https://github.com/FAQGURU/FAQGURU/blob/master/topics/en/html5.md

Magyarázza el a meta tag-eket HTML-ben

A meta tag-ek mindig a HTML-oldal head tag-be kerülnek

A meta tag-ek etmindig név / érték párként továbbítják

A meta tag-ek nem jelennek meg az oldalon, de a böngészőnek szólnak

A meta tag-ek információkat tartalmazhatnak a karakterkódolásról, a leírásról, a dokumentum címéről stb.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="description" content="I am a web page with description">
  <title>Home Page</title>
</head>
<body>

</body>
</html>
```

Hogyan lehet jobban indexelni a keresőmotorok által?

Jobban indexelhető, ha a következő két állítást elhelyezi a dokumentumok <HEAD> részében

```
<META NAME="keywords" CONTENT="keyword keyword keyword keyword">
<META NAME="description" CONTENT="description of your site">
```

Mindkettő legfeljebb 1022 karaktert tartalmazhat. Ha egy kulcsszót több mint 7 alkalommal használnak, a kulcsszó címkét teljesen figyelmen kívül hagyják

Mi az a karakterkódolás?

A HTML-oldal helyes megjelenítéséhez a webböngészőnek tudnia kell, hogy melyik karakterkészletet (karakterkódolást) használja. Ezt a címke határozza meg:

HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
```

HTML5:

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

Melyek voltak a HTML5 specifikáció legfontosabb céljai és motivációi?

A HTML5-et a HTML 4, az XHTML és a HTML DOM 2. szintjének helyettesítésére tervezték. A HTML5 specifikáció mögött álló fő célok és motivációk a következők voltak:

- Gazdag tartalom (grafika, filmek stb.) továbbítása anélkül, hogy további pluginokra lenne szükség, például Flashre.
- Jobb szemantikai támogatást nyújt a weboldal struktúrájához új strukturális elemcímkék révén.
- Szigorúbb elemzési szabvány biztosítása a hibakezelés egyszerűsítése, a böngészők közötti következetesebb viselkedés és a régebbi szabványokhoz írt dokumentumokkal való kompatibilitás egyszerűsítése érdekében.
- Jobb platformok közötti támogatást nyújt, akár PC-n, táblagépen vagy okostelefonon fut.

Magyarázza el a különbséget a blokkelemek és az inline elemek között

- A blokkelemek azok, amelyek a weblapon elérhető teljes szélességet kihasználják, és gyakorlatilag megakadályozzák, hogy a többi elem bal vagy jobb oldalon legyen mellette.
- Az inline elemek azok, akik csak annyi szélességet vesznek fel, amennyire az elem tartalmának megjelenítéséhez szükség van, ezáltal lehetővé téve, hogy más elemek összhangban legyenek az inline elemmel.

Common block elements:

- Paragraphs (`<p>`)
- Headers (`<h1>` through `<h6>`)
- Divisions (`<div>`)
- Lists and list items (`<ol>`, `<ul>`, and `<li>`)
- Forms (`<form>`)

Common inline elements:

- Spans (`<span>`)
- Images (`<img>`)
- Anchors (`<a>`)

Mi az opcionális tag?

A HTML-ben egyes elemek opcionális tag-ekkel rendelkeznek. Valójában egyes elemek nyitó és záró tag-jei teljesen eltávolíthatók egy HTML-dokumentumból, még akkor is, ha maguk az elemek szükségesek.

Három kötelező HTML elem, amelyek kezdő és záró címkéi opcionálisak, a html, a head és a body elemek.

Mikor célszerű használni a small elemet?

A HTML <small> elem a szöveg betűméretét egy méretrel kisebbé teszi a böngésző minimális betűméretéig.

```
<img src="image.jpg" alt="London by night">
<small>The copyright of this image is owned by Aurelio De Rosa</small>
```

Mi a különbség a <section> és a között?

A <section> azt jelenti, hogy a belső tartalom csoportosítva van (azaz egyetlen témához kapcsolódik), és bejegyzésként kell megjelennie.

A viszont semmilyen jelentést nem közvetít, az osztályán, a lang és a title attribútumain kívül.

Mik azok a Web Workers?

