https://github.com/learning-zone/html-interview-questions

A localStorage hibát dob a maximális határérték elérése után?

Yes

Example:

```
<!DOCTYPE HTML>
<html>
   <head>
         <title>HTML5 localStorage</title>
   </head>
   <body>
      <script type="text/javascript">
        try{
            if(window.localStorage){ // Check if the localStorage object exists

                var result = "";
                var characters  = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
                var charactersLength = characters.length;
                for(var i = 0; i < 10000; i++){
                    result += characters.charAt(Math.floor(Math.random() * charactersLength));
                    localStorage.setItem("key"+i, result);
                }
            } else {
                alert("Sorry, your browser do not support localStorage.");
            }
        } catch(e) {
            console.log('Exception: '+e);
        }
      </script>
   </body>
</html>
```

Output

```
Exception: QuotaExceededError: Failed to execute 'setItem' on 'Storage':
           Setting the value of 'key3230' exceeded the quota.
```

Melyek az új űrlapelemek a HTML5-ben?

Öt új űrlapelem van a HTML5 űrlap-specifikációban: <datalist>, <output>, <keygen>, <progress> és <meter>.

1.) Datalist Tag: lehetővé teszi a javaslatok listájának csatolását egy szövegbeviteli elemhez. Amint a felhasználó elkezdi beírni a szövegmezőt, megjelenik a javaslatok listája, és a felhasználó az egérrel választhat a javaslatok közül.

```
<p>Enter your favorite browser name:</p>
<input type="text" list="browsers" name="favorite_browser">
<datalist id="browsers">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Internet Explorer">
    <option value="Opera">
    <option value="Safari">
</datalist>
```

2.) Meter tag: egy számértéket jelez, amely egy tartományba esik. A címke számos attribútumot támogat:

érték: Ha nem ad meg értéket, akkor a <meter> </meter> pár első számértéke lesz az érték.

- max: Az elem maximálisan lehetséges értéke.
- min: Az elem lehető legkisebb értéke.
- high: Ha az érték tartományként definiálható, akkor ez a tartomány felső vége.
- low: Ha az érték tartományként definiálható, akkor ez a tartomány alsó vége.
- optimum: Az elem optimális értéke.

```
<p>Disk Usage: <meter value="0.2">20%</meter></p>

<p>Total Score: <meter value="6" min="0" max="10">6 out of 10</meter></p>

<p>Pollution Level: <meter low="60" high="80" max="100" value="85">Very High</meter></p>
```

3.) Output Tag: a szöveges kimenet megjelenítésére szolgál. Jelzi az oldal azon részét, amelyet egy szkript (általában JavaScript) módosíthat.

```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="a" value="50"> +
  <input type="number" id="b" value="100"> =
  <output name="result" for="a b"></output>
</form>
```

4.) Progress Tag: azt jelzi, hogy egy feladat mekkora részét hajtotta végre (gyakran százalékosan jelölve). Várhatóan JavaScript kód segítségével módosul.

```
<p>Progress: <progress id="bar" value="0" max="100"><span>0</span>%</progress></p>

<script type="text/javascript">
    var i = 0;
    var progressBar = document.getElementById("bar");

    function countNumbers() {
      if(i < 100) {
        i = i + 1;
        progressBar.value = i;
        // For browsers that don't support progress tag
        progressBar.getElementsByTagName("span")[0].textContent = i;
      }

      // Wait for sometime before running this script again
      setTimeout("countNumbers()", 100);
    }
    countNumbers();
</script>
```

5.) Keygen Tag: A <keygen> elem létrehoz egy titkosítási kulcsot a titkosított adatok szerverhez továbbításához. Ha HTML-űrlapot küld be, a böngésző létrehoz egy kulcspárt, a magánkulcsot a böngésző helyi kulcstárolójában tárolja, és elküldi a nyilvános kulcsot a szervernek.

```
<form action="process-key.php" method="post">
    <label>Username: <input type="text" name="username"></label>
    <label>Encryption: <keygen name="key"></label>
    <input type="submit" value="Submit">
</form>
```

Mi az a DOM? Hogyan működik a DOM?

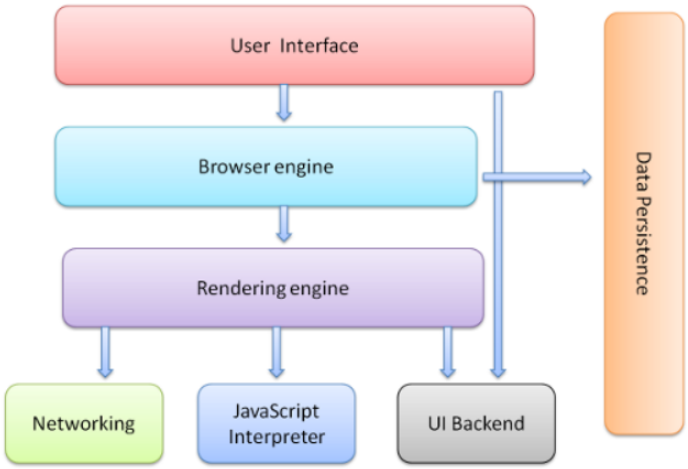
A DOM (Document Object Model) egy cross-platform API, amely a HTML dokumentumokat csomópontokból álló fa struktúraként kezeli. Ezek a csomópontok (például elemek és szöveges csomópontok) olyan objektumok, amelyek programozottan kezelhetők, és a rajtuk végrehajtott bármilyen látható változás élőben tükröződik a dokumentumban. Egy böngészőben ez az API elérhető a JavaScript számára, ahol a DOM csomópontok manipulálhatók stílusuk, tartalmuk, elhelyezésük megváltoztatására a dokumentumban, vagy az event listener-eken keresztül interakcióba léphetnek velük.

- A DOM-ot úgy tervezték, hogy független legyen egy adott programozási nyelvtől, és a dokumentum strukturális ábrázolását egyetlen, következetes API-ból tette elérhetővé.
- A document.getElementById () és a document.querySelector () a DOM-csomópontok kiválasztásának általános funkciói.
- Az innerHTML tulajdonság új értékre állítása futtatja a karakterláncot a HTML elemzőn keresztül, és egyszerű módon kínál dinamikus HTML tartalmat egy csomóponthoz.

Hogyan működik a böngésző renderelő motorja?

A tartalom megjelenítéséhez a böngészőnek számos lépést kell végrehajtania:

- Document Object Model(DOM)
- CSS object model(CSSOM)
- Render Tree
- Layout
- Paint



Mire való egy <DOCTYPE html>?

A DOCTYPE mindig társul egy DTD-hez (Document Type Definition). A DTD meghatározza, hogy egy bizonyos típusú dokumentumok hogyan legyenek felépítve (azaz egy gomb tartalmazhat spanot, de nem div-t), míg a DOCTYPE kijelenti, hogy a dokumentum milyen DTD-t használ. Weboldalak esetén a DOCTYPE deklaráció szükséges. Arra szolgál, hogy megmondja a felhasználói kliensnek, hogy a dokumentum melyik HTML-specifikáció verzióját használja.

Miután a kliens felismerte a helyes DOCTYPE-t, elindítja a DOCTYPE-hez illeszkedő no-quirks módot a dokumentum olvasásakor. Ha egy kliens nem ismeri fel a helyes DOCTYPE-t, akkor a quirks módot indítja el.

Mi történik, ha a DOCTYPE nem adott?

A weboldal quirks módban jelenik meg. A webböngésző motorjai quirks módot használnak a régebbi böngészők támogatásához, amely nem felel meg a W3C specifikációinak. Quirks módban a CSS osztály- és azonosítónevek nem érzékenyek a kis- és nagybetűkre. Normál üzemmódban a kis- és nagybetűk különböznek egymástól.

Mi a különbség a normál mód és a quirks mód között?

Quirks módban az elrendezés a szokásos viselkedést emulálja a Navigator 4 és az Internet Explorer 5 böngészőben. Ez elengedhetetlen a webes szabványok széleskörű elterjedése előtt épített webhelyek támogatásához. Normál módban a viselkedést a HTML és CSS specifikációk írják le.

