



# Webfejlesztés alapjai

**KnowCamp**

# SZERVEROLDALI FEJLESZTÉS

**Ismétlés**

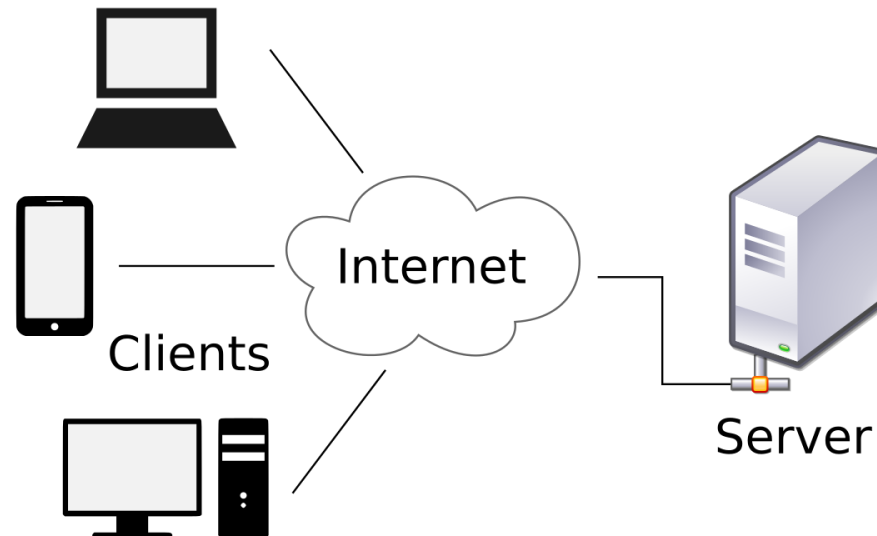
**Kliensoldali és szerveroldali programozás**

**Szoftverfejlesztés**

**Minta alkalmazás bemutatása**

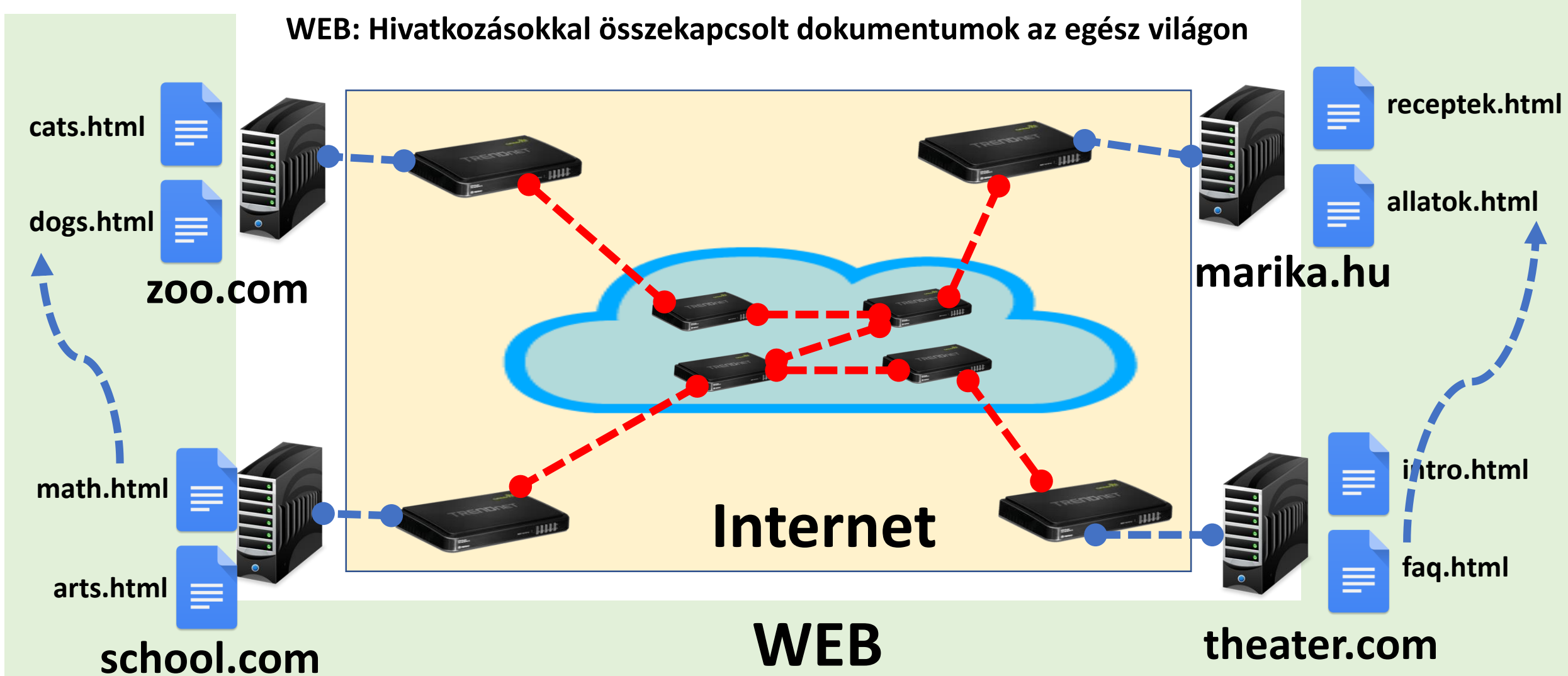
**Egyéb programozási nyelvek**

- **Internetezés:** Távoli számítógépekkel (kiszolgálókkal) való információcsere.
- **Kliens:** Az adatot kérő felhasználó számítógépe
- **Szerver:** Az adatot biztosító távoli számítógép