- A Web Workers segít javascript kód futtatásában a háttérben az alkalmazás blokkolása nélkül.
- A Web Workers egy elszigetelt (új) szálban fut a javascript kódunk végrehajtásához.
- A Web Workers t általában nagy feladatokra használják.
- A Web Workers-nek külön fájlra van szükségük a javascript kódunkhoz.
- A Web Workers fájlok aszinkron módon kerülnek letöltésre.
- A Web Workers minden legújabb böngésző támogatja.

Example:

Our client side js file below:

```
var myWebWorker = new Worker("task.js"); // Creating a worker

// Listen to task.js worker messages
worker.addEventListener("message", function(event) {
  console.log("Worker said: ", event.data);
}, false);

worker.postMessage("From web worker file"); // Will start the worker
```

task.js (worker file) file below:

```
// Listen to client js file post messages
self.addEventListener("message", function(event) {
  self.postMessage(event.data); // Send processed data to listening client js file.
}, false);
```

Mit csinál a DOCTYPE?

A DOCTYPE a „dokumentumtípus - document type” rövidítése. Ez egy olyan deklaráció, amelyet HTML-ben használnak a szabványos és a furcsa mód (quirks) megkülönböztetésére. Jelenléte arra utasítja a böngészőt, hogy a weboldalt normál módban jelenítse meg.

csak adja hozzá az <! DOCTYPE html> -t az oldal elejéhez.

Hogyan lehet egy olyan oldalt kiszolgálni, amelynek tartalma több nyelven jelenik meg?

Amikor HTTP-kérést küldenek egy szerverre, a felhasználói kliens általában információkat küld a nyelvi beállításokról, például az Accept-Language fejlécben. A kiszolgáló ezután felhasználhatja ezeket az információkat a dokumentum megfelelő nyelvű változatának visszaadására, ha rendelkezésre áll ilyen alternatíva. A visszaküldött HTML dokumentumnak deklarálnia kell a lang attribútumot is a <html> címkében, például <html lang = "en"> ... </html>.

A backend-ben a HTML markup i18n placeholder-eket és az adott nyelv YML vagy JSON formátumban tárolt tartalmát fogja tartalmazni. Ezután a szerver dinamikusan generálja a HTML oldalt az adott nyelv tartalmával, általában egy backend framework segítségével.

Mi a célja a cache busting-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 (header) 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 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 nevezik el, mint a régi fájlt.

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

src = "js / script.js" => src = "js / script.js? v = 2"

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

Tartalmazhat egy weboldal több <header> elemet? Mi a helyzet a <footer> elemekkel?

Igen mindkettőnek. Tehát nemcsak a <body> oldal tartalmazhat fejlécet és láblécet, hanem minden <article> és <section> elem is.

Mik a defer és async attribútumok egy <script> tag-en?

Ha egyik attribútum sincs, a szkript szinkron módon kerül letöltésre és végrehajtásra, és mindaddig leállítja a dokumentum elemzését, amíg befejezi a végrehajtást (alapértelmezett viselkedés). A parancsfájlok sorrendben kerülnek letöltésre és végrehajtásra.

A defer attribútum akkor tölti le a parancsfájlt, amikor a dokumentum még elemzés alatt áll, de a végrehajtás előtt megvárja, amíg a dokumentum befejezi az elemzést, ami megegyezik a DOMContentLoaded eseményfigyelő belsejében végrehajtással. a defer szkriptek sorrendben fognak futni.

Az async attribútum letölti a parancsfájlt a dokumentum elemzése közben, de az elemzőt szünetelteti a parancsfájl végrehajtása előtt, mielőtt az az elemzés teljesen befejeződött. az aszinkron szkriptek nem feltétlenül fognak sorrendben futni.

Megjegyzés: mindkét attribútumot csak akkor szabad használni, ha a szkript src attribútummal rendelkezik (azaz nem inline szkript).

```
<script src="myscript.js"></script>
<script src="myscript.js" defer></script>
<script src="myscript.js" async></script>
```

HTML specifikáció és a böngésző megvalósítása közötti különbségeket.

A HTML specifikációk, például a HTML5, olyan szabályokat határoznak meg, amelyeket a dokumentumnak be kell tartania ahhoz, hogy az adott specifikáció szerint érvényes legyen. Ezenkívül a specifikáció utasításokat tartalmaz arra vonatkozóan, hogyan kell a böngészőnek értelmeznie és megjelenítenie az ilyen dokumentumot.

