyors
  - Tetszőlegesen testreszabható
  - Titkosítja a kapcsolatot

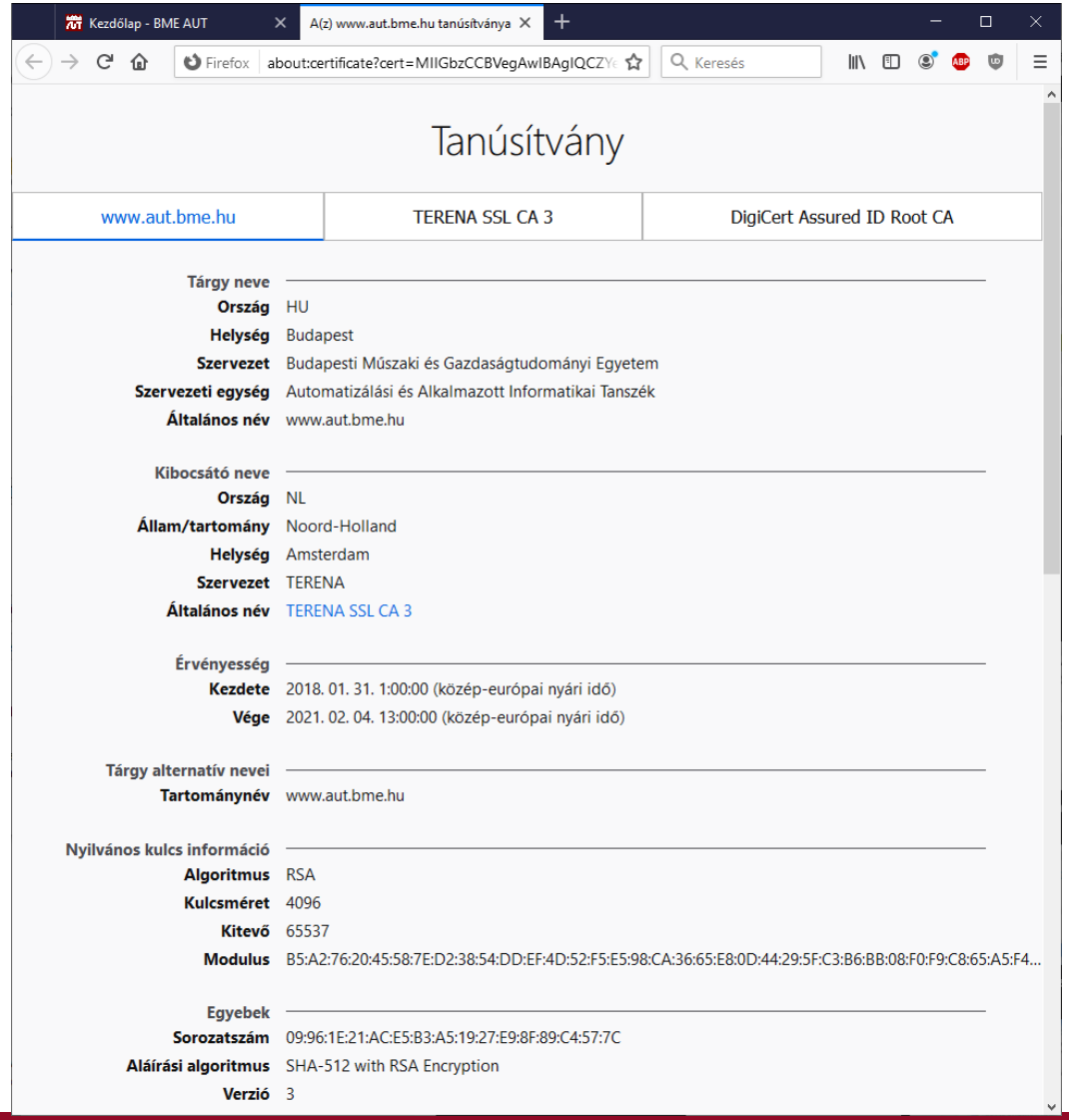


- Hátrányok
  - Nem azonosítja a szervert.
  - Man-in-the-middle támadással könnyen kijátszható, ezért nem ajánlott.
  - A felhasználókat arra tanítja, hogy elfogadják a nem hiteles tanúsítványt.