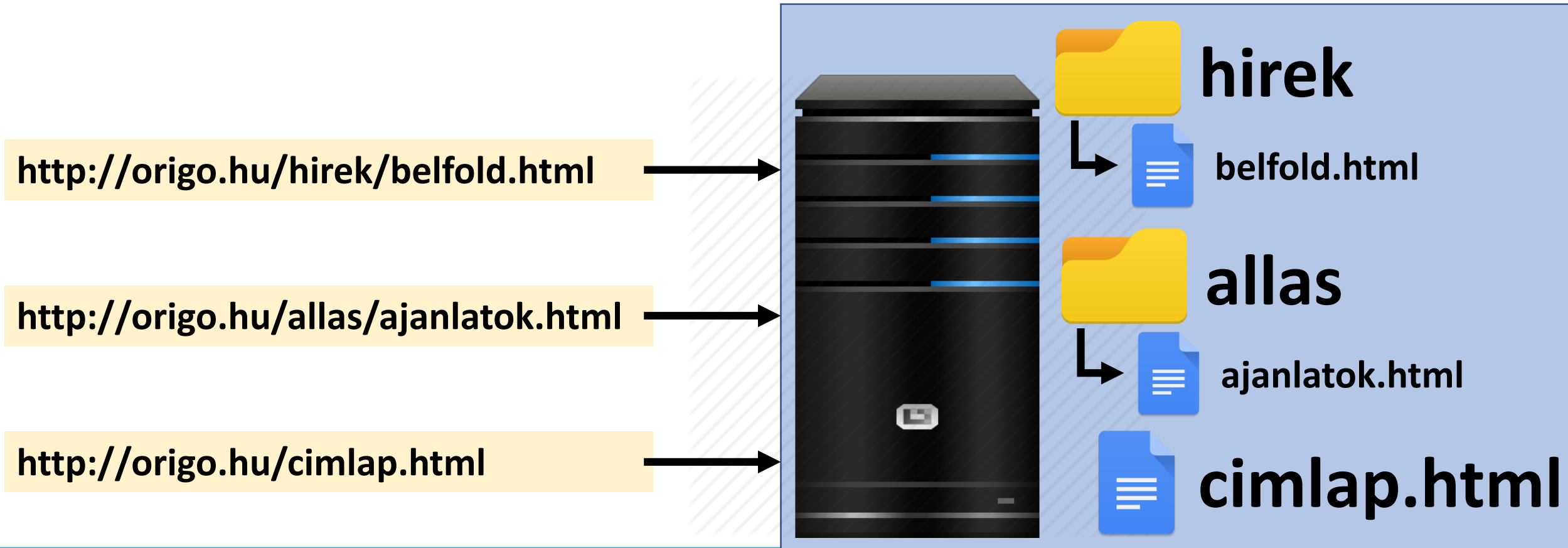
- **Internet:** Hálózatok hálózata, egymással összeköttetésben lévő számítógép hálózatok (mindenki el tud juttatni elektronikus adatot a másikhöz)
- **Web:** Egymással összekapcsolt dokumentumok összessége
- **Web 3 szabványa**
  - **URL** (Uniform Resource Locator)
  - **HTML** (HyperText Markup Language)
  - **HTTP** (HyperText Transfer Protocol)

WEB: Hivatkozásokkal összekapcsolt dokumentumok az egész világon



- **URL (Uniform Resource Locator)**

- Az összetartozó dokumentumok egyedi címmel való ellátása
- Hierarchia szervezése a dokumentumokból



- **HTML (Hyper Text Markup Language)**

- Séma a dokumentumok elkészítésére
- Egységes, szabványos leíró nyelv

eredmeny.html  
tartalma

```
<table border = "1">  
  <tr>  
    <th>I. hely</th>  
    <th>II. hely</th>  
    <th>III. hely</th>  
  </tr>  
  <tr>  
    <td>Béla</td>  
    <td>Gizi</td>  
    <td>Saci</td>  
  </tr>  
</table>
```



I. hely	II. hely	III. hely
Béla	Gizi	Saci

Ezt látjuk a böngészőben

- **HTTP (Hyper Text Transfer Protocol)**

- A weboldalak tartalmának átvitelét biztosító protokoll
- Kérés-válasz alapú, **mindig a kliens kezdeményez** valamit, a szerver csak válaszol
- Küldünk egy parancsot és paramétereket (adatok), kapunk egy választ (adatok)

```
GET http://users.nik.uni-obuda.hu/sima/201617springnotesPRAMSc.htm
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/we
Referer: http://users.nik.uni-obuda.hu/sima/oktatas.htm
Accept-Encoding: gzip, deflate
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: SESS5271c37b9aa6f51d903b4e8f7292bcf0=cmcp0o54c97moao5ve9a79v77
```

```
HTTP/1.1 200 OK
Date: Tue, 22 Aug 2017 16:23:52 GMT
Server: Apache
Last-Modified: Mon, 08 May 2017 05:48:50 GMT
ETag: "7a6024-af4-54efccb1db480"
Accept-Ranges: bytes
Content-Length: 2804
Keep-Alive: timeout=15, max=96
Connection: Keep-Alive
Content-Type: text/html
```