HTML dokumentumokhoz a böngészők a dokumentum elején egy <! DOCTYPE html> fájlt használnak annak eldöntésére, hogy quirks vagy standard módban kezeljék-e.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset=UTF-8>
    <title>Hello World!</title>
  </head>
  <body>
  </body>
</html>
```

Mi a különbség a HTML és az XHTML között?

Az Extensible Hypertext Markup Language (XHTML) két fontos megjegyzéssel rendelkezik a frontend fejlesztők számára.

- Jól kell kialakítani, vagyis minden elemet le kell zárni és helyesen be kell ágyazni, különben hibákat ad vissza.
- Mivel szigorúbb, mint a HTML, ezért a böngésző kevesebb előfeldolgozást igényel, ami javíthatja webhelye teljesítményét.

Melyek a HTML5 építőkövei?

- Szemantika: lehetővé teszi, hogy pontosabban leírja, mi a tartalma.
- Kapcsolódás: új és innovatív módon kommunikálhat a szerverrel.
- Offline és tárolás: lehetővé teszi a weboldalak számára, hogy az adatokat a kliens oldalon helyben tárolják, és hatékonyabban működjenek offline módban.
- Multimédia: a videó és a hang első osztályú polgárai az Open Weben.
- 2D / 3D grafika és effektusok: a bemutatási lehetőségek sokkal változatosabb tartományának lehetővé tétele.
- Teljesítmény és integráció: nagyobb sebesség optimalizálás és a számítógépes hardver jobb használata.
- Eszköz hozzáférés: lehetővé teszi különféle bemeneti és kimeneti eszközök használatát.
- Stílus: hagyjuk, hogy a szerzők kifinomultabb témákat írjanak.

Írja le a különbséget a cookie, a sessionStorage és a localStorage között?

cookie: A felhasználói számítógépre mentett szövegfájl az adatok tárolásához és visszakereséséhez

sessionStorage: memória a böngészőben az ideiglenes adatok mentéséhez, amíg az ablak vagy tab be nem záródik.

localStorage: Mint a süti, ahol az adatokat a böngésző munkamenetei után el lehet menteni és visszakeresni, de a memóriában tárol, mint például a sessionStorage. Az adatokat egyszerű kulcs-érték párokként tárolják, és Json objektumokként tárolhatják..

	`cookie`	`localStorage`	`sessionStorage`
Kezdeményező	Kliens vagy szerver. A szerver használhatja a Set-Cookie fejlécet	Client	Client
Lejárat	Kézzel állítva	Örökké	A tab bezárása
A böngésző munkamenetei között állandó	Attól függ, hogy van-e beállítva a lejárat	Igen	Nem
Minden HTTP-kéréssel elküldve a szerverre	A sütik küldése automatikusan a Cookie fejlécen keresztül történik	Nem	Nem
Kapacitás (domainenként)	4kb	5 MB	5 MB
Megközelíthetőség	Bármelyik ablak	Bármelyik ablak	Ugyanaz a tab

Megjegyzés: Ha a felhasználó úgy dönt, hogy a böngésző adatait a böngésző által biztosított bármely mechanizmuson keresztül törli, akkor ezzel minden tárolt cookie, localStorage vagy sessionStorage törlődik. Fontos ezt szem előtt tartani a helyi perzisztencia megtervezésekor, különösen akkor, ha összehasonlítjuk olyan alternatívákkal, mint például a kiszolgálóoldali adatbázis-tárolás vagy hasonló (ami természetesen a felhasználói műveletek ellenére is fennmarad).

Mi a kritikus megjelenítési útvonal (Critical Rendering Path)?

- A DOM fa felépítése
- A CSSOM fa felépítése
- JavaScript futtatása - elemző blokkoló erőforrás
- A Render Tree létrehozása
- Az elrendezés létrehozása
- Kirajzolás (Painting)

Milyen előnyei vannak a Server Side Rendering (SSR)-nek a Client Side Rendering (CSR)-el szemben?

- A kiszolgálóoldali megjelenítést két okból használjuk:
    - teljesítményelőny kliensek számára
    - Állandó SEO teljesítmény
- A fő különbség az, hogy az SSR-re a szerver válasza a böngészőre az oldal HTML-je, amely készen áll a megjelenítésre, míg CSR-re a böngésző egy üres dokumentumot kap, amely linkeket tartalmaz a javascriptedre. Ez azt jelenti, hogy az SSR esetében a böngésző elkezdi a HTML-szolgáltatást a szerveréről anélkül, hogy megvárná az összes JavaScript letöltését és végrehajtását.
- SSR esetén a felhasználó elkezdheti megtekinteni az oldalt, amíg mindez megtörténik. A CSR világához meg kell várni, amíg a fentiek mind bekövetkeznek, majd a virtuális dom-ot át kell helyezni a böngésző doménjébe, hogy az oldal megtekinthető legyen.

Mi a különbség a és a között?

- A egy blokkszintű elem, ami azt jelenti, hogy a saját vonalon jeleníti meg, a szülőelem 100% -os szélességével.
- A egy soros elem, ami azt jelenti, hogy ugyanazon a vonalon jelenik meg, mint az előző elem, és szélességét a tartalma határozza meg.

Nevezzen meg 5 általános blokkszintű és inline HTML elemet?

- block elements <h1>, <p>, <ul>, <ol>, <li>,
- inline elements , <a>, , <i>, <img>

Mik azok a szemantikai és nem szemantikai elemek?

- Szemantikai elemek: világosan leírja jelentését mind a böngésző, mind a fejlesztő számára. Például: <form>, <table>, <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header>, <main>, <mark>, <nav>, <section>, <summary>, <time> egyértelműen meghatározza annak tartalmát.
- Nem szemantikai elemek: A és a nem mond semmit a tartalmáról.

Mi a main elem célja?

A HTML <main> elem a dokumentum <body> domináns tartalmát képviseli. A main tartalmi terület olyan tartalomból áll, amely közvetlenül kapcsolódik a dokumentum központi témájához, vagy egy alkalmazás központi funkcióihoz, vagy kibővíti azt.

```
<main role="main">
    <p>Geckos are a group of usually small, usually nocturnal lizards.
       They are found on every continent except Australia.</p>
    <p>Many species of gecko have adhesive toe pads which enable them to climb walls and even
windows.</p>
</main>
```

Megjegyzés: A dokumentumban nem lehet több, mint egy <main> elem, amelynél nincs megadva a rejtett (hidden) attribútum.

Definiálja a szemantikai jelölést. Mit jelentenek a <section>, <article>, <aside>, <nav>, <header>, <footer> szemantikai jelentése és mikor / hogyan kell felhasználni mindegyiket a html jelölés strukturálásában?

- A <header> bevezető és navigációs információkat tartalmaz az oldal egy szakaszáról. Ez tartalmazhatja a szakasz címsorát, a szerző nevét, a megjelenés időpontját és dátumát, a tartalomjegyzéket vagy más navigációs információkat.
- Az <article> egy önálló kompozíciót hivatott elhelyezni, amelyet logikailag önállóan lehet újra létrehozni az oldalon kívül anélkül, hogy elveszítené a tartalmát. Jó példák az egyes blogbejegyzések vagy hírek.
- A <section> egy rugalmas tároló olyan tartalom tárolására, amely közös információs témával vagy céllal rendelkezik.
- A <footer> olyan információk tárolására szolgál, amelyeknek meg kell jelenniük a tartalom egy részének végén, és tartalmazniuk kell további információkat a szakaszról. A szerző neve, a szerzői jogi információk és a kapcsolódó linkek tipikus példák az ilyen tartalomra.

Mikor érdemes használni a section-t, div-t vagy article-t?

- A <section>, a belső tartalomcsoport egyetlen témához kapcsolódik, és bejegyzésként kell megjelennie az oldal körvonalában. Ez egy csomó kapcsolódó tartalom, például egy hosszú cikk alrésze, az oldal nagy része (pl. A hírek szakasz a kezdőlapon), vagy egy oldal a webapp füles felületén. Egy szakasznak általában van címsora (címe), és lehet, hogy lábléc is.
- Az <article> egy teljes, vagy önálló összetételt képvisel egy dokumentumban, oldalon, alkalmazásban vagy webhelyen, és elvileg függetlenül terjeszthető vagy újrafelhasználható. Ez lehet fórumbejegyzés, magazin vagy újságcikk, blogbejegyzés, felhasználó által beküldött megjegyzés, interaktív eszköz vagy bármilyen más független tartalmi elem.
- A viszont semmilyen jelentést nem közvetít, az osztályán, a lang és a cím attribútumain kívül.

Mi az a karakterkódolás?

A karakterkódolás a bájtok karakterekké történő átalakításának módszere. A HTML-dokumentum megfelelő érvényesítéséhez vagy megjelenítéséhez a programnak megfelelő karakterkódolást kell választania. Ezt tag határozza meg:

```
<meta charset="utf-8"/>
```

UTF-8: Unicode Translation Format, amely 8 bites egységekben, azaz bájtokban érkezik. Egy karakter az UTF8-ban 1 és 4 bájt hosszú lehet, így az UTF8 változó szélességű lehet.

Mi a célja a meta tag-eknek?

A META elemek felhasználhatók a HTML dokumentum tulajdonságait leíró név / érték párok, például szerző, lejárati dátum, kulcsszólista, dokumentum szerző stb.