A böngésző „támogatja” a specifikációt, ha érvényes dokumentumokat kezel a specifikáció szabályainak megfelelően. Eddig egyetlen böngésző sem támogatja a HTML5 specifikáció összes aspektusát (bár az összes fő böngésző támogatja a legtöbbet), és ennek eredményeként a fejlesztőnek meg kell erősítenie, hogy az általuk használt szempont támogatott lesz-e. az összes olyan böngésző által, amelyen remélik megjeleníteni tartalmukat. Éppen ezért a böngészők közötti támogatás a fejlettebb specifikációk ellenére továbbra is fejfájást jelent a fejlesztők számára.

- A HTML5 meghatároz néhány szabályt, amelyet be kell tartani egy érvénytelen HTML5 dokumentum esetében (vagyis olyan, amely szintaktikai hibákat tartalmaz)
- Az érvénytelen dokumentumok azonban bármit tartalmazhatnak, ezért lehetetlen, hogy a specifikáció minden lehetőséget átfogóan kezeljen.
- Így a hibásan formázott dokumentumok kezelésével kapcsolatos számos döntés a böngészőre marad.

Milyen különbségek vannak az XHTML-ben a HTML-hez képest?

Néhány fő különbség:

- Az XHTML elemnek XHTML <DOCTYPE> kell lennie
- Az attribútumértékeket idézőjelek közé kell tenni
- Az attribútumminimalizálás tilos (pl. A checked="checked -t kell használnia a cheched helyett)
- Az elemeket mindig megfelelően be kell ágyazni
- Az elemeknek mindig zárva kell lenniük
- A speciális karaktereket el kell kerülni

Mi új a HTML 5-ben?

A HTML 5 sok új funkcióval egészíti ki a HTML specifikációt

1.) Új Doctype

Régi doctype:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Új doctype:

```
<!DOCTYPE html>
```

2.) Új struktúra

- `<section>` - to define sections of pages
- `<header>` - defines the header of a page
- `<footer>` - defines the footer of a page
- `<nav>` - defines the navigation on a page
- `<article>` - defines the article or primary content on a page
- `<aside>` - defines extra content like a sidebar on a page
- `<figure>` - defines images that annotate an article

New Inline Elements

These inline elements define some basic concepts and keep them semantically marked up, mostly to do with time:

- `<mark>` - to indicate content that is marked in some fashion
- `<time>` - to indicate content that is a time or date
- `<meter>` - to indicate content that is a fraction of a known range - such as disk usage
- `<progress>` - to indicate the progress of a task towards completion

New Form Types

- `<input type="datetime">`
- `<input type="datetime-local">`

- `<input type="date">`
- `<input type="month">`
- `<input type="week">`
- `<input type="time">`
- `<input type="number">`
- `<input type="range">`
- `<input type="email">`
- `<input type="url">`

New Elements

There are a few exciting new elements in HTML 5:

- `<canvas>` - an element to give you a drawing space in JavaScript on your Web pages. It can let you add images or graphs to tool tips or just create dynamic graphs on your Web pages, built on the fly.
- `<video>` - add video to your Web pages with this simple tag.
- `<audio>` - add sound to your Web pages with this simple tag.

No More Types for Scripts and Links

You possibly still add the `type` attribute to your `link` and `script` tags.

```
<link rel="stylesheet" href="path/to/stylesheet.css" type="text/css" />
<script type="text/javascript" src="path/to/script.js"></script>
```

This is no longer necessary. It's implied that both of these tags refer to stylesheets and scripts, respectively. As such, we can remove the `type` attribute all together.

```
<link rel="stylesheet" href="path/to/stylesheet.css" />
<script src="path/to/script.js"></script>
```

Make your content editable

The new browsers have a nifty new attribute that can be applied to elements, called `contenteditable`. As the name implies, this allows the user to edit any of the text contained within the element, including its children. There are a variety of uses for something like this, including an app as simple as a to-do list, which also takes advantage of local storage.

```
<h2> To-Do List </h2>
<ul contenteditable="true">
  <li> Break mechanical cab driver. </li>
  <li> Drive to abandoned factory
  <li> Watch video of self </li>
</ul>
```

Attributes

- `require` to mention the form field is required
- `autofocus` puts the cursor on the input field

Magyarázza el a különbséget a cookie, a session és a local storage között

Cookie-k

- Korlátozott tárhely 4096 bájt / ~ 4 kb
- Csak stringként tárolhatja az adatokat
- A tárolt adatokat minden HTTP-kérés esetén visszaküldik a szerverre, például HTML, CSS, Képek stb.

