9. Az OSI modell alkalmazási rétege és a weboldal

(Kidolgozta: Kaprinay Patrik)

1. Írja le az OSI modell alkalmazási rétegét és hasonlítsa össze a TCP/IP modellel.
   1. Mutassa be és írja le a C/S és a P2P hálózati kommunikációt.

**C/S** – *Client Server*, kapcsolatteremtés egy szerver és egy (vagy több) kliens között. A szerver a kiszolgáló, a kliens pedig kér, vagy ad valamit a szervertől vagy a szerverre.

**P2P** – *Peer to Peer*, két peer (kliens) közötti kapcsolatot teremt, a kliens-szerver modellel szemben mind a két kliens fogad és átad információkat. Manapság a legelterjedtebb használati módja az interneten való fájlátviteli módszerek. Ezt a technológiát használják a torrentezés világában, egyenesen arra a kliensre csatlakozik, amelyiknél megtalálható az általunk keresett fájl.

* 1. Röviden jellemezze: DNS, HTTP, HTTPS (SSL/TLS), Telnet, SSH, FTP, SFTP, FTPS, TFTP, SMTP, POP, IMAP, DHCP.

**DNS** – *Domain Name System*, a tartománynév-hierarchiát üzemelteti és fordítási szolgáltatást nyújt a tartománynevek és az IP-címterek között (lefordítja a domain neveket IP címekre). Alapértelemezett portja: 53

**HTTP** – *Hypertext Transfer Protocol*, egy információátviteli protokoll elosztott, kollaboratív, hipermédiás, információs rendszerekhez. A HTTP egy kérés-válasz alapú protokoll kliens és szerver között. **Kizárólagosan** TCP/IP protokollt használ, mivel az adatveszteség nem megengedhető. Alapértelmezett portja: 80

**HTTPS** – *Hypertext Transfer Protocol Secure*, a HTTP bővítménye, ami titkosítja a kapcsolatot a szerver és a kliens között. TLS-sel vagy SSL-lel titkosítják a kapcsolatot.   
TLS/SSL – Transport Layer Security, Secure Sockets Layer, egy titkosításra szolgáló protokoll, használják e-mailek, csevegőapplikációk és VOIP titkosítására is, de legfőképpen a HTTPS használatánál lett ismert. 2011-ben leváltotta az SSL-t, mert az SSL nem minősült biztonságos protokollnak és számos sebezhetőségi pontja volt. Alapértelmezett portja: 443

**Telnet** – Az egyik legősibb hálózati protokoll, TCP/IP alapon működik. Megfelelő kapcsolatteremtésre két számítógép között, fájlokhoz való hozzáférésre, programok futtatására a hálózaton keresztül. Helyette inkább az SSH-t használják, ami a telnetet, rlogin-t, rsh-t vette alapul. A telnet protokoll az adatokat a hálózaton nyers szövegformátumban továbbítja, tehát érzékeny adatokkal való munkára nem megfelelő. Alapértelmezett portja: 23

**SSH** – *Secure Shell*, szinte megegyezik a Telnettel, annyi különbséggel, hogy biztosított a kapcsolat a két számítógép között. Funkciójai közé tartozik a távoli parancssoros felülethez való hozzáférés, bejelentkezés a rendszerbe és egyéb távoli parancsvégrehajtás. Két része van, az SSH kliens és az SSH szerver. Ez a protokoll is TCP/IP protokollt használ és az alapértelmezett portja a 22.

**FTP** – *File Transfer Protocol*, az interneten való állományküldésre szolgál, a kapcsolat egy szerver és egy kliens között alakulhat ki, ezáltal a kliens-szerver modell alapelvére épül. A kapcsolatfelvételhez szükséges a hitelesítés, ami név és jelszó beadásával történik meg, de névtelen módban is csatlakozhatunk, ha úgy van beállítva a szerverünk. A titkosított név és jelszó hálózaton való átviteléhez SSL-t illetve TLS-t használhatunk (FTPS), vagy helyettesíthetjük SFTP-vel (SSH File Transfer Protocol). Weboldal szerverek általában FTP-t használnak, hogy a fejlesztő számítógépéről a szerverre kerüljenek a szükséges fájlok, amelyek majd futnak a szerveren (weboldal, HTML, PHP). Alapértelmezett portok: 21 (csatlakozás létrehozása), 22 (adatátvitel)

**TFTP** – *Trivial File Transfer Protocol*, manapság már nem használt interneten való filetovábbításra, inkább a helyi hálózatban használt. Felépítése igen egyszerű és nem memóriaigényes. UPD/IP protokollt használ és alapértelmezett portja a 69. A TFTP képes írni, olvasni a szerverre vagy a szerverről, viszont nem tudja kilistázni a szerveren lévő fájlokat, átnevezni, illetve törölni sem azokat.