```
<!DOCTYPE html>
<html>
  <head>
        <!--Recommended Meta Tags-->
        <meta charset="utf-8">
        <meta name="language" content="english">
        <meta http-equiv="content-type" content="text/html">
        <meta name="author" content="Author Name">
        <meta name="designer" content="Designer Name">
        <meta name="publisher" content="Publisher Name">
        <meta name="no-email-collection" content="name@email.com">
        <meta http-equiv="X-UA-Compatible" content="IE=edge"/>

        <!--Search Engine Optimization Meta Tags-->
        <meta name="description" content="Project Description">
        <meta name="keywords" content="Software Engineer,Product Manager,Project Manager,Data
Scientist">
        <meta name="robots" content="index,follow">
        <meta name="revisit-after" content="7 days">
        <meta name="distribution" content="web">
        <meta name="robots" content="noodp">

        <!--Optional Meta Tags-->
        <meta name="distribution" content="web">
        <meta name="web_author" content="">
        <meta name="rating" content="">
        <meta name="subject" content="Personal">
```

```
        <meta name="title" content=" - Official Website.">
        <meta name="copyright" content="Copyright 2020">
        <meta name="reply-to" content="">
        <meta name="abstract" content="">
        <meta name="city" content="Bangalore">
        <meta name="country" content="INDIA">
        <meta name="distribution" content="">
        <meta name="classification" content="">

        <!--Meta Tags for HTML pages on Mobile-->
        <meta name="format-detection" content="telephone=yes"/>
        <meta name="HandheldFriendly" content="true"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <meta name="apple-mobile-web-app-capable" content="yes" />

        <!--http-equiv Tags-->
        <meta http-equiv="Content-Style-Type" content="text/css">
        <meta http-equiv="Content-Script-Type" content="text/javascript">

    <title>HTML5 Meta Tags</title>
  </head>
  <body>
      ...
  </body>
</html>
```

Mire utal az async és a defer a szkriptcímkében? Írja le a különbséget a <script>, a <script async> és a <script defer> között

Async: Letölti a parancsfájlt a HTML elemzése során, és a letöltés befejezése után szünetelteti a HTML elemző végrehajtását.

Defer: A Defer letölti a parancsfájlt a HTML-elemzés során, és csak a HTML-elemző befejezése után hajtja végre. Nem minden böngésző támogatja ezt.

Az async attribútum arra szolgál, hogy jelezze a böngésző számára, hogy a szkriptfájl aszinkron futtatható. A HTML elemzőnek nem kell szünetelnie abban a pillanatban, amikor a szkript címkéjéhez eljut a letöltéshez és végrehajtáshoz, a végrehajtás akkor történhet, amikor a szkript készen áll.

A defer attribútum azt mondja a böngészőnek, hogy csak a HTML-dokumentum teljes elemzése után hajtsa végre a szkriptfájlt.

Mi a célja a cache busting (gyorsítótár-törlés)-nak, és hogyan érheti el?

A böngészőknek van egy gyorsítótáruk a fájlok ideiglenes tárolásához a webhelyeken, így az oldalak közötti váltáskor vagy ugyanazon oldal újratöltésekor nem kell őket újra letölteni. A szerver úgy van beállítva, hogy fejléceket küldjön, amelyek utasítják a böngészőt, hogy a fájlt adott ideig tárolja. Ez nagymértékben növeli a weboldal sebességét és megőrzi a sávszélességet.

Ugyanakkor problémákat okozhat, amikor a fejlesztők megváltoztatták a webhelyet, mert a felhasználó gyorsítótárában továbbra is a régi fájlokra hivatkoznak. Ez vagy régi funkciókat hagyhat, vagy megszakíthat egy webhelyet, ha a gyorsítótárazott CSS és a JavaScript fájlok olyan elemekre hivatkoznak, amelyek már nem léteznek, áthelyezték őket vagy átnevezték őket.

A gyorsítótár-törlés az a folyamat, amely arra kényszeríti a böngészőt, hogy töltse le az új fájlokat. Ez úgy történik, hogy a fájlt valami másnak nevezi el, mint a régi fájl.

A böngésző újbóli letöltésére kényszerített általános technika az, hogy egy lekérdezési karakterláncot (query string) fűznek a fájl végéhez.