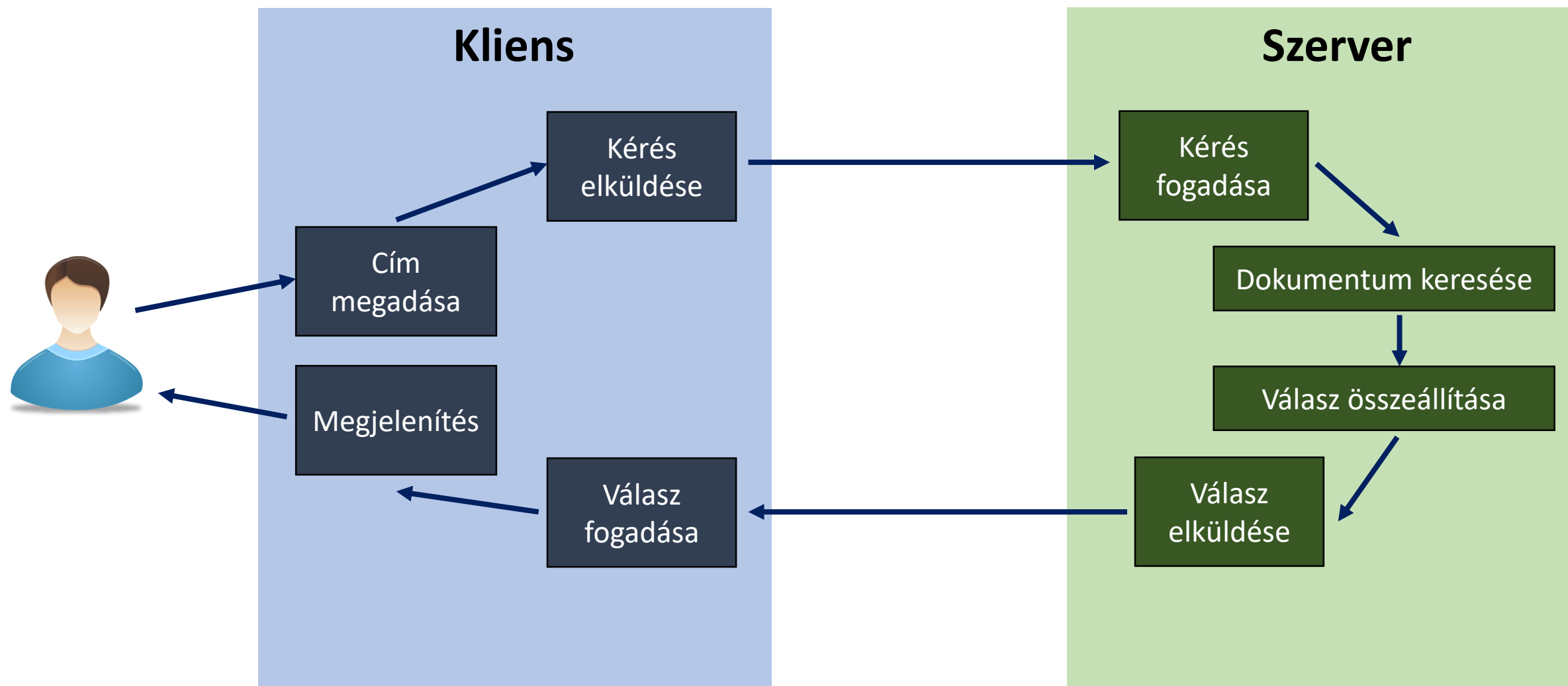
## HTTP kérés

- Dokumentum neve
- Elvárt formátum
- Előzőleg látogatott oldal
- Elvárt nyelv

## HTTP válasz

- Dátum
- Utolsó módosítás
- Hossz
- Formátum
- Maga a dokumentum törzse



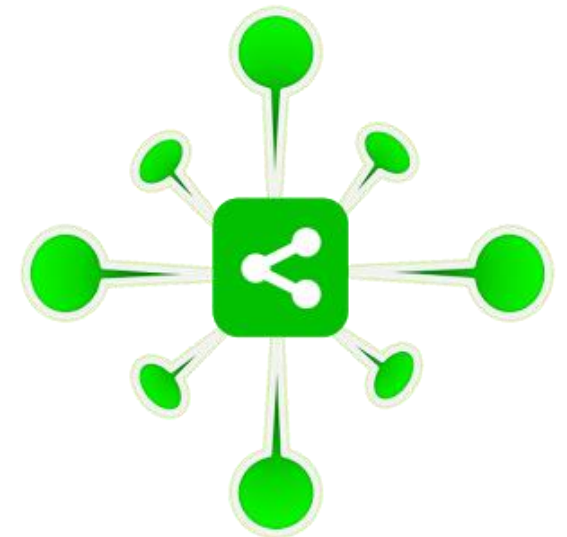


- **HTML eddigi ismereteink**

- Létrehozunk a számítógépen egy szöveges állományt tetszőleges szerkesztőben (notepad, notepad++, sublime, VScode)
- HTML struktúrával alakítjuk ki az állományt
- Elmentjük .html formátumban
- Megnyitjuk tetszőleges böngészővel (google chrome, mozilla firefox, microsoft edge, opera, safari, stb.)

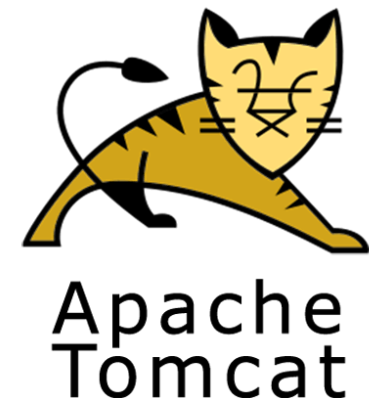
- **Miért kell nekünk akkor szerver?**

- Amit így létrehoztunk, csak mi látjuk a saját gépen
- Meg kellene osztani másokkal is



- Megosztáshoz **webservert szoftvert** telepítünk a számítógépre
- Működésének fázisai
  1. HTTP kérést fogad
  2. Kérésből megtudja, hogy a kliens mely dokumentumot kéri
  3. Kikeresi a dokumentumot
  4. Tartalmát HTTP válaszként továbbítja
- Egyéb szolgáltatások, amiket nyújthat
  - AAA (Authentication-Authorization-Accounting)
  - Titkosítás
  - IP címek tiltása
  - Megosztási szabályok
  - Egyéb programok futtatása