AUT portál tanúsítványa Firefox





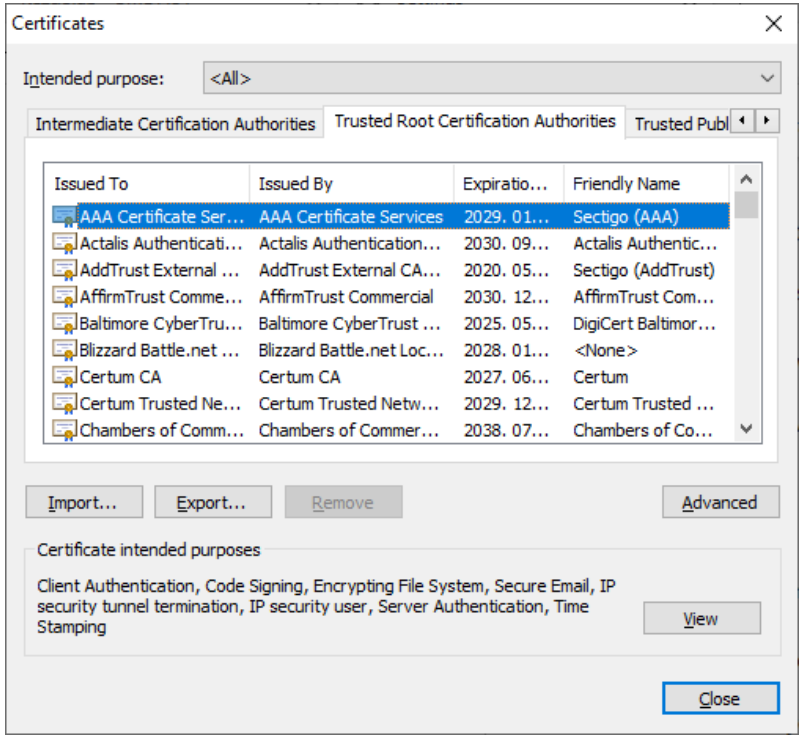


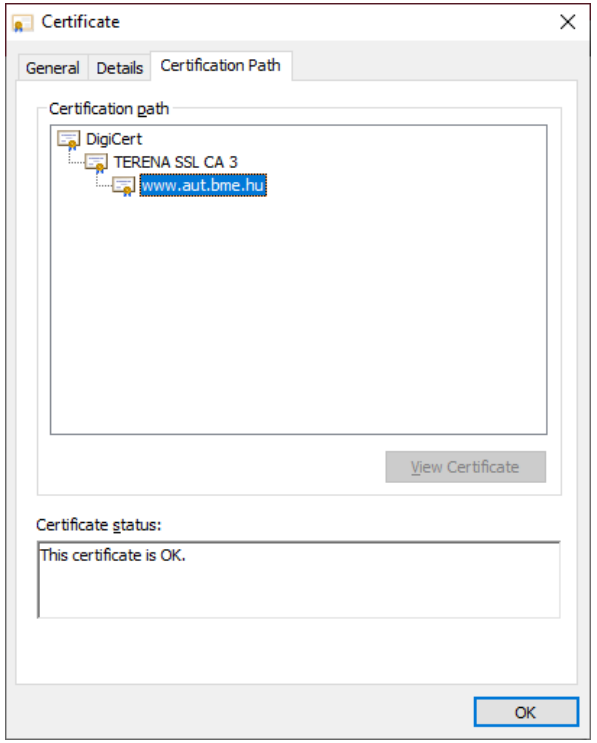
Egy tanúsítvány érvényes, ha

1. A kiállító hiteles.
2. Nem járt le.
3. Az aktuális szerver számára állították ki.
4. Nem vonták vissza.

1. A kiállító hiteles

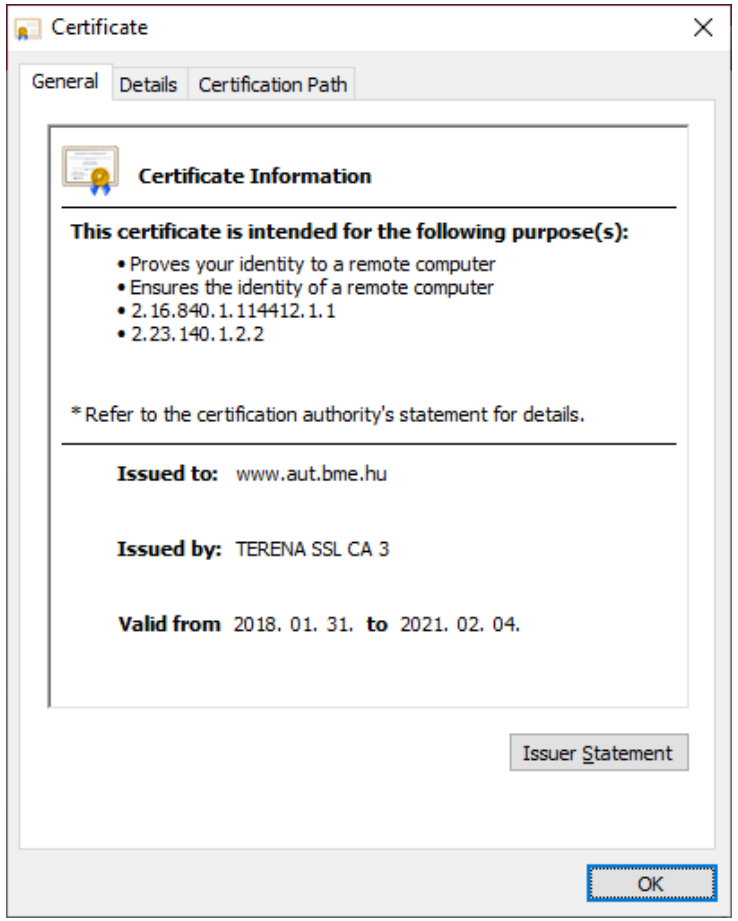