```
<!-- src="js/script.js" => src="js/script.js?v=2" -->
<script src="js/script.js?v=2"></script>
```

A böngésző más fájlnak tekinti, de megakadályozza a fájlnév módosítását.

Nevezzen meg 3 módszert az oldal betöltésének csökkentésére?

1. LocalStorage
2. Caching resources
3. DNS-prefetch (sample below)
4. Keep resources on a CDN

Mik az ARIA és a képernyőolvasók, és hogyan lehet egy weboldalt elérhetővé tenni?

A képernyőolvasók olyan szoftverprogramok, amelyek olyan segítő technológiákat kínálnak, amelyek lehetővé teszik a fogyatékossággal élő emberek számára (például látás, hang vagy egérgátlás nélkül) webalkalmazások használatát. Hozzáférhetőbbé teheti webhelyeit az ARIA szabványok, például a szemantikus HTML, az alt attribútumok követésével és a [role = button] használatával a várt módon

Mi a célja az alt attribútumnak az image-en?

Az alt attribútum alternatív információt nyújt a képhez, ha a felhasználó nem tudja megtekinteni. Az alt attribútumot kell használni minden olyan kép leírására, amely csak díszítő célokat szolgál, ebben az esetben üresen kell hagyni.

Magyarázza el a CSS-animációk és a JavaScript-animációk előnyeit és hátrányait?

- A CSS animációk lehetővé teszik a böngésző számára, hogy kiválassza az animáció feldolgozásának helyét, a CPU-t vagy a GPU-t. (Central or Graphics Processing Unit)
- Ez azt jelenti, hogy sok réteg hozzáadása egy dokumentumhoz végül teljesítménysikereket fog elérni.
- A JS animáció több kódot jelent a felhasználó számára letöltésre és a fejlesztő fenntartására.
- Egy elemre több animációtípus alkalmazása nehezebb a CSS-sel, mivel az összes transzformációs teljesítmény egy tulajdonságtranszformában van
- A deklaratív CSS animációk nem programozhatók, ezért korlátozott a képességük.

Mit képvisel a CORS és milyen kérdéssel foglalkozik?

A Cross-Origin Resource Sharing (CORS) egy W3C specifikáció, amely lehetővé teszi a tartományok közötti (cross-domain) kommunikációt a böngészőből. Az XMLHttpRequest objektum tetejére építve a CORS lehetővé teszi a fejlesztők számára, hogy ugyanazokkal az idiómákkal dolgozzanak, mint az azonos tartományú kérések. A CORS a webszervereknek tartományok közötti hozzáférés-vezérlést biztosít, amelyek lehetővé teszik a tartományok közötti (cross-domain) biztonságos adatátvitelt.

A weboldal teljesítményének javításának módjai

- Minimalizálja a HTTP kéréseket
    - A webhelyek főleg lassúak a túl sok (vagy túl nagy) HTTP kérés miatt. Megszüntethetjük a felesleges kéréseket;
        - kombinált fájlok: js fájlba, css fájlba
        - CSS sprites: A CSS Sprites az előnyös módszer a kép-kérelmek számának csökkentésére. Egyesítse a háttérképeket egyetlen képpé, és használja a CSS background-image and background-position tulajdonságait a kívánt képszegmens megjelenítéséhez.
- Használjon Content Delivery Network CDN-t
    - A CDN lényegében számos optimalizált szerver a világon, amely földrajzi elhelyezkedése alapján webtartalmat szállít a felhasználóknak. Ez nagy teljesítménybeli javulást jelent a webhely felhasználói számára. Mert mondjuk, ha egy személy Indiában látogat el a webhelyére, akkor a közeli szerverről fogja lekérni a webtartalmat
- Képek optimalizálása:
    - a képméretek hatalmas különbséget jelentenek a webhely sebességében. Minél nagyobb a tartalom / képek, annál lassabb az oldal. tudnánk:
        - A felbontás megváltoztatása: a kép (és ezáltal a fájlméret) „minőségének” csökkentése
        - Kép tömörítése: a képadatok tárolásának hatékonyságának növelése
        - A kép kivágása: kivágáskor szükségtelen területeket vág ki, és így kisebb méretűvé teszi a képet
- Helyezze a szkripteket az aljára:
    - A Javascript fájlok az oldal többi része után tölthetők be. A legegyszerűbb megoldás az, ha a külső Javascript-fájlokat az oldal aljára helyezi, közvetlenül a body tag bezárása előtt. Most webhelye többi része betöltődhet a szkriptek előtt. Egy másik módszer, amely még nagyobb ellenőrzést tesz lehetővé, a defer vagy async attribútumok használata külső .js fájlok webhelyén történő elhelyezésekor.
        - Az aszinkron tag-ek betöltik a szkripteket, miközben az oldal többi része betöltődik, de ez azt jelenti, hogy a parancsfájlok soron kívül tölthetők be. Alapvetően a könnyebb fájlok töltődnek be először. Ez egyes szkriptek számára jó lehet, mások számára katasztrofális lehet.

        - A defer attribútum a tartalom betöltése után tölti be a szkripteket. A szkripteket is sorrendben futtatja.
- Adjon hozzá egy lejárati vagy egy gyorsítótár-vezérlő fejlécet (Expires or a Cache-Control Header)
    - A weblaptervek egyre gazdagabbak, ami több szkriptet, stíluslapot, képet és Flash-t jelent az oldalon. Lehet, hogy az oldal első látogatójának több HTTP kérést kell elküldenie, de a Expires fejléc használatával ezeket az összetevőket gyorsítótára lehet tenni. Ezzel elkerülhetők a felesleges HTTP-kérések a későbbi oldal megtekintéseknél. A lejárt fejléceket leggyakrabban a képekhez használják, de azokat minden összetevőnél fel kell használni, beleértve a szkripteket, a stíluslapokat és a Flash-összetevőket.
- Gzip Components
    - A tömörítés csökkenti a válaszidőket a HTTP válasz méretének csökkentésével. A gzip-elés általában körülbelül 70% -kal csökkenti a válasz méretét.
- Helyezze a stíluslapokat a tetejére:
    - Ez azért van, mert a stíluslapok elhelyezése a HEAD-ban lehetővé teszi az oldal fokozatos megjelenítését.
- Kerülje a CSS kifejezéseket
- A GET használata az AJAX kérésekhez:
    - Az Ajax az, hogy azonnali visszajelzést ad a felhasználónak, mivel aszinkron módon kéri az információkat a háttér-webszervertől
- Tedd külsővé a JavaScriptet és a CSS-t:
    - Külső fájlok használata a való világban általában gyorsabb oldalakat eredményez, mert a JavaScript és CSS fájlokat a böngésző tárolja. A HTML dokumentumokba beágyazott JavaScript és CSS minden alkalommal letöltésre kerül, amikor a HTML dokumentumot igénylik. Ez csökkenti a szükséges HTTP kérések számát, de megnöveli a HTML dokumentum méretét. Másrészt, ha a JavaScript és a CSS a böngésző által tárolt külső fájlokban található, akkor a HTML-dokumentum mérete csökken a HTTP-kérelmek számának növelése nélkül.
- Használja a get-et az ajax kéréshez:
    - A POST a böngészőkben két lépésből áll: először a fejléceket, majd az adatokat küldi el. Ezért a legjobb, ha a GET-et használjuk, amelynek elküldéséhez csak egy TCP-csomag szükséges (hacsak nincs sok cookie-ja).
- Ne 404s:
    - A HTTP kérések drágák, ezért a HTTP kérés és a haszontalan válasz (azaz a 404 nem található) megszerzése teljesen felesleges, és minden előny nélkül lassítja a felhasználói élményt.
- Csökkentse a sütik méretét:
    - A HTTP cookie-kat számos okból használják, például hitelesítéshez és személyre szabáshoz. A sütikről szóló információkat a HTTP fejlécekben cserélik a webszerverek és a böngészők. Fontos, hogy a sütik mérete a lehető legkisebb legyen, hogy minimalizáljuk a felhasználó válaszidejére gyakorolt hatást.
- Csökkentse a DNS-kereséseket
- Minify-olja a JavaScriptet és a CSS-t

- Kerülje az átirányításokat
- Távolítsa el az ismétlődő scipt-eket
- Konfigurálja az Etags-t
- Tegye az Ajax cache-elhetővé
- Post-load Components
- Preload Components
- Csökkentse a DOM-elemek számát
- Minimalizálja az iframe-ek számát
- Minimalizálja a DOM-hozzáférést
- A CSS Sprites optimalizálása
- Ne méretezze a képeket HTML-ben
- Készítse el a favicon.ico-t kicsi-re és cache-elhetővé
- Kerülje az üres Image src alkalmazását

Az olyan böngészőmotorok összehasonlítása, mint a Chrome, Firefox, Internet Explorer, Safari?

- Chrome:
    - Layout rendering engine **Webkit**.
    - JavaScript engine **V8**
- Firefox:
    - Layout rendering engine **Gecko**.
    - JavaScript engine **Spider monkey**
- Internet explorer:
    - Layout rendering engine **Trident**.
    - JavaScript engine **Chakra**
- Safari:
    - Layout rendering engine **Webkit**.
    - JavaScript engine JavascriptCore i.e **Nitro**

Mit csinál a lang attribútum a html-ben?

Segít az oldalak stílusában azáltal, hogy css-ben használja őket: css:lang () pseudo class-al. Helyesírási és nyelvtani ellenőrzők.

Mi az desktop first és mobile first tervezési megközelítés?

Desktop-first: Általános választók és stílusok, amelyek célja, hogy a webhely jól nézzen ki a globálisan definiált DESKTOP képernyőkön. De ezek minden eszközt érintenek, és felül kell írni őket a minimális képernyőméretet megcélzó maximális szélességű média lekérdezésekkel (max-width media queries)

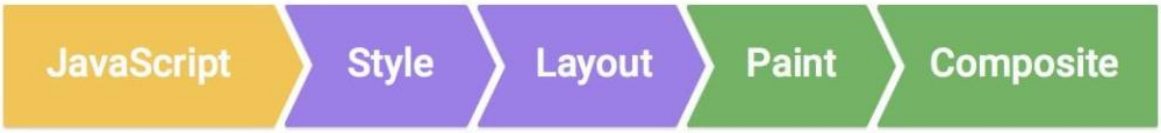
Mobile-first: Ide kerülnek azok az általános választók és stílusok, amelyek célja, hogy a webhely jól nézzen ki a kis MOBILE képernyőkön. De ezek minden eszközt érintenek, és a maximális képernyő méretet megcélzó min szélességű média lekérdezések (min-width media queries) felül kell őket írni

Desktop-first megközelítés esetén a média lekérdezéseket a max-width, míg a mobil első megközelítésben a media lekérdezéseket a min-width vonatkozásában írjuk.

Mire jók az data attribútumok?

A HTML5 data attribútummal egyedi adatokat rendelhet hozzá egy elemhez. Amikor több információt / adatot akarunk tárolni az elemről, ha nincs megfelelő HTML5 elem vagy attribútum.

Magyarázza el a különbséget az layout, painting és compositing között?



JavaScript: Általában a JavaScript-et használják olyan munkák kezelésére, amelyek vizuális változásokat eredményeznek, legyen szó a jQuery animációs funkciójáról, egy adatkészlet rendezéséről vagy DOM-elemek hozzáadásáról az oldalra. Ennek azonban nem feltétlenül a vizuális változást kiváltó JavaScript-nek kell lennie: a CSS-animációkat, az Transitions-ot és a Web Animations API-t is gyakran használják.

Stílusszámítások: Ez annak a folyamata, hogy kitaláljuk, melyik CSS-szabályok vonatkoznak mely elemekre az egyező szelektorok alapján, például .headline vagy .nav> .nav__item. Innentől kezdve, ha ismertek a szabályok, azokat alkalmazzák, és kiszámítják az egyes elemek végső stílusait.

Elrendezés: Amint a böngésző tudja, mely szabályok vonatkoznak egy elemre, elkezdheti kiszámolni, hogy mennyi helyet foglal és hol van a képernyőn. Az elrendezési modellje azt jelenti, hogy az egyik elem hatással lehet másokra, például az elem szélessége jellemzően befolyásolja gyermekei szélességét, és így tovább a fán felfelé és lefelé, így a folyamat meglehetősen érintett lehet a böngésző számára.

Paint: A festés a pixelek kitöltésének folyamata. Szöveget, színeket, képeket, szegélyeket és árnyékokat rajzol ki, lényegében az elemek minden vizuális részét. A rajz általában több felületre készül, amelyeket gyakran rétegeknek neveznek.

Összetétel: Mivel az oldal egyes részeit potenciálisan több rétegbe rajzolták, a megfelelő sorrendben kell a képernyőhöz húzni őket, hogy az oldal megfelelően jelenjen meg. Ez különösen fontos a

másikat átfedő elemek esetében, mivel egy hiba azt eredményezheti, hogy az egyik elem helytelenül jelenik meg a másik tetején.

HTML Canvas?

A canvas egy HTML elem, amellyel grafikákat lehet rajzolni JavaScript segítségével. Ez felhasználható például grafikonok rajzolására, fotók kombinálására vagy animációk készítésére.

Colors, Styles, and Shadows

Property	Description
fillStyle	Beállítja vagy visszaadja a rajz kitöltéséhez használt színt, színátmenetet vagy mintát
strokeStyle	Beállítja vagy visszaadja az ecsetvonásokhoz használt színt, színátmenetet vagy mintát
shadowColor	Az árnyékhoz használt szín beállítása vagy visszaadása
shadowBlur	Beállítja vagy visszaadja az árnyékok elmosódási szintjét
shadowOffsetX	Beállítja vagy visszaadja az árnyék vízszintes távolságát az alakzattól
shadowOffsetY	Beállítja vagy visszaadja az árnyék függőleges távolságát az alakzattól

Line Styles

Property	Description
lineCap	Beállítja vagy visszaadja a vonal végződésének stílusát
lineJoin	Beállítja vagy visszaadja a létrehozott sarok típusát, amikor két vonal találkozik
lineWidth	Beállítja vagy visszaadja az aktuális vonalszélességet
miterLimit	Beállítja vagy visszaadja a maximális miter hosszat

Rectangles

Method	Description
rect()	Téglalapot hoz létre

Method	Description
fillRect()	"Töltött" téglalapot rajzol
strokeRect()	Rajzol egy téglalapot (nincs kitöltés)
clearRect()	Törli a megadott pixeleket egy adott téglalapon belül

Paths

Method	Description
fill()	Kitölti az aktuális rajzot (útvonal)
stroke()	Valójában megrajzolja az Ön által meghatározott utat
beginPath()	Útvonalat kezd, vagy visszaállítja az aktuális útvonalat
moveTo()	Az útvonalat a vászon meghatározott pontjához mozgatja, vonal létrehozása nélkül
closePath()	Útvonalat hoz létre az aktuális ponttól a kiindulópontig
lineTo()	Új pontot ad hozzá, és egy vonalat hoz létre ahhoz a ponthoz a vászon utolsó megadott pontjától
clip()	Kivág bármilyen formájú és méretű régiót az eredeti vászonról
arc()	Ívet / görbét hoz létre (körök vagy körrészek létrehozására használják)
arcTo()	Ívet / görbét hoz létre két érintő között

Transformations

Method	Description
scale()	Az aktuális rajzot nagyobbra vagy kisebbre méretezheti
rotate()	Forgatja az aktuális rajzot
translate()	Áthelyezi a (0,0) pozíciót a vásznon
transform()	Helyettesíti a rajz aktuális transzformációs mátrixát

Method	Description
setTransform()	Visszaállítja az aktuális transzformációt az identitásmátrixra. Ezután futtatja a transform () függvényt

Text

Property	Description
font	Beállítja vagy visszaadja a szöveges tartalom aktuális betűtípus-tulajdonságait
textAlign	Beállítja vagy visszaadja a szöveges tartalom aktuális igazítását
textBaseline	Beállítja vagy visszaadja a szöveg rajzolásakor használt aktuális szöveg alapvonalát
fillText()	"Töltött" szöveget rajzol a vásznra
strokeText()	Szöveget rajzol a vászonra (nincs kitöltés)
measureText()	Olyan objektumot ad vissza, amely tartalmazza a megadott szöveg szélességét

a böngészők által használt HTML Layout Engine-ek

Engine	Status	Embedded in
WebKit	Active	Safari browser, plus all browsers hosted on the iOS App Store
Blink	Active	Google Chrome and all other Chromium-based browsers like Opera and Microsoft Edge
Gecko	Active	Firefox browser and Thunderbird email client, plus forks like SeaMonkey and Waterfox
KHTML	Discontinued	Konqueror browser
Presto	Discontinued	formerly in the Opera browser
EdgeHTML	Discontinued	formerly in the Microsoft Edge browser
Trident	Discontinued	Internet Explorer browser and Microsoft Outlook email client

Melyek a HTML5-ben elérhető szemantikus tag-ek?

A HTML5 szemantikus tag-ek meghatározzák a szöveg funkcióját és kategóriáját, megkönnyítve ezzel a munkát a böngészők és a keresőmotorok, valamint a fejlesztők számára.

A HTML5 új szemantikai elemeket kínál a weboldal különböző részeinek meghatározásához:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

