1 Alapfogalmak megismerése

1.1 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.

1.1.1 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

1.2 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.

1.2.1 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*).

1.3 HyperText

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

1.4 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.

1.5 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

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

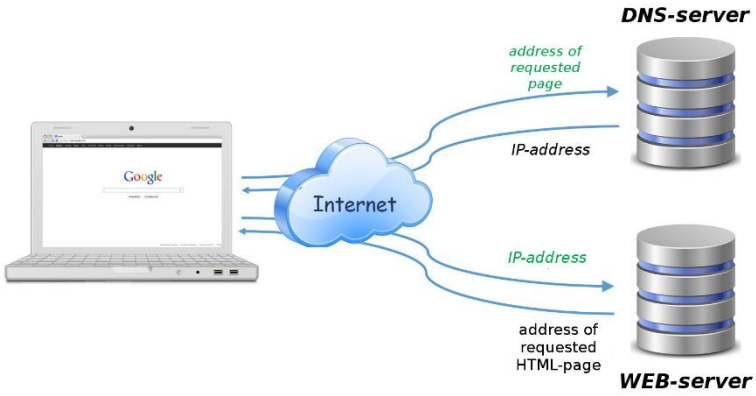
Rész	Név	Leírás
#reviews	**Fragmentum (Anchor)**	Az oldal adott részére való ugrás, pl. egy HTML elem azonosítója

1.6 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.

1.7 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**.

1.7.1 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.

1.8 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).

1.9 API (*Application Programming Interface*)

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

1.10 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.

1.11 Adatbázis (*Database*)

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

2 Fontos elvek és gyakorlatok

2.1 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

2.2 SEO *(Search Engine Optimization)*

Hogyan legyen a weboldal keresőbarát.

2.3 Biztonság *(Web Security)*

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

3 Modern webfejlesztési trendek

3.1 Single Page Applications (SPA)

Pl. React, Vue, Angular használatával.

3.2 Progressive Web Apps (PWA)

Offline működés, app-szerű élmény.

3.3 JAMstack

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

4 Eszközök és fejlesztési környezet

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

5 HTML – a weboldal váza

5.1 HTML alapstruktúra

```html
<!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>
```

5.2 Szöveges tartalmak

5.2.1 Címek

5.2.2 Bekezdések

5.2.3 Listák

5.2.4 Képek

5.2.5 Linkek

5.2.6 Táblázatok

5.2.7 Űrlapok

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

6 CSS – a megjelenés kialakítása

6.1 Alap szintaxis és szabályok

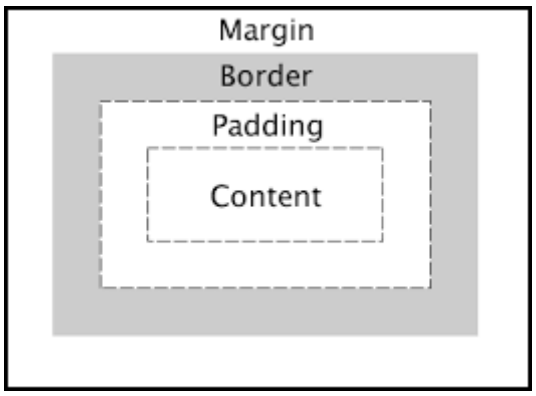
```css
body {
  background-color: rgb(235, 235, 104);
  font-family: Arial, sans-serif;
}
```

```
szelektor {
  tulajdonság1: érték1;
  tulajdonság2: érték2;
}
```

6.2 Színek

6.3 Betűk

6.4 Box model



2. ábra Box model

margók, padding, border

6.5 Osztályok és azonosítók használata

6.6 Elhelyezés: Flexbox és Grid

6.7 Media query-k (reszponzív design alapjai)

7 Gyakorlás – egyszerű weboldal készítése

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

8 JavaScript – interaktivitás hozzáadása

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

9 Modern front-end technológiák (alapszint)

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

10 Weboldal publikálása

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

11 További tanulási irányok

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