- A böngészőnek meg kell bíznia a CA lánc minden szereplőjében.
- A gyökér tanúsítvány kiadónak szerepelnie kell a böngésző Trusted Root CAs listájában.





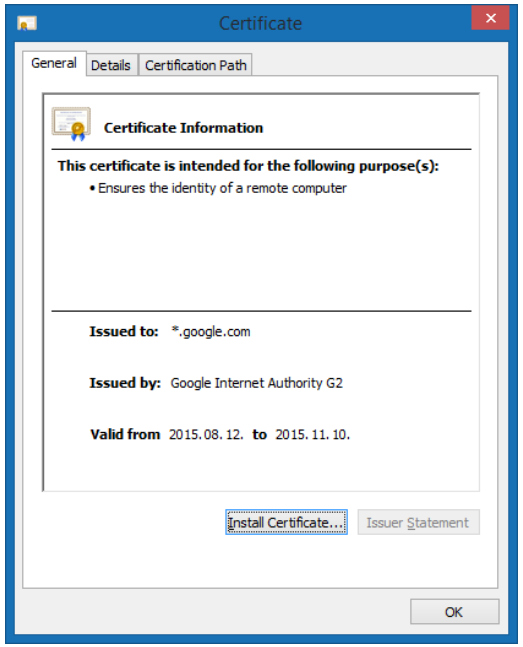
2. Nem járt le

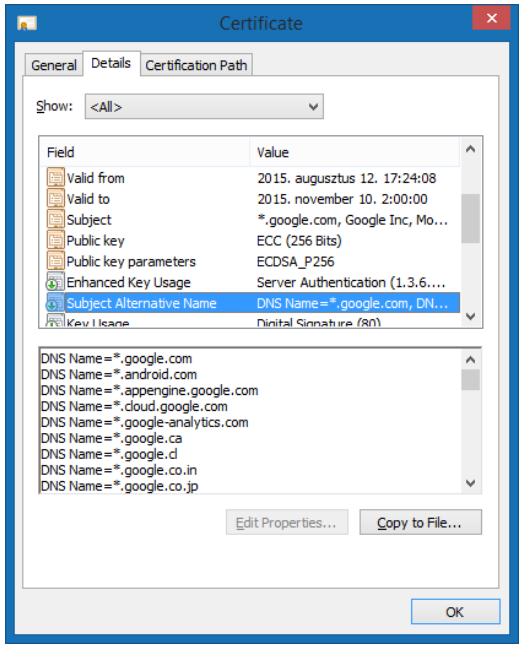
- A tanúsítványok érvényességi ideje általában 1-3 év.