**SMTP** – *Simple Mail Transfer Protocol*, email-ek küldésére és fogadására alkalmas kommunikációs protokoll. TCP/IP protokollt használ, 587-es (25) a portja. Manapság inkább csak email küldésre használják, fogadásra IMAP-ot és POP3-mat használnak.

**POP** – *Post Office Protocol*, ezt a kommunikációs protokollt használják email-ek webszerverről való letöltésére a kliens számítógépére. Amint megtörtént a letöltés a szerverről letörlődhet az összes átadott email, attól függően, hogy hogy állítottuk be. Legújabb verziója a POP3. Portja a 110. A titkosított változata TLS/SSL-t használ (POP3S), portja 995.

**IMAP** – *Internet Message Access Protocol*, TCP/IP protokollt használ, e-mailek, emailszerverről való letöltésére használják. Működési elve hasonló a POP-hoz, viszont az IMAP nem törli le automatikusan a már letöltött email-eket, ezáltal elérhetővé teszi a több kliensről való kezelést és ezáltal támogatja az online és offline módban történő szerkesztést is. Aktuális verziója az IMAP4.

**DHCP** – *Dynamic Host Configuration Protocol*, a DHCP szerver **automatikusan** hozzárendel a kliensekhez egy bizonyos IP-címet és átad nekik egyéb kapcsolódással kapcsolatos információt, ezáltal nem kell minden egyes kliensnél manuálisan beállítani. A DHCP szerverünkön be kell állítanunk azt a hatótávolságot, amelyben a DHCP kioszthatja az IP címeket (IP DHCP pool) a csatlakozott klienseknek.

1. Írja le az OSI alkalmazási réteg protokolljaival kapcsolatos támadásokat és ellenintézkedéseket.

Mivel az alkalmazási réteg áll a legközelebb a felhasználóhoz, ezért kedvező a hackerek részére ezt a réteget megtámadni, réseket keresni benne. Alkalmazás rétegi támadások közé sorolhatjuk pl: DDOS-t (HTTP flood), SQL injekciót és a XSS-t is.

**DDoS** – *Distributed Denial of Service - elosztott szolgáltatásmegtagadással járó támadás*

Egy támadási forma, ami egy adott eszköz felé irányul, több számítógép hajtja végre. A DDoS céljából együttműködő számítógépek (akár a számítógép felhasználójának tudta nélkül) hálózatát botnet hálózatnak nevezzük. Célja, hogy megakadályozza a megcélzott eszközök, rendszer, szolgáltatás elérését minden felhasználó számára. A támadás alapelve, hogy egy vagy több számítógépről annyi packetot (ICMP) küldünk a megcélzott szolgáltatásnak, szervernek, hogy nem képes feldolgozni annyi kérést. Sikeres támadáskor a szolgáltatás, eszköz, rendszer lényegesen lelassul, elérhetetlenné válik, vagy akár teljesen össze is omolhat. A DoS támadásnál a támadást kezdeményező számítógép egyedül van, azaz egy az egy ellen való támadás. A DDoS támadásnál több különböző számítógép is részt vesz a támadás kezdeményezésében. Ezeket a számítógépeket zombiknak is szokták nevezni. Ahhoz, hogy több számítógéppel tudjunk ilyen támadást végrehajtani, valahogy telepítenünk kell egy rejtett programot a többi számítógépen, általában vírusokban találhatóak meg. A HTTP Flood a DDoS egy fajtája, célja, hogy a megcélzott webszervert leterhelje vagy használhatatlanná tegye úgy, hogy sok kis méretű HTTP kérést küld neki. Két típusa van, GET és POST. A GET-et kérést általában a szerveren található képek megjelenítésére használják, a POST-ot pedig űrlapok kiértékelésre használják. Ezek gyakori, nagy mennyiségű elküldése DDoS-t eredményezhet. Ez a támadás ellen bevált módszer a CAPTCHA, az oldal egy felugró ablakban megkér, hogy válasszuk ki, hogy nem vagyunk robotok, majd a megadott képek közül ki kell választani azokat, amelyeken szerepel a program által megadott elem. A legismertebb CAPTCHA szolgáltatás a reCAPTCHA, amit a Google fejlesztett ki.

