# Alapfogalmak megismerése

## Weboldal / Webalkalmazás

Fontos, hogy különbséget tegyünk a kettő között. A **weboldal** statikus, tartalma nem vagy alig változik. A **webalkalmazás** ezzel szemben egy olyan program, amelyhez egy webböngészőn keresztül lehet hozzáférni, és amely lehetővé teszi a felhasználók számára, hogy adatokkal interakcióba lépjenek és azokat manipulálják, feladatokat hajtsanak végre, és személyre szabott élményeket kapjanak.

### Főbb különbségek

|  |  |  |
| --- | --- | --- |
| **Jellemző** | **Weboldal** | **Webes alkalmazás** |
| Cél | Elsősorban az információterjesztés céljából | Feladatorientált, interaktív és dinamikus |
| Tartalom | Általában statikus, változatlan a frissítésig | Dinamikus, a felhasználói műveletek vagy adatok alapján változó |
| Kölcsönhatás | Korlátozott interakció, többnyire passzív nézés | Aktív interakció, felhasználói bevitel és válaszok |
| Funkcionalitás | Alapvető funkciók, mint például a navigáció és a keresés | Speciális funkciók, adatfeldolgozás stb. |
| Példák | Hírportálok, blogok, információs oldalak | Online banki szolgáltatások, közösségi média, e-kereskedelem |

## Protokoll

Ha valamit közölni akarunk egy másik személlyel a körülöttünk lévő világ minden részére szavakat – kódokat – használunk valamilyen ismert nyelven. A számítógépes kommunikációban ugyanúgy, mint a beszédben a továbbítandó mondanivalót kódolnunk kell. A protokollok az informatikában a beszélt nyelvi szabályoknak megfelelő kódolási szabályokat jelenti. Mint ahogyan a beszélt nyelvben is van több nemzeti nyelvi szabály, ugyanúgy van az informatikában is nagyon sok protokoll. Mindegyik használatának más-más célja van.

### HTTP *(HyperText Transfer Protocol)*, HTTPS (… Secure)

Az internet leggyakrabban használt protokolljai. Multimédiás dokumentumok leírására használják az 1990-es évek elejétől. Úgy lett megalkotva, hogy a számítógép hardverétől és operációs rendszerétől függetlenül ugyanabban az elrendezésben tudja megjeleníteni a szöveget, képet, mozgóképet és hangokat tartalmazható dokumentumokat, mint ahogyan azt a készítő meghatározta. Minden internetes dokumentumnak rendelkeznie kell **URI**-val*(Uniform Resource Identifier, vagyis egységes erőforrás-azonosító*).

## **H**yper**T**ext

A **H**yper**T**ext a web dokumentumnak azon szöveges része, amely egy másik internetes dokumentumra való utalást tartalmaznak.

## Portok

Minden kommunikáció kódolás – továbbítás – dekódolás lépésekből áll. Mivel nagyon sok kódolási forma – protokoll – használatos, ezért a fogadó eszközt „tolmáccsal” kell ellátni. Egy tolmács csak egyetlen egy nyelven tud beszélni. Ezek a „tolmácsok” csak egy adott kapun keresztül érhetőek el, amelyeket 0 és 65 535 közötti számmal azonosítanak. A HTTP „tolmácsa” alapértelmezetten a 80-as kapun, a HTTPS „tolmácsa” a 443-as kapun várja a dekódolandó üzeneteket.

## Domain nevek

Az URI (*Uniform Resource Identifier, vagyis egységes erőforrás-azonosító*) karakterlánc, amelyet webes erőforrások azonosítására és elérésére használnak leggyakrabban URL-ek (*Uniform Resource Locator*) formájában jelenik meg, azaz webcímként.