3. Az aktuális szerver számára állították ki.

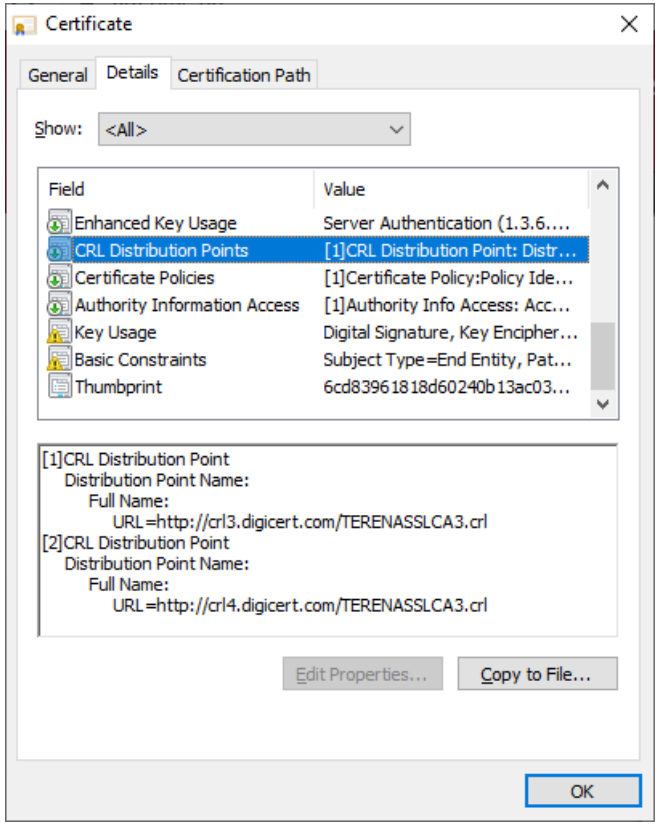
- A tanúsítvány Subject mezőjében lévő CN-nek meg kell egyeznie az oldal betöltéséhez használt FQDN-nel.
  - > https://example.com != https://www.example.com
- SAN-ben megadható több, alias FQDN is.
- Wildcard certificate: ***.example.com**
  - > Több al-domainhez
  - > Csak 1 szint mélységig
  - > Extended Validation nem támogatja





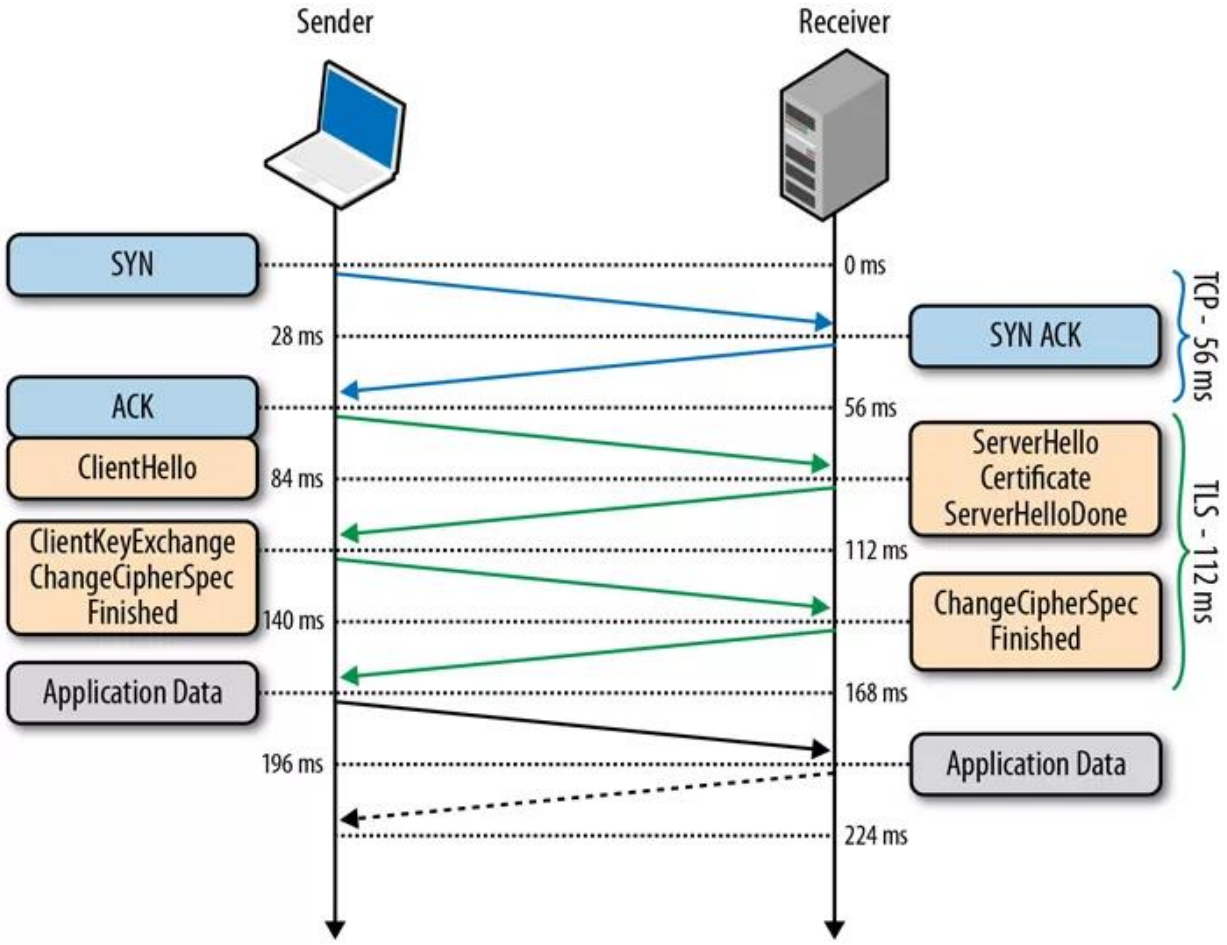
4. Nem vonták vissza.