**SQL injekció** – *Structured Query Language* injekció

Az SQL a legelterjedtebb programozási nyelv a relációs adatbázis rendszerek körében. Míg a DDoS támadások nem eredményeznek információvesztést, az SQL injekció igencsak veszélyes lehet e szempontból. Alapelve az, hogy van egy űrlap, amit a felhasználónak ki kell töltenie, regisztráció, bejelentkezés, stb. és ebbe a bemeneti rész(ek)be SQL kódot írunk, amivel manipulálhatjuk az adatbázisban rejlő adatokat, vagy akár kitörölhetjük is azokat. Általában ebbe a bemeneti részbe beírt szöveg végén mindig megtalálhatjuk a (--)-t, ami a -- utáni részt kommentként kezeli, ezáltal elvetve az eredeti, tervezett operációt a felhasználó által beadott információval. Ha tudjuk az adatbázisban a táblázat nevét, a DROP TABLE *tablazat\_neve*; paranccsal megsemmisíthetünk akármilyen sok adattal rendelkező táblázatot, kereshetünk az adott táblázat sorai között, és ami a legveszélyesebb: adatot módosíthatunk, bankszámla esetén bankszámla egyenlegünket megváltoztathatjuk. Az adatok módosítása azért a legveszélyesebb, mert ez a legnehezebben észrevehető, hiszen az adminisztrátor számára nem kell, hogy egyértelmű legyen, hogy egy bizonyos sor bizonyos oszlopban lévő értéke megváltozott. Ezen támadás következményeit úgy tudjuk megakadályozni, hogy amikor elküldjük az űrlapot a szervernek feldolgozásra, akkor minden bemenetet, amit a felhasználó adott be, különböző ellenőrző feltételeknek vetjük alá. Ezen feltételek segítségével leellenőrizzük, hogy valóban megtalálható a választott érték a lehetőségek között, és a beadott szöveganyagot pedig egy változóban eltároljuk, és megmondjuk a rendszernek, hogy ez a változó tartalmát kizárólag csak szövegként vegye és ne utasításként, még ha azt is tartalmaz.

**XSS** – *Cross Site Scripting*

Az XSS támadás hasonló az SQL injekcióhoz, de míg az SQL injekció a szerver részen fut le, az XSS támadás a böngészőben hajt végbe kéretlen kódot. A JavaScript egy olyan programozási nyelv, ami 1995-ben arra volt kitalálva, hogy a böngészők is dolgozzanak valamilyen programozási nyelvvel, ezáltal dinamikus adatot tudjanak megjeleníteni a felhasználó részére. Az XSS támadást használják hitelesítés kijátszása érdekében, de akár más oldalakról való kódok beillesztésére is.

1. Magyarázza el a HTML nyelvvel kapcsolatos alapfogalmakat: címke (TAG), a címke jellemzője (Attribute) és értéke (Value). Hasonlítsa össze a páros és a páratlan címkéket, mutasson be példákat.

A HTML, *HyperText Markup Language*, egy 1993-ban kidolgozott és megjelentetett programozási nyelv, amit a weboldalak tartalmának megjelenítésére használnak. A HTML címkék vagy elemek a weboldal összetevői, ezekkel határozhatjuk meg, hogy az általunk beírt adat hogyan jelenjen meg, kép, szöveg, választási lista stb. Ismerünk páros és páratlan címkéket. A páros címkéknek van egy nyitó és egy záró tagja. Vegyük például a

<p></p>

címkepárt. Ahogy láthatjuk a két címke egy ’/’-vel különbözik. Ez a *slash* mutatja nekünk, hogy a p címke itt végződik. Ez a két címke közé beírt szöveg (érték - *value*) egy paragrafusként fog megjelenni az oldalunkon. A páratlan címkék annyiban különböznek, hogy nincs párjuk, ezáltal csak önmaguk alkotnak egy elemet. Pl.