```
<header></header>
<section>
          <article>
                    <figure>
                              <img>
                              <figcaption></figcaption>
                    </figure>
          </article>
</section>
<footer></footer>
```

Teljes példa:

```
<!DOCTYPE
html>
           <html>
             <head>
                 <meta charset="utf-8">
                 <title>HTML5 Semantic Tags</title>
             </head>
             <body>
               <!-- Header Tag -->
               <header>
                 <h1>JavaScript</h1>
                 <h3>What is JavaScript?</h3>
                 <p>Today we are going to talk about JavaScript</p>
               </header>
               <!-- Nav Tag -->
               <nav>
                   <ul>
                       <li><a href="#">Gamified Courses</a></li>
                       <li><a href="#">Tutorials</a></li>
                       <li><a href="#"">Space doggo courses</a></li>
                       <li><a href="#">Game Dev Courses</a></li>
                   </ul>
               </nav>
```

```
    <!-- Section Tag -->
    <section>
      <h1>Section Heading</h1>
      <p>The section tag can contain any elements.</p>
      <img src="image.png" alt="section example">
    </section>
    <!-- Article Tag -->
    <article>
      <h1>Fun Fact</h1>
      <p>Fun fact: most of the fun facts on the Internet are not actually fun.</p>
    </article>
    <!-- Aside Tag -->
    <aside>
      <h4>Lake</h4>
      <p>Oxford lake is a lake in the state.</p>
    </aside>

    <!-- Details Tag -->
    <details>
      <summary>Copyright</summary>-by Your company All the rights are owned by your company
    </details>

    <!-- Figure Tag -->
    <figure>
      <figcaption>Dog</figcaption>
      <img src="image.png" alt="The Bread Dog" width="300" height="300">
    </figure>
    <!-- Summary Tag -->
    <details>
      <summary>Some details</summary>
      <p>Provide more info about the details here.</p>
    </details>

    <!-- Footer Tag -->
    <footer>
      <address>Postal Address: Door No.00, Street, City, State, Country.</address>
      <p>Copyright © 2018 All rights reserved.</p>
    </footer>
  </body>
</html>
```

Miért érdemes használni a szemantikus tag-et?

Keresőmotor-optimalizálás, hozzáférhetőség, újrafelhasználás, könnyű kód.

Sok látássérült ember bízik a böngésző beszédében, és a szemantikus tagek segítenek az oldal tartalmának egyértelmű értelmezésében.

A keresőmotornak meg kell értenie az oldal tartalmát a rangsoroláshoz, és a szemantikus tagek segítenek.

A szemantikai kód segíti az akadálymentességet. Különösen sok ember, akinek nem jó a szeme, a beszédböngészőkre támaszkodik, ha oldalakat olvas nekik. Ezek a programok csak akkor tudják jól értelmezni az oldalakat, ha világosan meg vannak magyarázva.

Segítsen a keresőmotoroknak az oldalak jobb megértésében. A keresőmotoroknak meg kell érteniük, hogy miről szól a tartalom, amikor megfelelő rangsorolják a keresőmotorokon. A szemantikus kód javítja a keresőmotorokon való elhelyezkedését, mivel a "search engine spiders" könnyebben megértik.

Könnyebb olvasni és szerkeszteni, ami időt és pénzt takarít meg a karbantartás során.

Hogyan lehet az oldalt reszponzívvá tenni?

A reszponzív webdesign arról szól, hogy a HTML és a CSS segítségével automatikusan átméretezheti, elrejtheti, összezsugoríthatja vagy kibővítheti a weboldalakat, hogy azok minden eszközön (asztali számítógépen, táblagépen és telefonon) jól kinézzenek.

1) Setting the viewport

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

2) Responsive Images

Ha a CSS width tulajdonság 100% -ra van állítva, akkor a kép reszponzív és felfelé és lefelé skálázódik

```
<img src="img.png" style="width:100%;">
```

3) Show different Images depending on Browser Width

A HTML <picture> elem segítségével különböző képeket határozhat meg a böngésző különböző ablakméreteihez.

```
<picture>
  <source srcset="img_small.jpg" media="(max-width: 600px)">
  <source srcset="img_large.jpg" media="(max-width: 1500px)">
  <source srcset="img.jpg">
  <img src="img_small.jpg" alt="Image">
</picture>
```

4) Responsive Text Size

A szövegméret egy "vw" egységgel állítható be, ami a " viewport width" jelenti. Így a szövegméret követni fogja a böngészőablak méretét.

```
<h1 style="font-size:10vw">Hello World</h1>
```

5) Media Queries

Media queries segítségével teljesen különböző stílusokat határozhat meg a különböző böngészőméretekhez.

```
/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
```

```
  }
}
```

Mi a különbség a span tag és a div tag között?

Az elsődleges különbség a div és a span között az alapértelmezett viselkedésük. Alapértelmezés szerint a blokkszintű elem, a pedig sorszintű (inline) elem.