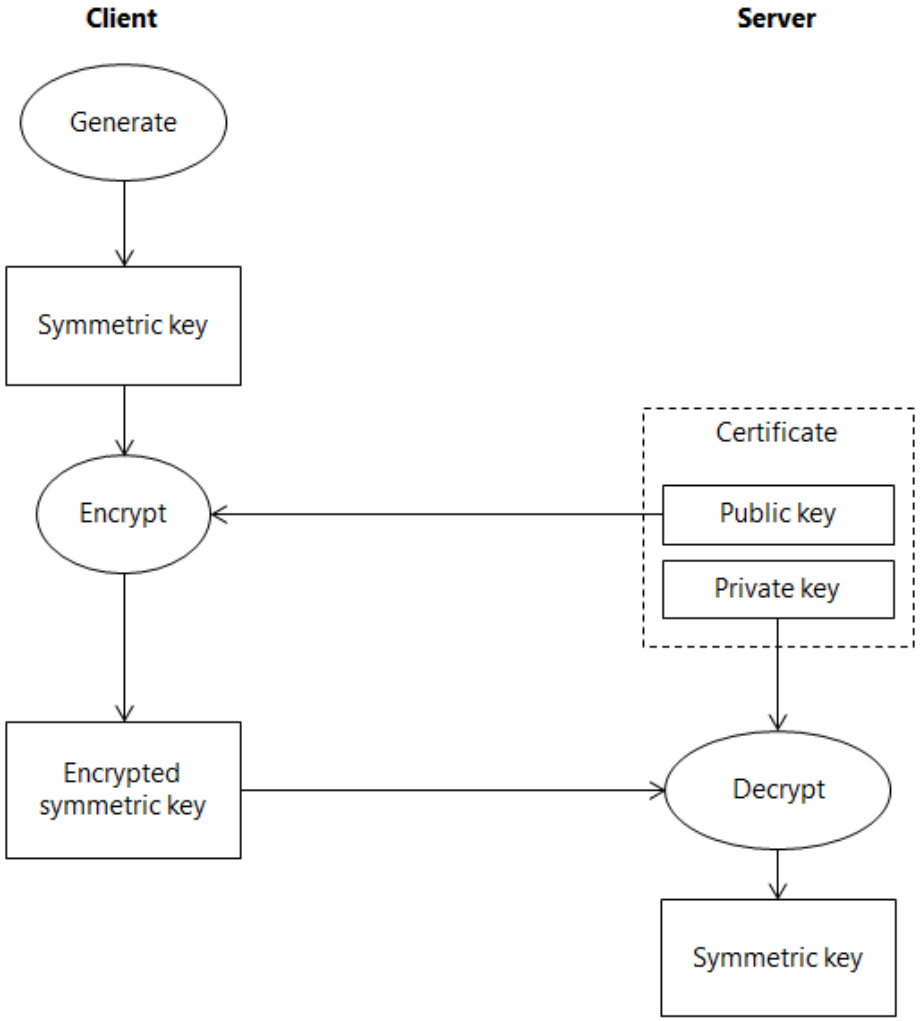
- A tanúsítvány vagy a CA kompromittálódhat.
  - > 2011. március: iráni hekkerek Comodo és DigiNoar tanúsítványokkal man-in-the-middle támadásokat hajtottak végre.
- Certificate Revocation List
  - > Aláírt, TTL-lel (24 óra) ellátott, nyilvános lista.
  - > A tanúsítványban lévő CRL Distribution Point határozza meg az URL-t.
- Online Certificate Status Protocol (OCSP, RFC 6960)
  - > Egy tanúsítvány státuszának lekérdezésére a CA-tól.
  - > A kliensnek nem kell a teljes CRL-t feldolgoznia.