<img src=”logo.png” alt=”Brand Logo”>

A fenti kódrészlet egy képcímkét mutat be. Ahogy láthatjuk a <> zárójelben találhatunk most egyebet is, mint a címke nevét. Ezek a src=”logo.png” és a alt=”Brand Logo”. Ezeket attribútumoknak, jellemzőknek hívjuk. Ezek egyéb paraméterek, amelyeket a címke tartalmaz. A src megadja, hogy a rendszer hol keresse az általunk megadott képet. Az alt az a szöveg, ami akkor jelenik meg, ha a képet nem sikerül betölteni. Példa egy bemeneti mezőre:

<input type=”text” name=”username” placeholder=”Enter your username”>

A placeholder az a szöveg, ami akkor jelenik meg, ha a felhasználó üresen hagyta az adott mezőt. Egy szöveges beviteli mezőt hasznos ellátni egy name atttribútummal is, hogy az űrlap elküldésénél és az adatfeldolgozásnál azonosítani tudjuk a belé írt adatot.

* 1. Sorolja fel az alapvető formázási csoportokat és a legfontosabb címkéket (TAG) mutassa be és hasonlítsa össze használatukat.
  2. Mutassa be a szövegtömb formázó címkéket, hasonlítsa össze a DIV és SPAN címkéket, hozzon létre a szövegformázást bemutató példát.

A <div></div> címkepár közé általában az oldal egyes valamilyen módon elkülöníthető részeit szoktuk helyezni, pl. az oldal fejléce, tartalmi része, lábjegyzet.   
A <span></span> címkepár hasonló a <div>-hez, de ebbe szövegformázás céljából helyezünk egy szövegrészt. Ha már behelyeztünk valamilyen szöveget a <span> tagba akkor elláthatjuk a <span>-t egy style=”” attribútummal, amiben a „”-ben megadhatjuk, hogy hogyan szeretnénk módosítani azt a szövegrészt. Pl.:

<p>Lorem <span style=”color: green;”>ipsum</span> dolor sit amet...</p>

Ezzel a módosítással az *ipsum* szövegrész zölden fog megjelenni.

A <div></div> részletbe helyezett oldalrészt is tudjuk kedvünk szerint módosítani. Meghatározhatjuk, hogy az adott blokk milyen magas, széles legyen, vagy akár a pozícióját is megadhatjuk. Pl.:

<div style=”width: 20px; top: 0;”><p>Lorem ipsum...</p></div>

Ezzel azt adtuk meg, hogy a blokkunk 20 pixel széles legyen és az oldalunk tetején helyezkedjen el.

1. Írja le a webűrlap létrehozását, mutassa be az űrlapon használható elemeket. Készítsen űrlapot az egyes elemek gyakorlati felhasználásának bemutatására!

<div>

   <form action="registration.php">

       <p><input type="text" name="username" placeholder="Username"></p>

       <p><input type="password" name="password" placeholder="Password"></p>

       <p><input type="number" name="age" placeholder="Age"></p>

       <select name="sex">

           <option value="m">Man</option>

           <option value="w">Woman</option>

           <option value="o">Other</option>

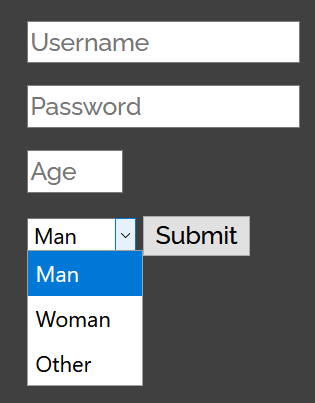
       </select>

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

   </form>

</div>

A következő kódsorok begépelésével valami hasonló egyszerű kis űrlapot tudunk készíteni.



A <form></form>-ba helyezzük el azokat az információkat, amelyeket továbbítani szeretnénk az űrlap elküldésénél. Az <input/> taggal beviteli értéket várunk a felhasználótól. <select></select> tagokba választási lehetőséget adhatunk a felhasználónak. A tagon belül elhelyezhetünk annyi <option><option> tagot, amennyi választási lehetőséget szeretnénk. Az <input type=”submit” /> gombbal tudjuk elküldeni az űrlapunkat.