```
<div>Demo Text, with <span>some other</span> text.</div>
```

Mi az opcionális záró címke?

<p>, <li>, <td>, <tr>, <th>, <html>, <body> stb. nem kell megadniuk a végcímkét. Amikor a böngésző eléri az új címkét, automatikusan befejezi az előző címkét

Mi az a self closing tag?

A HTML5-ben nem feltétlenül szükséges bizonyos HTML-tag-et bezárni. Azokat a címkéket, amelyekhez nem szükséges külön záró ctag, „self closing tag”-nek nevezzük.

Az self closing tag-re példa például a sortörés (
) vagy a meta tag (<meta>). Ez azt jelenti, hogy a következők egyaránt elfogadhatók:

```
<meta charset="UTF-8">
...
<meta charset="UTF-8" />
```

Mi a különbség az SVG és a Canvas között?

1.) SVG: A skálázható vektorgrafika (Scalable Vector Graphics - SVG) egy XML-alapú képformátum, amelyet kétdimenziós vektoralapú grafika definiálására használnak a web számára. A raszteres képektől (pl. .Jpg, .gif, .png stb.) ellentétben a vektorkép bármilyen mértékben fel- vagy kicsinyíthető a képminőség romlása nélkül.

Az SVG használatának a következő előnyei vannak más képformátumokkal szemben, például JPEG, GIF, PNG stb.

- Az SVG képek kereshetők, indexelhetők, szkriptelhetőek és tömöríthetők.
- Az SVG képek valós időben hozhatók létre és módosíthatók a JavaScript használatával.
- Az SVG képek kiváló minőségben, bármilyen felbontással kirajzolhatóak.
- Az SVG-tartalmak a beépített animációs elemek segítségével animálhatók.
- Az SVG képek hiperhivatkozásokat tartalmazhatnak más dokumentumokra.

Example:

```
<!DOCTYPE html>
<html>
```

```
   <head>
      <style>
         #svgelem {
            position: relative;
            left: 50%;
            -webkit-transform: translateX(-20%);
            -ms-transform: translateX(-20%);
            transform: translateX(-20%);
         }
      </style>
      <title>HTML5 SVG</title>
   </head>
   <body>
      <h2 align="center">HTML5 SVG Circle</h2>
      <svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
         <circle id="bluecircle" cx="60" cy="60" r="50" fill="blue" />
      </svg>
   </body>
</html>
```

2.) Canvas: A Canvas egy HTML elem, amelyet a weboldal grafikáinak rajzolásához használnak. Ez egy bitkép, egy grafikus alkalmazás programozási felülettel (API), amellyel rajzolni lehet rá. A <canvas> elem csak a grafikák konténerje. A grafika megrajzolásához szkriptet kell használnia. A Canvas-nak számos stratégiája van az utak, dobozok, körök, szöveg és képek hozzáadásával kapcsolatban.

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML5 Canvas Tag</title>
   </head>
   <body>
      <canvas id="newCanvas" width="200" height="100" style="border:1px solid #000000;"></canvas>
      <script>
         var c = document.getElementById('newCanvas');
         var ctx = c.getContext('2d');
         ctx.fillStyle = '#7cce2b';
         ctx.fillRect(0,0,300,100);
      </script>
   </body>
</html>
```

SVG	Canvas
Vektor alapú (formákból áll)	Raszter alapú (pixelből áll)
Több grafikus elem, amelyek az oldal DOM fa részévé válnak	A viselkedéshez hasonló egyetlen elem. A canvas diagram PNG vagy JPG formátumba menthető
Szkript és CSS segítségével módosítva	Csak szkript segítségével módosítható
Jó szövegmegjelenítési képesség	Gyenge szövegmegjelenítési képességek
Job teljesítményt kisebb számú objektummal vagy	Nagyobb teljesítmény, nagyobb számú tárgy

SVG	Canvas
nagyobb felülettel, vagy mindkettővel	vagy kisebb felület, vagy mindkettő esetén
Jobb méretezhetőség. Kiváló minőségű, bármilyen felbontással nyomtatható. Pixeláció nem történik meg	Gyenge méretezhetőség. Nem alkalmas nagyobb felbontású nyomtatásra. Pixeláció előfordulhat

Magyarázza a HTML5-ben a Drag and Drop elemet?

A HTML5 Drag and Drop használja a DOM eseménymodellt, és Drag and Drop az egér eseményekből örökölt eseményeket. Egy tipikus Drag and Drop művelet akkor kezdődik, amikor a felhasználó kiválaszt egy húzható elemet, az elemet egy droppolható elemre húzza, majd elengedi a húzott elemet.

Event	Description
Drag	Minden alkalommal elindul, amikor az egeret mozgatják, miközben az objektumot húzzák.
Dragstart	Ez egy nagyon kezdeti szakasz. Akkor indul, amikor a felhasználó elkezdi az objektum húzását.
Dragenter	Akkor indul, amikor a felhasználó az egérmutatót a célelem fölé helyezi.
Dragover	Ez az esemény akkor indul el, amikor az egér egy elem felett mozog.
Dragleave	Ez az esemény akkor indul el, amikor az egér elhagy egy elemet.
Drop	Drop A drag művelet végén elindul.
Dragend	Akkor indul, amikor a felhasználó elengedi az egérgombot a drag művelet befejezéséhez.

```
<!DOCTYPE HTML>
<html>
   <head>
   <script>
        function allowDrop(ev) {
            ev.preventDefault();
        }

        function drag(ev) {
            ev.dataTransfer.setData("text", ev.target.id);
        }
```

```
        function drop(ev) {
            ev.preventDefault();
            var data = ev.dataTransfer.getData("text");
            ev.target.appendChild(document.getElementById(data));
        }
    </script>
</head>
<body>
  <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
  <img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336"
height="69">
</body>
</html>
```

Miért érdemes az IndexedDB-t használni a WebSQL helyett a HTML5-ben?

1.) A WebSQL egy olyan API, amelyet csak a Chrome és a Safari (valamint kiterjesztéssel Android és iOS) támogat. Aszinkron, tranzakciós felületet biztosít az SQLite számára. 2010 óta elavult .

Előnyök

- Támogatott nagyobb mobil böngészőkben (Android Browser, Mobile Safari, Opera Mobile), valamint számos asztali böngészőben (Chrome, Safari, Opera).
- Jó teljesítmény általában aszinkron API-ként. Az adatbázis-interakció nem zárja le a felhasználói felületet. (A WebWorkers számára is elérhető a szinkron API.)
- Jó keresési teljesítmény, mivel az adatok keresési kulcsok alapján indexelhetők.
- Robusztus, mivel támogatja a tranzakciós adatbázis modellt.
- Az adatok integritásának könnyebb fenntartása a merev adatszerkezet miatt.

Hátrányok

- Elavult. Nem fog támogatni az IE vagy a Firefox, és valószínűleg egy szakaszban fokozatosan megszűnik a többi böngészőből.
- Meredek tanulási görbe, amely megköveteli a relációs adatbázisok és az SQL ismeretét.
- Az objektum-reláció impedancia eltérése szenved.
- Csökkenti az agilitást, mivel az adatbázis-sémát előre meg kell határozni, a táblában szereplő összes rekordnak meg kell egyeznie a struktúrával.

2.) Az IndexedDB mind a LocalStorage, mind a WebSQL utódja, amelyet úgy terveztek, hogy lecserélje őket „egy igazi” böngésző adatbázisként. Aszinkron API-t tesz közzé, amely állítólag elkerüli a DOM blokkolását, de mint alább látni fogjuk, nem feltétlenül felel meg a hype-nak. A böngésző támogatása rendkívül foltos, csak a Chrome és a Firefox rendelkezik teljesen használható megvalósításokkal.

Előnyök

- Jó teljesítmény általában aszinkron API-ként. Az adatbázis-interakció nem zárja le a felhasználói felületet. (A WebWorkers számára is elérhető szinkron API.)
- Jó keresési teljesítmény, mivel az adatok keresési kulcsok alapján indexelhetők.
- Támogatja a verziószámot.

- Robusztus, mivel támogatja a tranzakciós adatbázis modellt.
- Meglehetősen könnyű tanulási görbe, egy egyszerű adatmodellnek köszönhetően.
- Tisztességes böngészőtámogatás: Chrome, Firefox, mobile FF, IE10.

Hátrányok

- Nagyon összetett API, amely nagy mennyiségű beágyazott callback hívást eredményez.

Magyarázza el az Application Cache-t HTML5-ben. VAGY Mi a manifest fájl a HTML-ben?