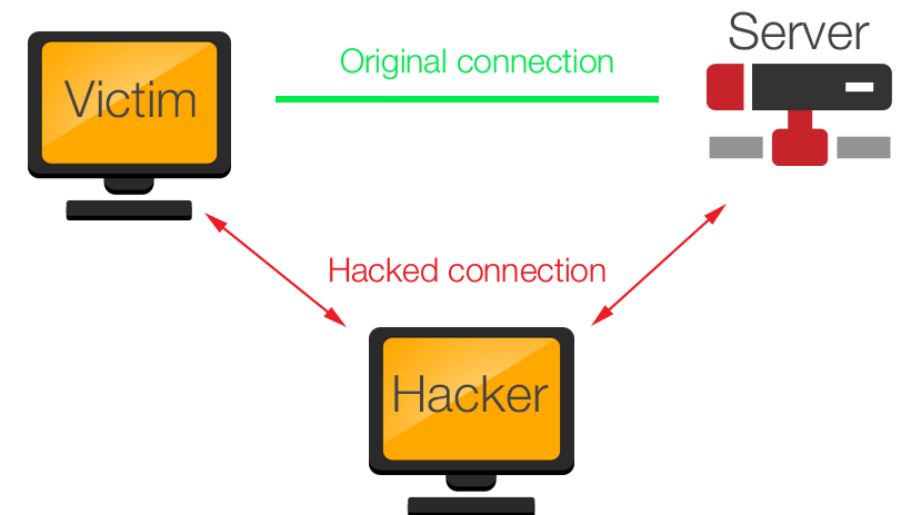




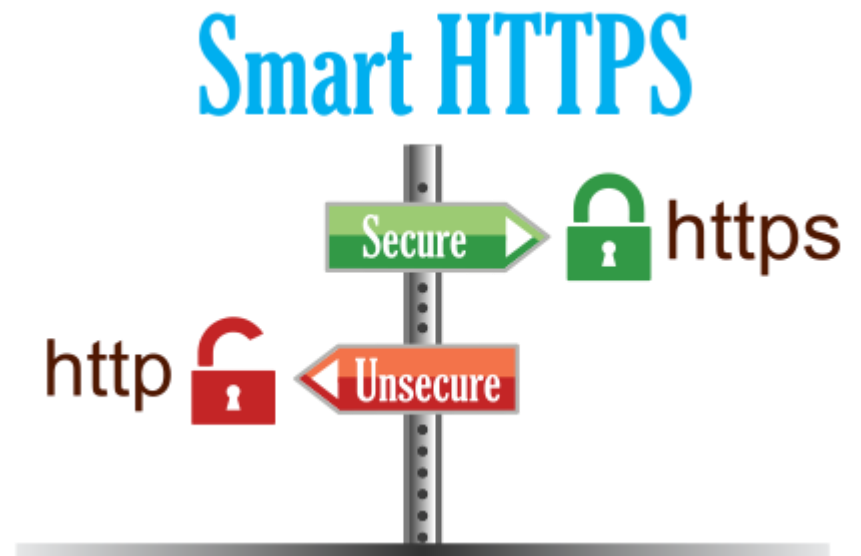
- Egy biztonsági architektúra az alábbi 3 szolgáltatással
  - **Authentication (azonosítás)**
    - Felhasználónévvel és jelszóval történő azonosítás (esetleg digitális aláírás, token, stb.)
  - **Authorization (jogosultságkezelés)**
    - Mely felhasználó mely erőforrásokhoz férhet hozzá
  - **Accounting (könyvelés)**
    - Ki, mikor és milyen erőforrást használt



- A HTTP forgalom **szövegalapú** és **titkosítatlan**.
- Mi ezzel a probléma?
  - Könnyűszerrel lehallgatható!
- HUB eszköz mindenkinek küldi az üzeneteket, de csak a címzett reagál rá
- Switch lehallgatásra példa (ARP Poisoning)
  - A kiterjesztett csillag topológiában tipikusan minden számítógép (kliens vagy szerver) hálózati switch-hez csatlakozik
  - A switch az üzeneteket MAC cím alapján továbbítja a címzethez
  - Egyszerű segédprogrammal átverhető a switch, a támadó a címzett személyében tudja feltűntetni magát



- **HTTPS:** A HTTP protokollt használja, de igénybe veszi az SSL/TLS biztonsági alrendszert.
- **Titkosítás működése**
  - A kliensnek és szervernek is vagy egyaránt egy **publikus** és egy **privát** kulcsa (egy szám)
  - A **publikus** kulcsot megosztják egymással
  - A **publikus** és a **privát** kulcs együtt kulcspárt alkot
  - Amit a **publikussal** kódolunk, csak a **privát** tudja dekódolni
  - Lépések:
    - A kliens a szerver **publikus** kulcsával kódolja az üzenetet
    - A szerver a **privát** kulcsával dekódolja
    - A szerver a kliens **publikus** kulcsával kódolja a választ
    - A kliens a **privát** kulcsával dekódolja
  - **Privát** kulcs kitalálható próbálgatással, ezért ma már nagyon nagy számot választanak (pl:  $2^{1024}$  nagyságrend)



- Könnyedén beállíthatjuk, hogy valamely IP cím hozzáférhet-e a dokumentumokhoz, vagy sem.
- IP cím tartományok is megadhatók
- **IP Whitelist:** Senki sem férhet hozzá, kivéve a megadottak
  - Bizonyos IP című illetők számára szeretnénk csak szolgáltatást nyújtani
  - Pl: Egyetemi könyvtár – szakdolgozatok olvasása csak az egyetem hálózatában lévő számítógépekről
  - Pl: Hallgatói gépek beállítási felülete: csak a tanári gépekről menedzselhető
- **IP Blacklist:** Mindenki hozzáférhet, kivéve a megadottak
  - Le akarunk tiltani valakit
  - Pl: Egy támadó a serverünket folyamatosan kérésekkel bombázza, hogy ellehetetlenítse a szolgáltatás működését. Letiltjuk a támadót (akár automatikusan is sorozatos egyező kérések után)



