**HTML (HyperText Markup Language)**

HTML az alapvető építőköve a weboldalaknak, amelyeket böngészők értelmeznek és jelenítenek meg.

**1.Elemek és Címkék:**

* Az HTML oldalak elemekből épülnek fel, amelyeket címkékkel határoznak meg. Pl a <div>, <p>, <h1>, <img> stb.
* Az elemek lehetnek beágyazottak, vagyis egy elem tartalmazhat más elemeket.
* A címkék közötti hierarchiát és struktúrát a weboldalak felépítéséhez használják.

**2.Szemantika:**

* Az HTML címkéknek szemantikus jelentésük van, ami segíti a böngészőket és a keresőmotorokat a tartalom megértésében. Például a <header>, <nav>, <section>, <article> stb.
* A helyes szemantika lehetővé teszi, hogy a weboldalak megfelelően jelenjenek meg keresési eredményekben, képernyőolvasókban és más eszközökben.

**3.Attribútumok:**

* Az elemekhez hozzáadhatunk attribútumokat, amelyek további információkat adnak az elemről. Például az id, class, href, src stb.

**4.Formázás és Stílus:**

* Bár az HTML némi alapvető formázást biztosít, például a fejlécek, listák és táblázatok létrehozásához, a részletesebb megjelenési tulajdonságokat általában a CSS segítségével állítjuk be.

**CSS (Cascading Style Sheets)**

A CSS az adatok megjelenítését és stílusát írja le, amelyekkel a weboldalak vonzóbbá és esztétikusabbá válnak.

**1.Szelektorok és Tulajdonságok:**

* A CSS-ben a stílusokat szelektorok segítségével alkalmazzuk az elemekre.
* A stílusok tulajdonságokból és értékekből állnak, amelyek befolyásolják az elemek megjelenését. Például a color, font-size, background-color stb.
* A CSS szelektorok hierarchikus struktúrában alkalmazhatók, ami lehetővé teszi az elemek részletes és precíz formázását.

**2.Osztályok és ID-k:**

* Az elemek stílusainak megadásához osztályokat és ID-kat is használhatunk, ami lehetővé teszi a stílusok újrafelhasználását.
* Az osztályok és az ID-k használata megkönnyíti az egyedi stílusok és design elemek létrehozását.

**3.Öröklődés:**

A CSS öröklő jellege lehetővé teszi, hogy az elemek stílusai az őket körülvevő elemek stílusait is örököljék. Ez általában egyszerű és hatékony stíluskezelést eredményez.

**SQL adatbázisok (Structured Query Language)**

Az SQL egy nyelv az adatok tárolására, lekérdezésére, frissítésére és törlésére az adatbázisokban.

**1.Táblák és Rekordok:**

* Az adatokat táblákban tároljuk, amelyek sorokból vagy rekordokból állnak.
* A táblák oszlopokból állnak, amelyek meghatározzák az adatok típusát és szerkezetét.

**2.Lekérdezések (SELECT):**

* Az adatok lekérdezéséhez az SQL-ben a SELECT parancsot használjuk, amely lehetővé teszi a különböző feltételek alapján történő adatok kiválasztását.
* A lekérdezések sorrendje, csoportosítása és szűrése segíti az adatok hatékony és pontos megjelenítését.

**3.Frissítések (UPDATE):**

* Az adatok frissítéséhez az SQL-ben az UPDATE parancsot használjuk, amely lehetővé teszi a meglévő adatok módosítását.
* A frissítések segítségével az adatok naprakésszé tehetők és karbantarthatók.

**4.Törlések (DELETE):**

* Az adatok törléséhez az SQL-ben a DELETE parancsot használjuk, amely lehetővé teszi a rekordok eltávolítását a táblákból.
* A törlés lehetővé teszi a régi vagy feleslegessé vált adatok eltávolítását az adatbázisból.

**5.Beszúrások (INSERT):**

* Az adatok beszúrásához az SQL-ben az INSERT INTO parancsot használjuk, amely lehetővé teszi új rekordok hozzáadását a táblákhoz.
* Az új adatok beszúrása lehetővé teszi az adatbázis bővítését és frissítését új információkkal.

**6.Kapcsolatok (JOIN):**

* Az adatok közötti kapcsolatok kezeléséhez az SQL-ben a JOIN operátort használjuk, amely lehetővé teszi a táblák közötti összekapcsolások létrehozását.
* A kapcsolatok segítségével összetett adatlekérdezéseket hajthatunk végre, amelyek kihasználják az adatok közötti kapcsolatokat és összefüggéseket.