- Domainenként csak 20 cookie tárolható
- Összesen 300 cookie engedélyezett egy webhelyen
- A Csak HTTP jelző beállítása megakadályozza a sütik javascripten keresztüli elérését
- Beállíthatja a lejárati időtartamot az automatikus törléshez (beállítható szerverről vagy kliensről)

```
// Set with expiration & path
document.cookie = "name=Gokul; expires=Thu, 18 Dec 2016 12:00:00 UTC; path=/;";

// Get
document.cookie;

// Delete by setting empty value and same path
document.cookie = "name=; expires=Thu, 18 Dec 2016 12:00:00 UTC; path=/;";
```

Session storage

- A tárhely 5 mb / ~ 5120 kb
- A tárhely egy kicsit változhat a böngésző függvényében
- Csak stringként tárolhatja az adatokat
- Az adatok ablakonként vagy fülenként állnak rendelkezésre
- Az ablak vagy a lap bezárása után a tárolt adatok törlődnek
- Az adatok csak azonos eredetről lesznek elérhetőek

```
// Set
sessionStorage.setItem("name", "gokul");

// Get
sessionStorage.getItem("name"); // gokul

// Delete
sessionStorage.removeItem("name");

// Delete All
sessionStorage.clear();
```

Local storage

- A tárhely 5 mb / ~ 5120 kb
- A tárhely egy kicsit változhat a böngésző függvényében
- Csak stringként tárolhatja az adatokat
- Az adatok csak azonos eredetről lesznek elérhetőek
- Az adatok állandóak (a külön törlésig)
- Az API hasonló a munkamenet-tároláshoz

```
// Set
localStorage.setItem("name", "gokul");

// Get
```

```
localStorage.getItem("name"); // gokul

// Delete
localStorage.removeItem("name");

// Delete All
localStorage.clear();
```

Magyarázza el a almost standard, full standard és quirks (furcsa) mode-t

Három módot használnak az elrendezőmotorok a webböngészőkben: almost standard, full standard és quirks mode.

- quirks mode-ban 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.
- full standard módban a viselkedés (remélhetőleg) a HTML és CSS specifikációk által leírt viselkedés.
- almost standard módban csak nagyon kevés furcsaság valósul meg.
- HTML-dokumentumok esetében a böngészők a dokumentum elején egy DOCTYPE-t használnak annak eldöntésére, hogy quirks vagy standard módban kezeljék-e.

Mi a progresszív renderelés?

A progresszív renderelés annak a technikának a neve, amelyet a weboldal teljesítményének javítására (különösen az észlelt betöltési idő javítására) használnak a tartalom mielőbbi megjelenítéséhez.

Korábban sokkal szélesebb körben elterjedt a szélessávú internet előtti napokban, de még mindig használják a modern fejlesztésekben, mivel a mobil adatkapcsolatok egyre népszerűbbek (és megbízhatatlanabbak)!

Példák ilyen technikákra:

- Lazy loading of images - Az oldalon lévő képek nem töltődnek be egyszerre. A JavaScript használatával egy kép betöltésre kerül, amikor a felhasználó görget az oldal azon részéhez, amely a képet megjeleníti.
- A látható tartalom rangsorolása - Csak a minimális CSS / tartalom / szkriptek szerepeltetése szükséges ahhoz az oldalmennyiséghez, amelyet a felhasználói böngészőben előbb megjelenítenének a lehető leggyorsabb megjelenítéshez, majd használhat deferred (halasztott) szkripteket vagy figyelje DOMContentLoaded / load esemény betöltését más erőforrásokba és tartalmakba.

Miért használna srcset attribútumot egy image tag-ben?

Akkor használja az srcset attribútumot, ha az eszköz megjelenítési szélességétől függően különböző képeket szeretne kiszolgálni a felhasználók számára - a retina kijelzővel rendelkező készülékek jobb

minőségű képei javítják a felhasználói élményt, miközben alacsonyabb felbontású képeket nyújtanak alacsony kategóriájú eszközöknek, így növelve a teljesítményt és csökkentve az adatokat pazarlás (mert nagyobb kép kiszolgálásakor nem lesz látható különbség).