1. Magyarázza el mi a CSS nyelv, mire használjuk és hogyan működik a HTML nyelvben.

**CSS** – *Cascading Style Sheet*, egy stílusleíró nyelv, amivel megadhatjuk, hogy a prezentált anyag, ami jelölőnyelvben van írva (általában HTML), hogy legyen megjelenítve.

Az a CSS stílus lesz kihatással az adott elemünkre, amelyik meghatározása a legközelebb áll a mi általunk módosítani kívánt elemhez. Azaz, tételezzük fel, hogy van egy <p></p> elemünk, amit becsomagolunk egy <div></div> tagba. Ha megadjuk, hogy a <div> színe zöld legyen és hogy a <p> színe pedig kék, akkor ha megjelenítjük az oldalt, akkor a <p> kék színű lesz, mert az a meghatározás áll a legközelebb az adott elemünkhöz. Ha hozzáadnánk pl. egy <h2> elemet a div-be, akkor az viszont zöld lenne.

* 1. Magyarázza el hogyan lehet CSS tulajdonságokat definiálni az egyes elemekben egészítse ki stílusokkal a mellékelt űrlapot.

A CSS tulajdonságokat egyes elemekre 3 féle képpen tudjuk meghatározni:

1. Az elembe beleírva:

Az elemet, amit személyre szeretnénk szabni kiegészítünk egy új attribútummal, a style=””-val, aminél a „” közé írjuk a CSS kódot.

1. A dokumentum elején:

A HTML dokumentumunk elején a <head></head> részben létrehozunk egy <style></style> tagpárat, amibe elhelyezhetjük a CSS stílusainkat.

1. Egy másik dokumentumba:

Létrehozunk egy új állományt, aminek a kiterjesztése .css, majd a HTML dokumentumunk <head></head> részében összekötjük a dokumentumunkkal.

<link rel="stylesheet" href="styles.css">

<div style="background-color: yellow;">

   <form action="registration.php">

      <p><input type="text" name="username" placeholder="Username" style="width: 200px;"></p>

      <p><input type="password" name="password" placeholder="Password" style="width: 200px;"></p>

      <p><input type="number" name="age" placeholder="Age" style="width: 70px;"></p>

      <select name="sex">

          <option value="m">Man</option>

          <option value="w">Woman</option>

          <option value="o">Other</option>

      </select>

      <input type="submit" value="Submit" style="width: 20px; padding: 15px;">

   </form>

</div>

* 1. Írja le a CSS nyelv lehetőségeit és a CSS definíciók HTML állományhoz való kapcsolását.

Ha külön CSS fájlba helyezzük a stílusainkat, akkor pontosan meghatározhatjuk egyes kulcsszavakkal, hogy melyik elemekre szeretnénk, hogy vonatkozzanak a megadott stílusok. Hasonlóan nézhet ki:

.photosBanner {

  font-size: 2.8rem;

  margin: 0;

}

Az első sorban láthatunk egy pontot (.) és egy nevet. Ebbe a sorba írjuk, hogy az ezt követő zárójelek között leírt stílusok az oldal melyik elemeire vonatkozzanak. Ez a kódrészlet az összes olyan elemre fog vonatkozni, amely rendelkezik photosBanner class-szal. CSS-ben a class-ot ponttal (.), az id-t pedig hashtaggel (#) jelöljük. Az első sorban használhatunk pontosabb kifejezéseket is, mint egy osztálynevet. Pl:

.progressBar ul li:not(:last-child) {

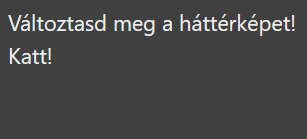
  margin-right: 25px;

}

Ez a stíluskifejezés azt mondja meg, hogy minden olyan elem, ami tartalmazza a progressBar  osztályt és létezik benne egy ul (azaz egy számozatlan lista) és abban egy li (azaz annak elemeire) vonatkozzon a zárójelben megadott módosítás. Az li utáni karaktereket úgy kellene olvasni, hogy: kivéve, ha utolsó elem. Szóval ez a stíluskifejezés minden egyes listaelemnek, kivéve az utolsónak, ad 25 pixelnyi jobb oldali margót.