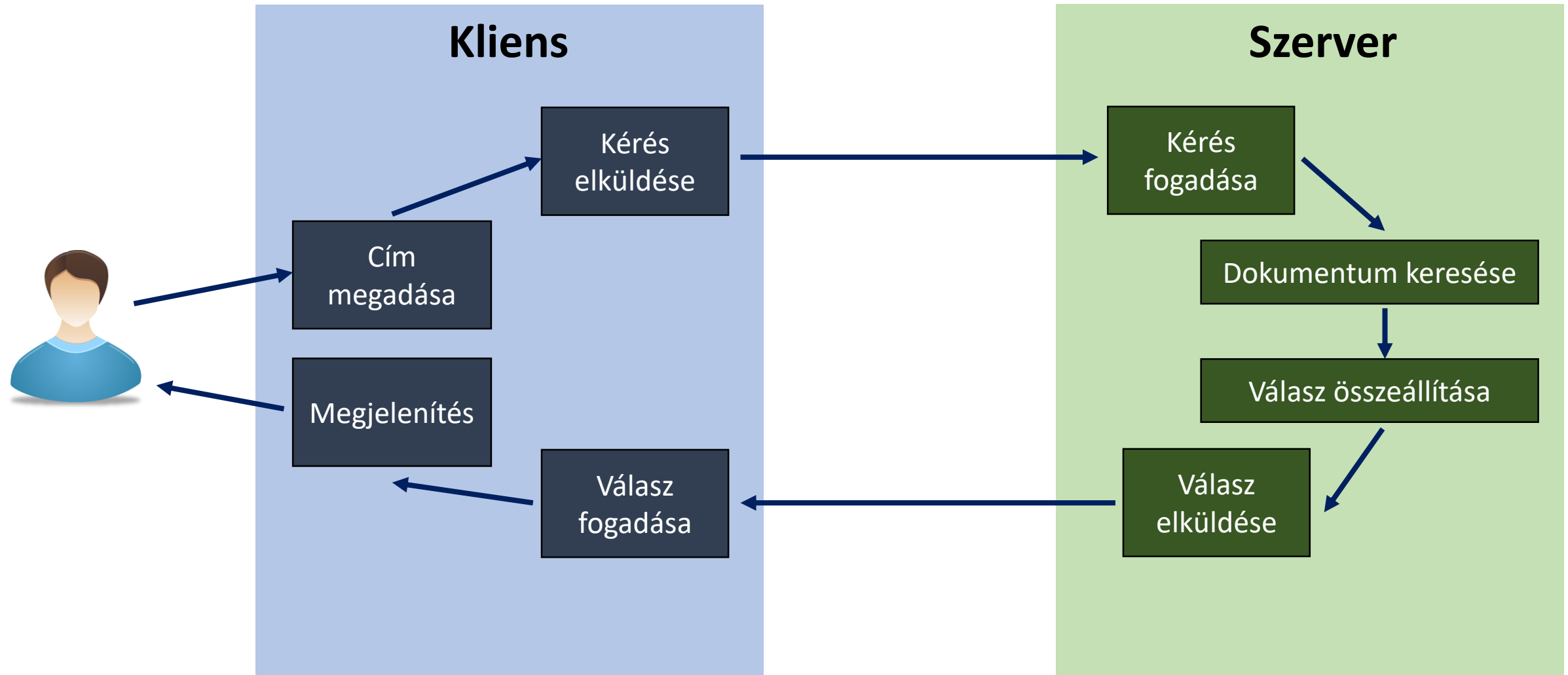


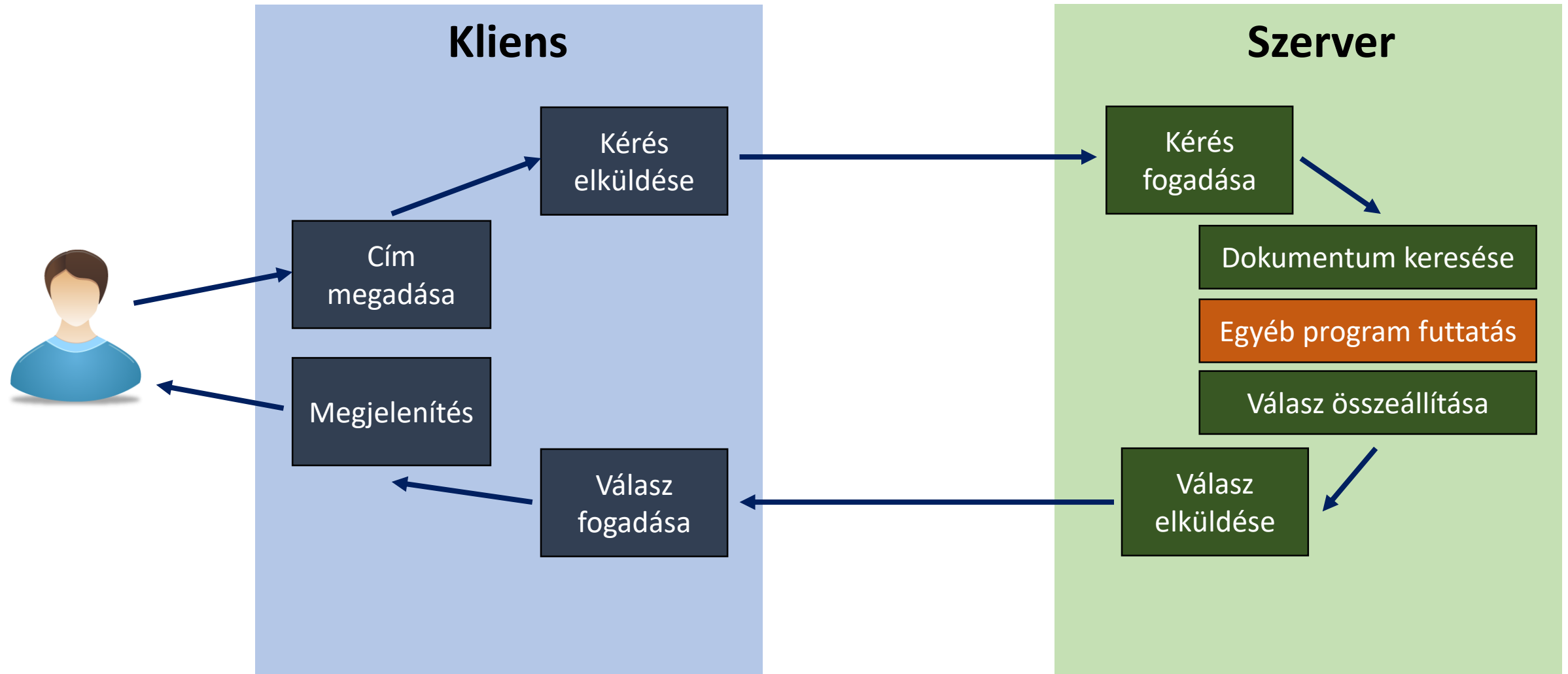
- A számítógépen/szerverszámítógépen nem feltétlenül csak a világhálón megosztani kívánt tartalmat tárolhatjuk.
- Egyéb tipikus példák:
  - Adatbázisok biztonsági mentései
  - Elektronikus levelek (levelezési szolgáltatáshoz)
  - Konfigurációs állományok
  - Privát kulcsok
  - Saját tartalmak (pl: szakdolgozat biztonsági mentése)
- A szerveren lévő minden egyes mappára pontosan beállíthatjuk, hogy részt vegyen-e a megosztásban (URL címen keresztül elérhető legyen-e)



- A webszerver szoftver adott fájlok kérése esetén két dolgot tehet:
  - **A fájl tartalmát HTTP válaszban visszaküldi** (HTML tipikus példa)
    - Egyéb példák: telepítőfájlok, videók, tömörített állományok, office dokumentumok
  - **A fájl tartalmát átadja egy tőle független programnak**, ami feldolgozza és a webszerver ennek a független programnak az eredményét küldi vissza
- Ez utóbbi módszer adja meg a lehetőségét webalkalmazások és webszolgáltatások fejlesztésének
- Miért jó ez?
  - A HTML dokumentumok segítségével **statikus** weboldalakat készíthetünk  
(pl: portfolio oldal, bemutatkozó oldal, receptgyűjtemény, stb.)
  - A mai világban **dinamikus** weboldalakat használunk minden nap
    - Felhasználói döntések függvényében más-más eredményt látunk