https://www.ssl.com/faqs/compromised-private-keys/

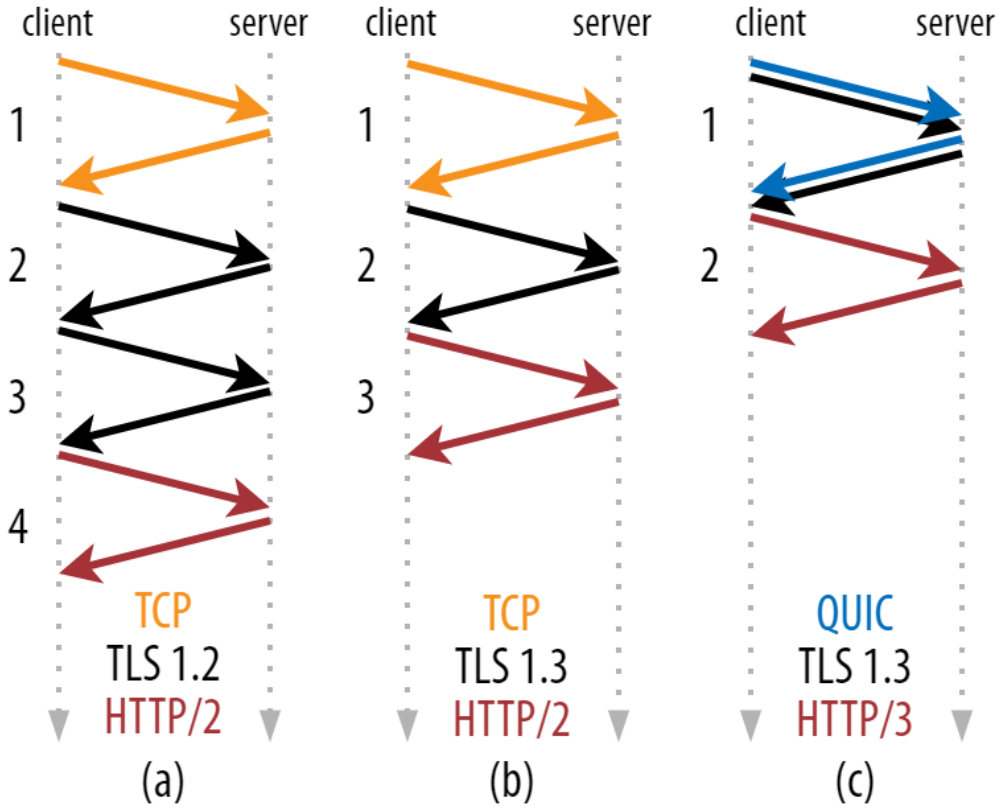
TLS handshake és a kulcscsere folyamata





HTTP protokollok és a TLS

- TLS 1.2
  - 2 network roundtrip
- TLS 1.3
  - 1 network roundtrip
  - Szigorúan korlátozza a különböző titkosítási algoritmusokat. Így a kliens azonal kitalálhatja, hogy a szerver melyiket fogja támogatni.
- HTTP/3 QUIC
  - HTTP + kriptográfiai handshake egyben.
  - A QUIC a TLS 1.3-at magába foglalja. Így nincs mód a QUIC használatára TLS nélkül.
  - A QUIC (és kiterjesztve a HTTP/3) mindig teljesen titkosított



Rövid ellenőrző lista

- Minden érzékeny adat HTTPS-en megy.
- HTTPS oldalakra nem kerül HTTP tartalom.
- Authentikációs sütik
  - nem mennek HTTP-n.
  - a Secure flag be van állítva.
- A bejelentkező oldalak is HTTPS-en mennek.

IIS-ben a HTTPS beállítása

- IIS Manager
  - Server Certificates
  - Ha nincs itt fel kell telepíteni

- Site binding-nál
  - Https binding kiválasztása
  - Certificate kiválasztása
  - Server Name Indication
    - Ha Https-es több weblap fut a szerveren