Például:   
[**https://www.example.com:8080/products/item?id=123&lang=hu#reviews**](https://www.example.com:8080/products/item?id=123&lang=hu#reviews)

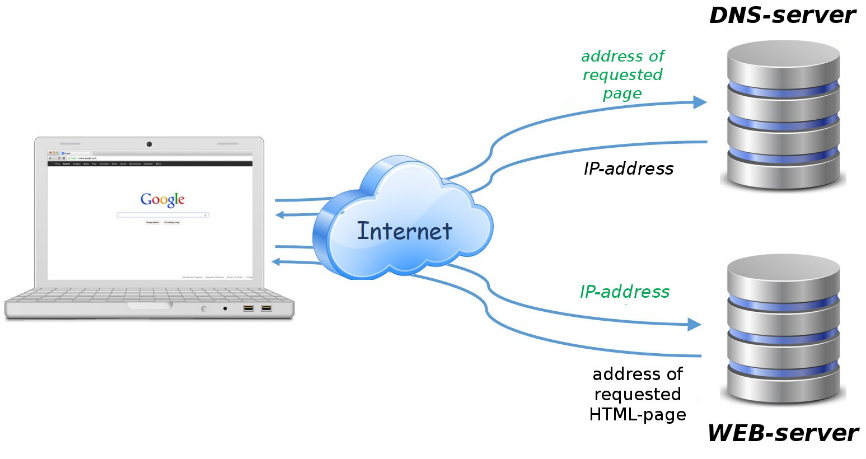
| **Rész** | **Név** | **Leírás** |
| --- | --- | --- |
| https | **Protokoll (Scheme)** | A kapcsolat típusa. Itt: **HTTPS** (biztonságos HTTP) |
| www.example.com | **Host (Domain név)** | A kiszolgáló címe, amelyhez csatlakozunk |
| :8080 | **Port** | A kiszolgáló portja (nem kötelező). Alapértelmezett: 80 (HTTP), 443 (HTTPS) |
| /products/item | **Elérési út (Path)** | A kiszolgálón belüli elérési út az adott erőforráshoz |
| ?id=123&lang=hu | **Lekérdezési karakterlánc (Query string)** | Paraméterek az URL-ben, kulcs-érték párokkal |
| #reviews | **Fragmentum (Anchor)** | Az oldal adott részére való ugrás, pl. egy HTML elem azonosítója |

## Böngészők

A böngészők olyan szoftverek, amelyek alkalmasak arra, hogy az internetes dokumentumokat, vagyis a weblapokat a készítő szándékának megfelelő formában tudják megjeleníteni. Ilyen programból nagyon sok készült. A jelenleg legnépszerűbbek: Google Chrome, Safari, Microsoft Edge, Mozilla Firefox, Brave és Vivaldi.

## kliens-szerver modell

Ha valaki meg akar jelenni az interneten, akkor kódolja a HTML5 szabályai szerint a dokumentumát és keres egy olyan szolgáltatót, amelyen bármelyik felhasználó elérheti. Itt kap egy azonosítót, egy URL-t. Ennek az URL-nek a megadásával a böngésző lekéri a dokumentumot és megjeleníti a weblapot.



1. ábra Hogyan működik az internet?

Aki olvassa a dokumentumot az a **kliens**, ahol elérhető az a **szerver**.

### Hosting (*Tárhelyszolgáltatás / Webszolgáltatás*)

A **hosting** (*tárhelyszolgáltatás*) az a szolgáltatás, amelyet egy hosting cég nyújt, lehetővé téve magánszemélyek vagy szervezetek számára, hogy weboldalaikat vagy webes alkalmazásaikat az interneten elérhetővé tegyék. Lényegében a hosting szolgáltatók szervereket adnak bérbe, amelyekre feltöltheted a weboldalad fájljait, adatbázisait és egyéb szükséges elemeit.

## Front-end és back-end

**Front-end**, amit a felhasználó lát és amivel interakcióba lép (*HTML, CSS, JavaScript*).

**Back-end**, a háttérben futó logika és adatkezelés (adatbázis, szerveroldali kód).

## API (*Application Programming Interface*)

Egy interfész, amely lehetővé teszi két szoftverrendszer (pl. egy weboldal és egy külső szolgáltatás) közötti kommunikációt.