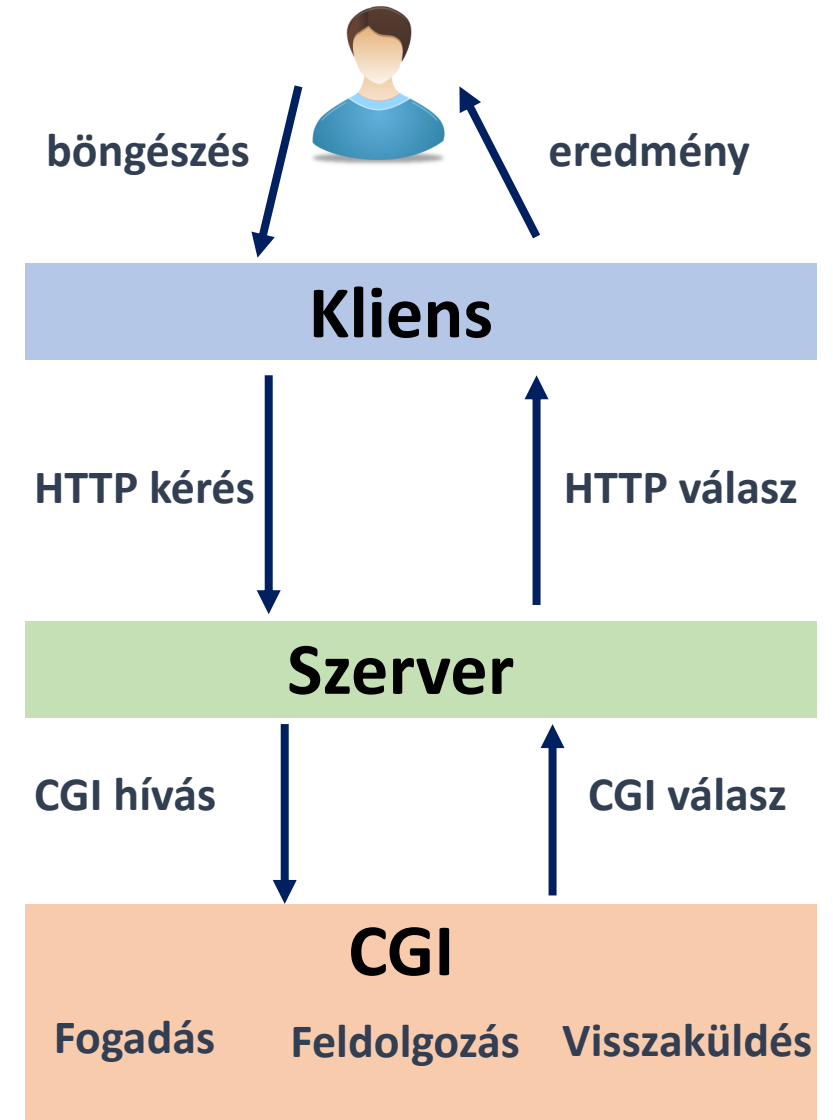






- **CGI alkalmazások**

- Tetszőleges programozási nyelven írt önmagában futtatható alkalmazások (.exe)
- Minden programozási nyelvben lehetőség van **szabványos bemenet** és **szabványos kimenet** kezelésére
  - Van egy bemeneti adathalmaz, és különböző műveletek segítségével valamilyen kimenetet ad a program a futtatás végén
  - Webszerver környezetben a **HTTP kérés** kerül a szabványos bemenetre, és a program válaszát a webszerver a **HTTP válaszba** illeszti
- Elavult módszer, korlátozott lehetőségekkel
  - Nagy erőforrásigény
  - Korlátolt szolgáltatáskészlet



- **Programozási nyelv specifikus webkonténer**

- A webszerver adott programozási nyelven írt kódrészleteket (objektumokat) egy speciális futtató környezetben tart, és kérés esetén az adott kódrészlet lefut és valamilyen eredményt ad

- Példák:

- **Java programozási nyelv: JavaEE Servlet**

- Igen sok webszerver alkalmazás támogatja

- **PHP programozási nyelv**

- Rendkívül népszerű, szinte az összes webszerver támogatja

- **Microsoft C#: ASP.NET**

- A Microsoft saját programozási nyelve, és csak Microsoft webszerverek támogatják
- Microsoft IIS: Internet Information Services

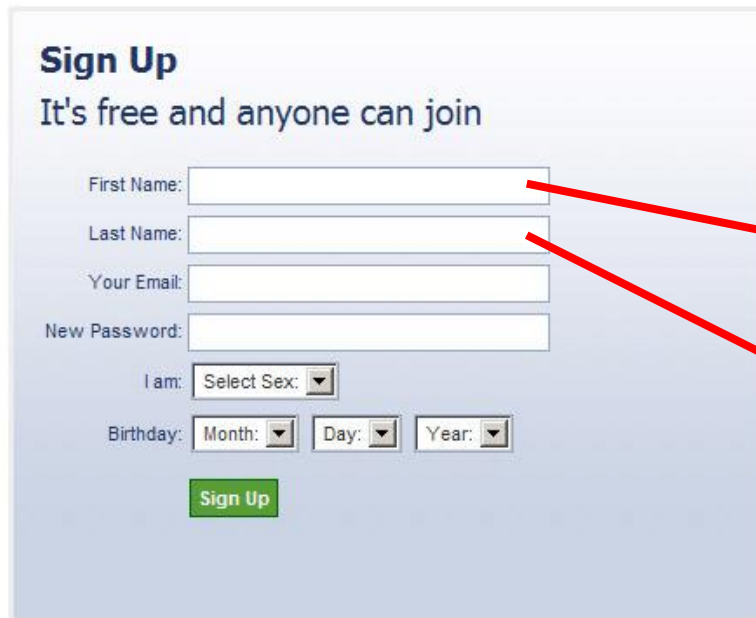
- **Microsoft C#: ASP.NET Core**

- A Microsoft saját cross-platform programozási nyelve



Visual C#

- A HTML tipikusan HTTP válaszok formázott **megjelenítésére** való
  - A szerver válasza szép, igényesen formázott oldalak/oldalrészletek
- Hogyan **küldhetünk** tehát adatot?
  - HTML űrlapelemek segítségével