A HTML5 olyan alkalmazás-gyorsítótár-mechanizmust (application-cacheing) biztosít, amely lehetővé teszi a webalapú alkalmazások offline futtatását. A fejlesztők az Alkalmazás-gyorsítótár (AppCache) felület segítségével meghatározhatják azokat az erőforrásokat, amelyeket a böngészőnek gyorsítótára kell tennie és elérhetővé kell tenniük az offline felhasználók számára. A gyorsítótárazott alkalmazások akkor is betöltődnek és megfelelően működnek, ha a felhasználók offline állapotban kattintanak a Frissítés gombra.

Az alkalmazás gyorsítótárának használata a következő előnyökkel jár az alkalmazás számára:

- Offline böngészés: a felhasználók offline állapotban is navigálhatnak egy webhelyen.
- Sebesség: a gyorsítótárazott erőforrások lokálisak, ezért gyorsabban töltődnek be.
- Csökkentett szerverterhelés: a böngésző csak a szerverről megváltozott erőforrásokat tölti le.

```
<html manifest="example.appcache">
  ...
</html>
```

Megjegyzés: Az itt leírt alkalmazás gyorsítótárazás funkció használata ezen a ponton nagyon nem javasolt; éppen eltávolítás alatt áll a webplatformról. Használja helyette a Service Workers alkalmazást. Valójában a Firefox 44-től kezdve, amikor az AppCache-t egy oldal offline támogatására használják, a konzolon most egy figyelmeztető üzenet jelenik meg, amelyben a fejlesztőknek azt javasolják, hogy inkább a Service worker-t használják (1204581-es hiba).

Magyarázza a Microdata-t HTML5-ben?

A Microdata egy szabványosított módszer arra, hogy további szemantikát biztosítson a weboldalakon. A Microdata segítségével meghatározhatja saját testreszabott elemeit, és elkezdheti az egyéni tulajdonságok beágyazását a weboldalain. Magas szinten a Microdata név-érték párok csoportjából áll.

A csoportokat elemeknek nevezzük, és minden név-érték pár tulajdonság. Az elemeket és tulajdonságokat szabályos elemek jelentik. A keresőmotoroknak nagy előnye származik a strukturált adatokhoz való közvetlen hozzáférésből, mivel ez lehetővé teszi a keresőmotorok számára, hogy

megértsék a weboldalakon található információkat, és relevánsabb eredményeket nyújtsanak a felhasználók számára.

Magas szinten a Microdata név-érték párok csoportjából áll

- itemscope: - Elem létrehozása
- itemprop: - Tulajdonság hozzáadása egy elemhez

```
<div itemscope>
    <p>My name is <span itemprop="name">Elizabeth</span>.</p>
</div>

<div itemscope>
    <p>My name is <span itemprop="name">Daniel</span>.</p>
</div>
```

Felsorolja a HTML5-ben elérhető API-t?

1.) High Resolution Time API

A High Resolution Time API az aktuális időt milliszekundum alatti felbontásban adja meg, olyannak, amelyre nem vonatkozik a rendszer óra torzulása vagy beállítása.

Csak egy metódust tár fel, amely az window.performance objektumhoz tartozik, amelyet now () –nak hívnak. Visszaad egy DOMHighResTimeStamp-ot, amely milliszekundumban ábrázolja az aktuális időt. A timestamp nagyon pontos, ezredmásodperc pontossággal, lehetővé téve a kódunk teljesítményének pontos tesztelését.

```
var time = performance.now();
```

2.) Felhasználói időzítés API (User Timing API)

Ez lehetővé teszi számunkra a JavaScript-kód egy szakaszának teljesítményének pontos mérését és jelentését. Két fő fogalommal foglalkozik: jelölés és mérés. Az előbbi egy pillanatot (időbélyeg) jelent, míg az utóbbi a két jel között eltelt időt.

```
performance.mark("startFoo");
// A time consuming function
foo();
performance.mark("endFoo");

performance.measure("durationFoo", "startFoo", "endFoo");
```

3.) Network Information API

Ez az API az window.navigator objektum kapcsolat tulajdonságához tartozik. Két csak olvasható tulajdonságot tár fel: sávszélességet és mérést. Az előbbi az aktuális sávszélesség becslését jelentő szám, míg az utóbbi egy logikai érték, amelynek értéke akkor igaz, ha a felhasználó kapcsolatára korlátozások és sávszélesség-használat vonatkozik, és egyébként hamis.

Sl.No	API	Description
01.	navigator.connection.type	Hálózattípus
02.	navigator.connection.downlink	Hatékony sávszélesség-becslés (downlink)
03.	navigator.connection.rtt	Hatékony oda-vissza becsült idő (rtt)
04.	navigator.connection.downlinkMax	A letöltési sebesség felső korlátja (downlinkMax)
05.	navigator.connection.effectiveType	Hatékony csatlakozási típus
06.	navigator.connection.saveData	Igaz, ha a felhasználó csökkentett adatfelhasználási módot kért a user agent-ről (saveData)

4.) Vibration API

Csak egy metódust tesz ki, a vibrate (), amely az window.navigator objektumhoz tartozik. Ez a módszer elfogad egy paramétert, amely milliszekundumban határozza meg a rezgés időtartamát. A paraméter lehet egész szám vagy egész tömb. A második esetben váltakozó rezgési időként és szünetként értelmezik.

```
// Vibrate once for 2 seconds
navigator.vibrate(2000);
```

5.) Battery Status API

Az Battery Status API négy tulajdonságot (charging, chargingTime, dischargingTime, and level) és négy eseményt tár fel. A tulajdonságok meghatározzák, hogy az akkumulátor töltődik-e, az akkumulátor teljes feltöltődéséig hátralévő másodpercek, az akkumulátor teljes lemerüléséig hátralévő másodpercek és az akkumulátor jelenlegi szintje. Ezek a tulajdonságok a window.navigator objektum akkumulátor tulajdonságához tartoznak.

```
// Retrieves the percentage of the current level of the device's battery
var percentageLevel = navigator.battery.level * 100;
```

6.) Page Visibility API

A Page Visibility API lehetővé teszi számunkra, hogy meghatározzuk az oldal aktuális láthatósági állapotát. A Page Visibility API különösen hasznos az erőforrások megtakarításához és a teljesítmény javításához azáltal, hogy lehetővé teszi az oldal számára, hogy elkerülje a felesleges feladatokat, ha a dokumentum nem látható.

```
//document.hidden retuns true if page is not visible.
console.log('Page Visibility: '+document.hidden);
```

7.) Fullscreen API

A Fullscreen API lehetővé teszi a teljes képernyős megjelenítés kérését a felhasználótól, és ha szükséges, kiléphet ebből az üzemmódból. Ez az API két módszert tár fel, a requestFullscreen () és az exitFullscreen () lehetőséget, lehetővé téve számunkra, hogy kérjünk egy elemet teljes képernyőssé és a teljes képernyős kilépéshez.

```
document.addEventListener("keypress", function(e) {
    if (e.keyCode === 13) { // Enter Key
        toggleFullScreen();
    }
}, false);

function toggleFullScreen() {
    if (!document.fullscreenElement) {
        document.documentElement.requestFullscreen();
    } else {
        if (document.exitFullscreen) {
        document.exitFullscreen();
        }
    }
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 API</title>
  </head>
  <body>
    <p>System Time: <span id="time"></span></p>
  </body>
  <script type="text/javascript">
        // -----------
        //  Time API
        // -----------
           var time = performance.now();
           document.getElementById('time').innerHTML = time + ' ms';

        // -------------------------
        //  Network Information API
        // -------------------------

            console.log('Network Type: ' + navigator.connection.type);

            console.log('Effective bandwidth estimate ( downlink ): ' +
navigator.connection.downlink + 'Mb/s');

            console.log('Effective round-trip time estimate ( rtt ): ' +
navigator.connection.rtt + 'ms');

            console.log('Upper bound on the downlink speed of the first network hop (
downlinkMax ): ' + navigator.connection.downlinkMax + 'Mb/s');

            console.log('Effective connection type: ' + navigator.connection.effectiveType);

            console.log('True if the user has requested a reduced data usage mode from the
user agent ( saveData ): ' + navigator.connection.saveData);

        // ---------------------
        //  Page Visibility API
        // ---------------------

        /**
           The Page Visibility API is especially useful for saving resources and improving
performance
           by letting a page avoid performing unnecessary tasks when the document isn't
visible.
        **/
```

```
           console.log('Page Visibility: '+document.hidden); //document.hidden retuns true if
page is not visible.

        // ------------------
        //  Fullscreen API
        // ------------------
            // Example
            document.addEventListener("keypress", function(e) {
                if (e.keyCode === 13) { // Enter Key
                    toggleFullScreen();
                }
            }, false);

            function toggleFullScreen() {
                if (!document.fullscreenElement) {
                    document.documentElement.requestFullscreen();
                } else {
                    if (document.exitFullscreen) {
                    document.exitFullscreen();
                    }
                }
            }

        // ---------------
        //  Vibration API
        // ---------------
        navigator.vibrate(2000); // Vibrate once for 2 seconds

        // --------------------
        //  Battery Status API
        // --------------------
            var percentageLevel = navigator.battery.level * 100; // Retrieves the percentage
of the current level of the device's battery
            console.log("Battery Percentage: " +percentageLevel);
  </script>
</html>
```

Milyen különféle új űrlapelemeket kínál a HTML5?