Például: <img srcset = "small.jpg 500w, medium.jpg 1000w, large.jpg 2000w" src = "..." alt = ""> megadja a böngészőnek, hogy jelenítse meg a kis, közepes vagy nagy .jpg grafikát a klienstől függően. Az első érték a kép neve, a második pedig a kép szélessége pixelben.

Milyen dolgokra kell vigyáznia, amikor többnyelvű webhelyeket tervez vagy fejleszt?

- Használja a lang attribútumot a HTML-ben.
- A felhasználók anyanyelvükre terelése - Lehetővé teszi a felhasználó számára, hogy gond nélkül könnyedén megváltoztassa országát / nyelvét.
- A képeken szereplő szöveg nem skálázható megközelítés - A szöveg képbe történő elhelyezése továbbra is népszerű módja annak, hogy szép megjelenésű, rendszeren kívüli betűtípusokat jelenítsen meg bármely számítógépen. A képszöveg lefordításához azonban minden szövegsornak külön nyelvet kell létrehoznia az egyes nyelvekhez.
- Korlátozó szavak / mondathossz - Egyes tartalmak hosszabbak lehetnek, ha más nyelven írják őket. Legyen óvatos az elrendezéssel a tervezés során. A karakterek száma olyan dolgoknál játszik nagy szerepet, mint a címsorok, a címkék és a gombok. Kevésbé számítanak a szabad szövegeknek, például a törzsszövegnek vagy a megjegyzéseknek.
- Ügyeljen arra, hogy miként érzékelik a színeket - a színeket különböző módon érzékelik a nyelvek és kultúrák között. A kialakításnak megfelelő színt kell használnia.
- Dátumok és pénznemek formázása - A naptári dátumokat néha különböző módon mutatják be.
- Ne összefűzze a lefordított karakterláncokat - például "A mai dátum" + datum, mert különböző szórendű nyelveken más-más lehet a tördelése. Ehelyett használjon sablont a karakterláncok helyettesítésével az egyes nyelvekhez. Nézze meg például a következő két mondatot angolul, illetve kínaiul: I will travel on {% date %} and {% date %} 我会出发. Vegye figyelembe, hogy a változó helyzete eltér a nyelv nyelvtani szabályai miatt.
- Nyelvolvasási irány - angolul balról jobbra, felülről lefelé olvasunk, hagyományos japán nyelven a szöveget fel-le, jobbról balra olvassuk.

Melyek a HTML5 építőkövei?

- Szemantika - Lehetővé teszi, hogy pontosabban leírja a tartalmát.
- 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 - Videó és audio létrehozása a nyílt 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 - Különböző bemeneti és kimeneti eszközök használatának engedélyezése.
- Stílus - hagyjuk, hogy a szerzők kifinomultabb stílusokat írjanak.

Mi az a HTML előfeldolgozó (preprocessor),?

A HTML preprocessor segít abban, hogy kevesebb kóddal gyorsabban tudjuk írni a HTML kódot.

Néhány előfeldolgozó:

- Pug
- Haml

Example: **index.jade**

```
html
  head
    title= HelloWorld
  body
    h1 Using Jade to create HelloWorld web page
```

Compiled file: **index.html**

```
<!DOCTYPE html>
<html>
  <head>
    <title>HelloWorld</title>
  <body>
    <h1>Using Jade to create HelloWorld web page</h1>
  </body>
</html>
```

Mik azok a web-komponensek?

- A webkomponensek egy webplatform-API-készlet.
- A webkomponensek lehetővé teszik számunkra, hogy egyedi újrafelhasználható modulokat vagy komponenseket hozzunk létre egy webalkalmazásban.
- A webkomponensek a böngésző részét képezik, és minden modern böngészőben működni fognak.
- A webkomponensek működéséhez nincs szükség külső könyvtárakra.

Jellemzők

- Egyedi elemek - írjuk meg a saját teljes funkcionalitású DOM elemeinket.
- Shadow DOM - az iframe, a stílus és a weboldal jelölésének legjobb tulajdonságát nyújtja.
- HTML sablonok - lehetővé teszi bizonyos jelölések tárolását a weboldalon, és később klónozhatjuk és felhasználhatjuk őket.
- HTML-importálás - lehetővé teszi egy külső HTML-dokumentum importálását.