**Sign Up**  
It's free and anyone can join

First Name:

Last Name:

Your Email:

New Password:

I am:

Birthday:

`<form>`

First name:`<br>`

`<input type="text" name="firstname"><br>`

Last name:`<br>`

`<input type="text" name="lastname">`

`</form>`

- Az űrlapelemek valamilyen egyedi névvel rendelkeznek (name attribútum)
- Az űrlapelemek tartalmát a böngésző a kérésbe ágyazva elküldi a szerver számára
  - **Adattovábbítás GET módszerrel**
    - Az URL címbe a dokumentum neve mögé kerülnek az űrlapelemek tartalmai
    - A szerver innen tudja kiolvasni és továbbítani a feldolgozó CGI program/konténer számára
    - Bárki láthatja (pl: előzményekben is szerepel a teljes, paraméterezett URL cím)
    - Hossza korlátozott

```
<form action="/action_page.php">
```

```
  First name:<br>
```

```
  <input type="text" name="firstname" value="Mickey"><br>
```

```
  Last name:<br>
```

```
  <input type="text" name="lastname" value="Mouse"><br><br>
```

```
  <input type="submit" value="Submit">
```

```
</form>
```

/action\_page.php?firstname=Mickey&lastname=Mouse  
.....



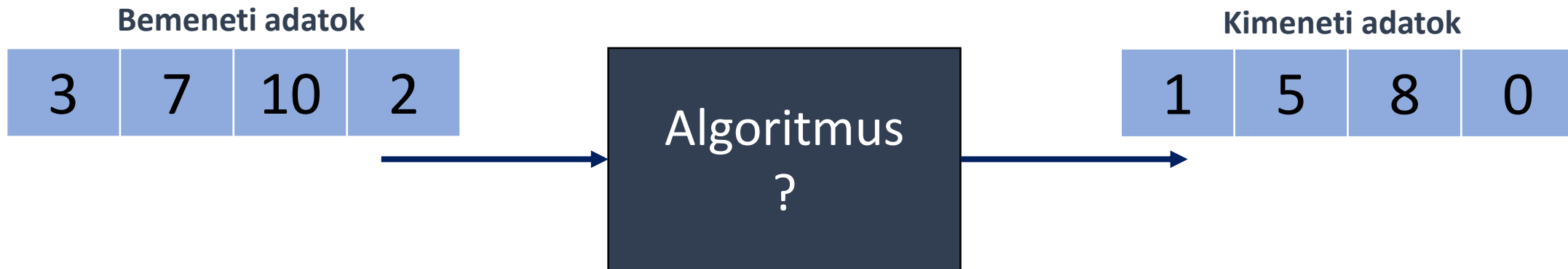
- Az űrlapelemek valamilyen egyedi névvel rendelkeznek (name attribútum)
- Az űrlapelemek tartalmát a böngésző a kérésbe ágyazva elküldi a szerver számára
  - **Adattovábbítás POST módszerrel**
    - A kérés törzsében egy gyűjteményként továbbítódnak az űrlapelemek tartalmai
    - Nem látható az URL-ben semmilyen utalás rá
    - Biztonságosabb a GET-nél
    - Korlátlan adatot lehet vele küldeni



- Az űrlapelemek értékeit GET vagy POST metódussal elküldjük a szervernek, ott átadásra kerül CGI programnak/webkonténernek
- Mindkét esetben az űrlapelemek értékei **változóként** lesznek elérhetőek (változó: adott névvel elérhető adat, amivel tudunk dolgozni a programban)
- De ez már szoftverfejlesztői feladat 😊



- Valamely hétköznapi probléma megoldása számítógéppel.
- A problémára találunk egy matematikailag helyes megoldást (**algoritmust**), majd ennek az algoritmusnak az **implementációja** a szoftverfejlesztés.
- Egy programnak/szoftvernek mindig vannak valamilyen bemenő adatai, kimeneti eredményei, és közöttük valamilyen műveletvégzés történik.



- Az összes létező szoftverre igaz ez a séma
  - **Táblázatkezelés**
    - **Bemenet:** rendelkezésre álló adatok
    - **Algoritmus:** felhasználói interakciók kezelése, halmazműveletek, összegzés
    - **Kimenet:** módosított adatok, szűrt adatok, grafikonok
  - **Közösségi oldalak**
    - **Bemenet:** Adatbázisban tárolt felhasználói adatok, üzenőfal postok, fényképek, like-ok
    - **Algoritmus:** felhasználói interakciók kezelése, adatbázis lekérdezések, szűrések, összegzés
    - **Kimenet:** Ezek megjelenítése, kategorizálása, like-ok száma
  - **Akkumulátor töltöttség kijelzés a tálcán**
    - **Bemenet:** Szenzor megméri az akkumulátor feszültségét, számként tárolja
    - **Algoritmus:** A számérték (pl: 0-1024) transzformálása 0-5 közötti számmá (pl: osztás 200-al)
    - **Kimenet:** Egy ábra, ahol 5 vonás mutatja a szintet

- A fejlesztés tehát algoritmus alkotás és ennek lekódolása programozási nyelven
  - **Algoritmizálás:** Programtervező informatikusok, matematikusok tipikus feladata
  - **Implementálás:** Fejlesztők tipikus feladata
    - (ez azért ennyire nem válik határozottan ketté, egy jó informatikus mindkét fázishoz ért)
- Az algoritmizálás során használható eszközök:
  - **Változók**
  - **Tömbök**
  - **Elágazás**
  - **Ciklus**

- A program bemeneti adatai a számítógép memóriájába kerülnek, és ezekre név szerint tudunk hivatkozni.
- PHP példakód a változók kezelésére

```
<?php
```

← PHP program kezdete

```
$a = 76;
```

← Létrehozunk egy a nevű változót 76-os értékkel

```
$b = $a + 100;
```

← A b változót is létrehozzuk, kapja meg a értékét és 100-at

```
echo $b;
```

← Küldjük vissza a felhasználónak b értékét, mint eredményt

```
?>
```

← PHP program vége

- Összetartozó változók, az egyes cellákat egész számokkal érjük el (indexelés)
- PHP példakód a tömbök kezelésére

```
<?php
```