Sl.No	Element	Description
01.	color	Natív színválasztót ad a felhasználónak a szín kiválasztására.
02.	date	Dátumválasztót kínál.
03.	datetime	A dátum és az idő kiválasztására szolgáló elem.
04.	datetime-local	A dátum és az idő kiválasztására szolgáló elem, a helyi beállítások támogatásával.
05.	email	Egy mező az e-mail cím (ek) megadásához.
06.	month	Válasszon egy teljes hónapot.
07.	number	Szám kiválasztása.
08.	range	Csúszkát kínál egy bizonyos érték / pozíció beállításához.

Sl.No	Element	Description
09.	search	Mező a keresési lekérdezésekhez.
10.	tel	Telefonszám kiválasztása.
11.	time	Adjon meg egy bizonyos időt.
12.	url	URL megadása.
13.	week	Egy adott hét kiválasztása.

```
<input type="color" value="#b97a57">
<input type="date" value="2020-06-08">
<input type="datetime" value="2020-06-09T20:35:34.32">
<input type="datetime-local" value="2020-06-09T22:41">
<input type="email" value="robert@robertnyman.com">
<input type="month" value="2020-06">
<input type="number" value="4">
<input type="range" value="15">
<!-- Note: If not set, default attribute values are min="0", max="100", step="1". -->
<input type="search" value="[Any search text]">
<input type="tel" value="[Any numeric value]">
<!-- Note: Most web browsers seem to let through any value at this time. -->
<input type="time" value="22:38">
<input type="url" value="https://www.google.com/">
<!-- Note: requires a protocol like http://, ftp:// etc in the beginning. -->

<input type="week" value="2020-W24">
```

Melyek azok a HTML-címkék, amelyek elavultak a HTML5-ben?

Elavult címkék

A következő elemek már nem érhetők el a HTML5-ben, és funkciójukat a CSS jobban kezeli.

Sl.No	Tags (Elements)	Description
01.	<acronym>	Defines an acronym
02.	<applet>	Defines an applet

Sl.No	Tags (Elements)	Description
03.	<basefont>	Defines an base font for the page.
04.	<big>	Defines big text
05.		Defines centered text
06.	<dir>	Defines a directory list
07.	<font>	Defines text font, size, and color
08.	<frame>	Defines a frame
08.	<frameset>	Defines a set of frames
10.	<isindex>	Defines a single-line input field
11.	<noframes>	Defines a noframe section
12.	<s>	Defines strikethrough text
13.	<strike>	Defines strikethrough text
14.	<tt>	Defines teletype text
15.	<u>	Defines underlined text

Deprecated Attributes

Removed Attributes	From the Elements
rev	link, a
charset	link and a
shape	a
coords	a
longdesc	img and iframe.
target	link
nohref	area
profile	head
version	html
name	img
scheme	meta
archive	object
classid	object
codebase	object
codetype	object
declare	object

Removed Attributes	From the Elements
standby	object
valuetype	param
type	param
axis	td and t
abbr	td and t
scope	td
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
alink	body
link	body
vlink	body
text	body
background	body
bgcolor	table, tr, td, th and body.
border	table and object.
cellpadding	table
cellspacing	table
char	col, colgroup, tbody, td, tfoot, th, thead and tr.
charoff	col, colgroup, tbody, td, tfoot, th, thead and tr.
clear	br
compact	dl, menu, ol and ul.
frame	table
compact	dl, menu, ol and ul.
frame	table
frameborder	iframe
hspace	img and object.
vspace	img and object.
marginheight	iframe
marginwidth	iframe
noshade	hr
nowrap	td and th
rules	table
scrolling	iframe

Removed Attributes	From the Elements
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead and tr
width	hr, table, td, th, col, colgroup and pre.

Hogyan használható a Modernizr a HTML5-ben?

A Modernizr egy JavaScript könyvtár, amely érzékeli, hogy mely HTML5 és CSS3 szolgáltatásokat támogatja a látogató böngészője. A szolgáltatás támogatásának felderítésében lehetővé teszi a fejlesztők számára, hogy teszteljék az új technológiákat, majd tartalékokat biztosítsanak azoknak a böngészőknek, amelyek nem támogatják őket. Ezt hívják szolgáltatásfelismerésnek

A Modernizr használata CSS-sel

Alapértelmezés szerint a Modernizr az összes teszt osztályát a gyökérelemre állítja be (<html> webhelyekhez). Ez azt jelenti, hogy hozzá kell adni az osztályt minden egyes szolgáltatáshoz, ha támogatott, és hozzá kell adni egy nem-előtaggal, ha nem (pl. .Feature vagy .no-feature).

```
.no-cssgradients .header {
  background: url("images/glossybutton.png");
}

.cssgradients .header {
  background-image: linear-gradient(cornflowerblue, rebeccapurple);
}
```

A Modernizr használata JavaScript-szel

A Modernizr nyomon követi az összes szolgáltatás-észlelés eredményét a Modernizr objektumon keresztül.

```
if (Modernizr.canvas) {
    alert("This browser supports HTML5 canvas!");
} else {
    alert("no canvas :(");
}
```

Mi a WebSocket API használata?

A WebSocket API egy fejlett technológia, amely lehetővé teszi kétirányú interaktív kommunikációs munkamenet megnyitását a felhasználó böngészője és egy szerver között. Ezzel az API-val üzeneteket küldhet egy kiszolgálónak, és eseményvezérelt válaszokat fogadhat anélkül, hogy válaszra lenne szüksége a szervernek.

Interfaces

Sl.No	API	Description
01.	WebSocket	Az elsődleges felület a WebSocket kiszolgálóhoz való csatlakozáshoz, majd a kapcsolat adatainak küldéséhez és fogadásához.
02.	CloseEvent	A WebSocket objektum által a kapcsolat lezárásakor küldött esemény.
03.	MessageEvent	A WebSocket objektum által küldött esemény, amikor üzenet érkezik a szerverről.

```
// Create WebSocket connection.
const socket = new WebSocket('ws://localhost:8080/');

// Connection opened
socket.addEventListener('open', function(event) {
    socket.send('Hello Server!');
});

// Listen for messages
socket.addEventListener('message', function(event) {
    console.log('Message from server ', event.data);
});
```

Mit jelent az enctype = 'multipart / form-data'?

Az enctype attribútum határozza meg, hogy az űrlap-adatokat hogyan kell kódolni, amikor elküldik a szervernek.

Example: 01

```
<form action="fileupload.php" method="post" enctype="multipart/form-data">
    <p>Please select the file you would like to upload.</p>
    <input type="file" name="upload">
    <br>
    <input type="submit" value="Upload File">
</form>
```

Example: 02

```
<form action="/urlencoded?token=A87412B" method="POST" enctype="application/x-www-form-urlencoded">
    <input type="text" name="username" value=""/>
    <input type="text" name="password" value=""/>
    <input type="submit" value="Submit" />
</form>
```

Example: 03

```
<form action="action.do" method="get" enctype="text/plain">
    Name: <input type="text" name="name" />
    Phone: <input type="number" name="phone" />
    <input type="submit" value="Submit" />
</form>
```

Sl.No	Value	Description
01.	application/x-www-	Alapértelmezett. Az összes karakter kódolása elküldés előtt (a

Sl.No	Value	Description
	form-urlencoded	szóközök "+" szimbólummá, a speciális karakterek pedig ASCII HEX értékekké alakulnak át)
02.	multipart/form-data	Nincs karakter kódolva. Erre az értékre akkor van szükség, ha olyan űrlapokat használ, amelyek fájlfeltöltési vezérléssel rendelkeznek
03.	text/plain	A szóközök "+" szimbólummá alakulnak át, de nincsenek speciális karakterek kódolva

Mi a különbség a Select és a Datalist között?

A select elemhez a felhasználónak ki kell választania az Ön által megadott lehetőségek egyikét. A datalista elemhez javasoljuk, hogy a felhasználó válassza ki az Ön által megadott lehetőségek egyikét, de valójában bármit beírhat a bemenetbe.

Select-Option

```
<select name="browser">
  <option value="firefox">Firefox</option>
  <option value="ie">IE</option>
  <option value="chrome">Chrome</option>
  <option value="opera">Opera</option>
  <option value="safari">Safari</option>
</select>
```

Datalist-Option

```
<input type="text" list="browsers">
<datalist id="browsers">
  <option value="Firefox">
  <option value="IE">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```