1. Írja le a Javascript nyelvet, annak alapvető jellemzését és, hogy mire használható. Hozzon létre tetszőleges eseményt, amit hozzákapcsol egy létrehozott űrlaphoz.

A Javascript, sokszor JS-ként (dzséjesz) emlegetve, egy objektumorientált programozási nyelv, ami általában a weboldal elindításának pillanatában kerül értelemzésre, és a weboldalak alapvető alkotóeleme a HTML és a CSS mellett. Javascript által az oldalakat dinamikussá tehetünk, ezáltal kilistázhatunk változó tartalmat egyes feltételek alapján. A Javascript a PHP-vel ellentétben (a PHP a szerveren fut) a böngészőben fut, minden kliensnek a saját böngészője futtatja. Kis- és nagybetűérzékeny.

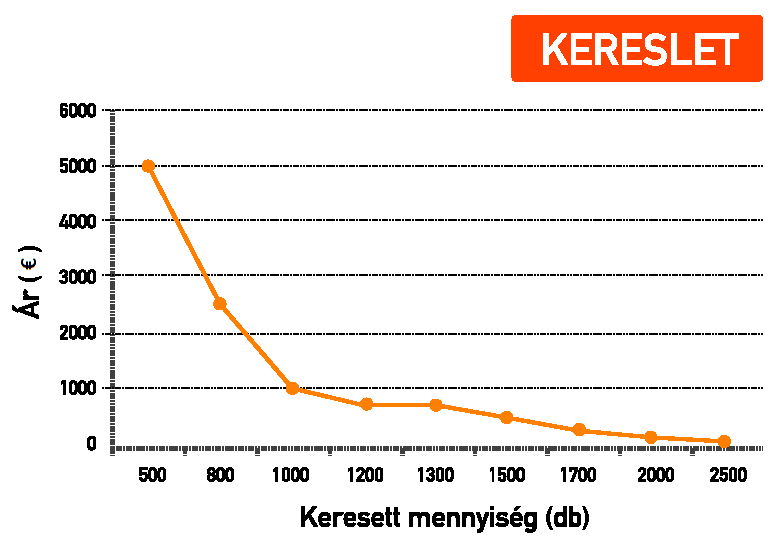


document.body.style.backgroundColor = "red";

1. Jellemezze a keresletet, a keresleti görbét, a kereslet csökkenésének törvényszerűségét és a keresletet befolyásoló tényezőket.

**Kereslet** – Azt az árumennyiséget jelenti, amelyet a vásárlók hajlandóak egy adott áron megvásárolni.

**Keresleti görbe** – Azt bizonyítja be, hogy az emberek alacsonyabb árak mellett egy adott terméket hajlandóbbak megvásárolni, mint nagyobb árnál.



**Kereslet csökkenésének törvényszerűsége** –

Ha egy általunk fogyasztott jószág normál jószág, vagyis – minden más tényező változatlansága mellett – a jövedelmünk (m) emelkedésének hatására a belőle vásárolt mennyiség (xi) is növekszik, akkor bizonyos, hogy az árának (pi) emelkedésekor a belőle fogyasztott mennyiség csökkenni fog. Hasonlóképpen, az árcsökkenés normál javak esetén minden esetben keresletnövekedést eredményez. *[Forrás: Wikipédia]*

*EGYSZERŰBBEN*:

Ha a vásárlónak megemelkedik a bevétele, akkor többet vásárol egyes termékekből.

Szintúgy ha az áru ára csökken, a vásárló többet engedhet meg magának belőle. És törvényszerűleg többet is fog vásárolni belőle.

**Keresletet befolyásoló tényezők** –

* fogyasztó jövedelme (a fogyasztó bevétele)
* a fogyasztó preferenciái (mit helyez előtérbe a fogyasztó)
* egyéb termékek árai
* jövőbeli elvárások (ha a fogyasztó feltételezi, hogy nagyobb lesz a bevétele a közeljövőben, akkor már a jelenben is hajlamosabb többet költeni)
* a fogyasztók száma (a fogyasztók mennyisége, hányan vásárolják az adott terméket,  
  fogyasztók növekedésével nő a kereslet is)