← PHP program kezdete

```
$tomb = array();
```

← Létrehozunk egy üres tömböt

```
$tomb[0] = 6;
```

```
$tomb[1] = 8;
```

← Első három celláját értékekkel töltjük fel (0 az első)

```
$tomb[2] = 2;
```

```
echo $tomb[1] + $tomb[2];
```

← Kiírjuk az 1-es és 2-es indexű elem összegét

```
?>
```

← PHP program vége

- Valamilyen feltételt kiértékelünk és ennek függvényében A vagy B művelet zajlik le, ezután a program folytatja szekvenciális futását.
- PHP példakód az elágazások kezelésére

```
<?php
```

```
$eletkor = 56;
```

```
$kedvezmeny = 0;
```

```
if ($eletkor < 50)
```

```
    $kedvezmeny = 0.2;
```

```
else
```

```
    $kedvezmeny = 0.3;
```

```
$ar = 100;
```

```
$vegosszeg = $ar * (1 - $kedvezmeny);
```

```
?>
```

Művelet igaz kiértékelés esetén

Művelet hamis kiértékelés esetén

Műveletek az elágazás után  
(mindkét ág esetén lefut)



- Valamilyen programrészlet folyamatos ismétlése, amíg a feltétel meg nem dől
- PHP példakód a ciklusok kezelésére

```
<?php
$honnan = 3;
$hova = 10;
$pillanatnyi = $honnan;
while ($pillanatnyi <= $hova)
{
    echo $pillanatnyi;
    $pillanatnyi = $pillanatnyi + 1;
}
?>
```

Addig kell ismételni a ciklust, amíg ez a feltétel hamissá nem válik

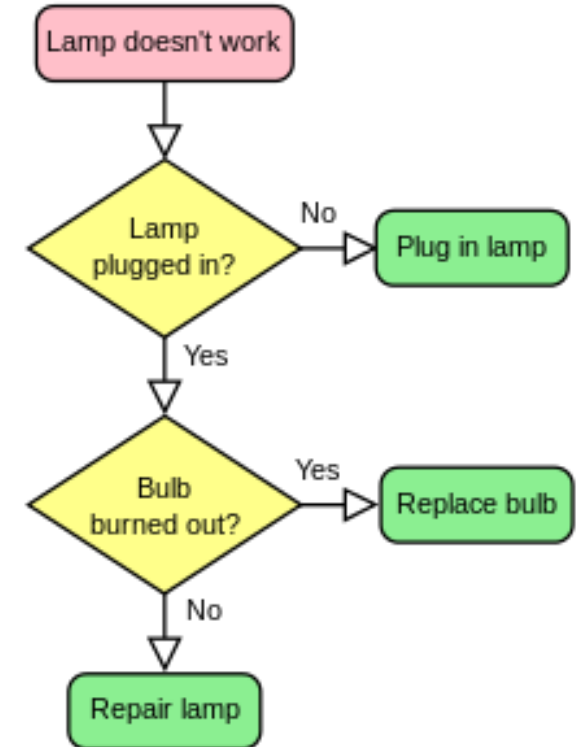
Ciklustörzs: ez a rész fog ismételten többször lefutni

- **Böhm-Jacopini tétel (1966)**

- Minden algoritmus megvalósítható a 3 programozási szerkezet, a szekvencia, a szelekció és az iteráció segítségével.
  - **Szekvencia:** utasítások egymás után
  - **Szelekció:** elágazás
  - **Iteráció:** ciklus

- **Algoritmusokról egyébek**

- Véges lépésből állnak
- Ha a probléma matematikailag megoldható, akkor le is programozható
- Ha az algoritmus a való élet egy problémáját oldja meg, akkor a valóságot szimulálni tudjuk (kutatási célú szimulációk, mesterséges intelligencia, stb.)



```
<?php
$tomb = array(2,8,3,4,9,10);
$i = 0;
$hossz = 6;
$db = 0;
while ($i < $hossz)
{
    if ($tomb[$i] > 5)
    {
        $db = $db + 1;
    }
    $i = $i + 1;
}
echo $db;
?>
```

## Megszámlálás algoritmus

- Adott egy tömb a felsorolt elemekkel
- Szeretnénk meghatározni, hogy hány olyan elem van a tömbben, amelyek 5-nél nagyobb
- Módszer
  - Egy \$i segédváltozóban tároljuk, hogy éppen melyik elemnél tartunk a tömbben
  - Pontosán ismerjük, hogy hány elemű a tömb (\$hossz)
  - Egy ciklust addig futtatunk, amíg el nem érjük az i-vel a tömb hosszát (hossz: 6, érvényes elemek indexe: 0..5)
  - Minden cikluslépésben (iteráció) megvizsgáljuk, hogy az adott helyen lévő elem nagyobb-e 5-nél
    - Ha nagyobb, akkor a darabszámot növeljük
    - Ha nem nagyobb, akkor nem csinálunk semmit
  - De minden iterációban növeljük az i-t, hogy a következő elemre léphessünk segítségével
  - Végül a felhasználónak kiírjuk a darabszámot

- A feladat, amit szeretnénk megoldani
  - Kérjük be a felhasználó nevét!
  - Kérjük be a felhasználó életkorát!
  - Hogyha 50 évnél fiatalabb a felhasználó, akkor citromsárga háttérrel adjunk vissza neki, ha idősebb, akkor narancssárgát!
  - Köszöntsük nevén!
  - Rajzoljunk ki annyi gyertyát a weboldalra, ahány éves!



- **Fordított nyelvek**

- Miután megírtuk a programot egy fordítóprogram (**compiler**) futtatható kódra fordítja.
- Ezután a futtatható kód vagy a processzoron fut (**natív alkalmazás**) vagy pedig egy **köztes rétegnek** lesz továbbítva, ami a processzoron futtatja
  - **Natív alkalmazás:** C és C++
  - **Köztes réteg által futtatott alkalmazás**
    - **Virtuális gép:** Java
    - **.NET keretrendszer:** C#, Visual Basic, F#, J#

- **Értelmezett nyelvek**

- Miután megírtuk a programot, a benne található utasítások sorról-sorra egy értelmező programhoz kerülnek (**interpreter**), amely értelmezi az adott sort, és meghívja a szükséges alkalmazást, operációs rendszer szolgáltatást, stb.
  - Példa ilyen nyelvekre: PHP, Batch script, Windows Powershell script

Köszönöm a figyelmet!