## Verziókövetés (*Version Control*)

A szoftverfejlesztésben használt fájlok változásainak nyomon követésére és kezelésére. A Git a leggyakrabban használt verziókövető rendszer, a GitHub pedig egy népszerű platform a Git technológiával történő tárolásra és együttműködésre.

## Adatbázis (*Database*)

Szervezett gyűjteménye az adatoknak, amelyeket a weboldal használ (pl. felhasználói adatok, termékek listája, …).

# Fontos elvek és gyakorlatok

## Reszponzív dizájn

Egy weboldal akkor reszponzív, ha jól jelenik meg mobilon, tableten és asztali gépen is. Ennek érdekében változtatja az elemek méretét, egymáshoz viszonyított helyét

## SEO *(Search Engine Optimization)*

Hogyan legyen a weboldal keresőbarát.

## Biztonság *(Web Security)*

Pl. XSS, CSRF, HTTPS, adatvédelem (GDPR).

# Modern webfejlesztési trendek

## Single Page Applications (SPA)

Pl. React, Vue, Angular használatával.

## Progressive Web Apps (PWA)

Offline működés, app-szerű élmény.

## JAMstack

Statikus oldalak dinamikus funkciókkal (pl. Next.js, Gatsby).

# Eszközök és fejlesztési környezet

* Kódeditor használata (pl. VS Code, https://code.visualstudio.com/)
* Böngészőfejlesztői eszközök (DevTools)
* Verziókezelés: Git és GitHub alapjai

# HTML – a weboldal váza

## HTML alapstruktúra

<!**DOCTYPE** html>

<**html** lang=**"en"**>

<**head**>

<meta charset=**"UTF-8"**>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1.0"**>

<title>**Document**</title>

<link rel=**"stylesheet"** href=**"harrmadik.css"**>

</head>

<**body**>

<p>**Közlendő szöveg**</p>

</body>

</html>

## Szöveges tartalmak

### Címek

### Bekezdések

### Listák

### Képek

### Linkek

### Táblázatok

### Űrlapok

* HTML5 szemantikus elemek (header, nav, section, article, footer)

# CSS – a megjelenés kialakítása

## Alap szintaxis és szabályok

body **{**

background-color: rgb(235, 235, 104);

font-family: Arial, sans-serif;

}

szelektor **{**

tulajdonság1: érték1;

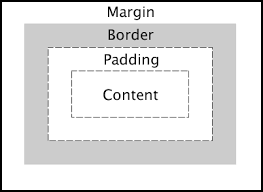
tulajdonság2: érték2;

}

## Színek

## Betűk

## Box model



2. ábra Box model

margók, padding, border

## Osztályok és azonosítók használata

## Elhelyezés: Flexbox és Grid

## Media query-k (reszponzív design alapjai)

# Gyakorlás – egyszerű weboldal készítése

* Statikus oldalak készítése HTML + CSS segítségével
* Reszponzív (mobilbarát) elrendezés készítése

# JavaScript – interaktivitás hozzáadása

* Alapfogalmak: változók, ciklusok, feltételek, függvények
* DOM manipuláció: elemek kiválasztása, eseménykezelés
* Űrlapellenőrzés
* Egyszerű animációk, események

# Modern front-end technológiák (alapszint)

* Bevezetés a CSS preprocesszorokba (pl. SCSS)
* JavaScript keretrendszerek (React, Vue, stb.)
* Build eszközök (pl. Vite, Webpack – opcionálisan)

# Weboldal publikálása

* Ingyenes hosting lehetőségek: GitHub Pages, Netlify
* Fájlok feltöltése, hibakeresés, végső tesztelés

# További tanulási irányok

* Backend (pl. Node.js, PHP, Python Flask/Django)
* Adatbázis-kezelés (SQL, MongoDB)
* API-k, REST, JSON használata