**7.Indexek és Triggerek:**

* Az SQL adatbázisokban indexeket és triggereket használhatunk a teljesítmény optimalizálása érdekében.
* Az indexek segítségével gyorsabb lekérdezéseket és kereséseket hajthatunk végre az adatbázisban.
* A triggerek automatikusan reagálnak az adatbázisban történő változásokra, és előre meghatározott műveleteket hajtanak végre.

**JavaScript**

JavaScript egy dinamikus, magas szintű programozási nyelv, amelyet általában a böngészőkben futó webalkalmazásokhoz használnak.

**1.Változók és Adattípusok:**

* A JavaScriptben változókat használunk az adatok tárolására, és különböző adattípusokat, például szövegeket, számokat, tömböket és objektumokat kezelünk.

**2.Függvények:**

* A függvények segítségével strukturálhatjuk a kódot és ismétlődő műveleteket végezhetünk el.
* A JavaScript függvények lehetnek beépítettek vagy egyedi, és paramétereket vehetnek át és visszatérhetnek értékekkel.

**3.Események és Aszinkronok:**

* A JavaScript lehetővé teszi az események kezelését, például a felhasználói interakciókat, és aszinkron műveletek végrehajtását, például adatkéréseket a háttérben.
* Az esemény vezérelt programozás lehetővé teszi a felhasználói felület interaktív és reaktív jellegét.

**4.DOM (Dokumentum Objektum Modell):**

* A DOM lehetővé teszi a JavaScript számára, hogy módosítsa a weboldal struktúráját és tartalmát, és dinamikusan változtassa azt a böngészőben.
* A DOM elemek hierarchikus fastruktúrában vannak szervezve, amely lehetővé teszi azok közötti navigációt és manipulációt.

**5.AJAX (Aszinkron JavaScript és XML):**

* Az AJAX lehetővé teszi a JavaScript számára, hogy adatokat kérjen és küldjön a szerverrel háttérben, anélkül hogy az oldal újra töltődik.
* Az AJAX segítségével dinamikus és interaktív webalkalmazásokat hozhatunk létre, amelyek felhasználói élménye folyamatos és sima.

**6.Modulok és csomagok:**

* A JavaScriptben modulokat használhatunk a kód szervezésére és újrafelhasználhatóságának növelésére.
* A modulok lehetővé teszik a kódbázis szétválasztását logikai egységekre, amelyek könnyen importálhatók és használhatók más részekben.
* A JavaScript ökoszisztémájában számos csomag és könyvtár elérhető, amelyek különféle funkciókat és kész megoldásokat kínálnak.

**7.Aszinkron programozás (Promise, Async/Await):**

* Az aszinkron programozás lehetővé teszi a hálózati kérések, fájlkezelés és más műveletek végrehajtását úgy, hogy közben a kód nem blokkolja a futást.
* A Promise objektumok segítségével lehetőség van aszinkron műveletek során visszatérési értékeket kezelni.
* Az Async/Await szintaktikai cukorként szolgál, amely lehetővé teszi az aszinkron kódrészletek írását szinkron módon, könnyen olvasható és karbantartható formában.

**8.Egységtesztelés (Unit Testing):**

* Az egységtesztelés fontos része a fejlesztésnek, amelynek célja a kódminőség javítása és a hibák kiszűrése.
* A JavaScriptben számos egységtesztelési keretrendszer áll rendelkezésre, például a Jest, a Mocha és az Ava, amelyek segítségével könnyen teszteket írhatunk és futtathatunk.
* Az egységtesztek biztosítják, hogy a kódunk várt eredményeket produkáljon különféle bemeneti értékek mellett, és segítenek megelőzni a visszaeséseket és hibákat a későbbi fejlesztés során.

**9.Teljesítmény és optimalizáció:**

* A JavaScript teljesítményének optimalizálása fontos tényező a webalkalmazások gyors és hatékony működése érdekében.
* A kód optimalizálása, a cache használata, a felesleges hívások és erőforrások kiszűrése, valamint a böngészők támogatott funkcióinak kihasználása mind hozzájárulhatnak a jobb teljesítményhez.
* A teljesítményproblémák felismerése és megoldása kulcsfontosságú a felhasználói élmény javítása és a weboldalak gyors betöltése érdekében.

**10.Biztonság és sérülékenységek:**

* A JavaScript alkalmazások biztonsága kritikus fontosságú, mivel azok sérülékenyek lehetnek különféle támadásokkal szemben, például XSS (Cross-Site Scripting) és CSRF (Cross-Site Request Forgery).
* A biztonsági intézkedések közé tartozik a bemenetek ellenőrzése, a biztonságos adattárolás, az autentikáció és az autorizáció megvalósítása, valamint a biztonságos kommunikáció és hálózati kérések kezelése.
* A sérülékenységek és hibák folyamatos monitorozása és javítása elengedhetetlen a biztonságos és megbízható JavaScript alkalmazások fenntartásához.