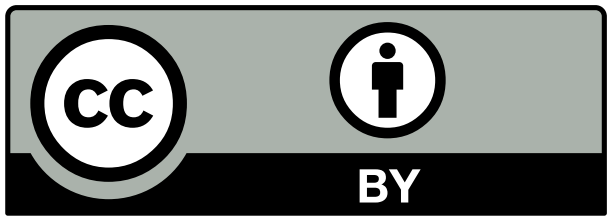
HTML

Jeszenszky Péter

Debreceni Egyetem, Informatikai Kar

2025. október 24.

Felhasználási feltételek



Jelölőnyelvek (1)

- A jelölőnyelvek szöveg annotálására szolgáló számítógépes nyelvek.
- Lehetővé teszik szövegrészekhez metaadatok megadását a szövegtől jól elkülöníthető módon.

Jelölőnyelvek (2)

- A “jelölő” (*markup*) kifejezés azokat a szintaktikai konstrukciókat jelenti, amelyek metaadatok a szövegrészekhez való megadására szolgálnak.
  - Használhatók egy szövegrész jelentésének jelzésére, vagy annak vezérlésére, hogy a feldolgozó alkalmazások hogyan jelenítsék meg azt.
- A “dokumentum” kifejezés jelölőkkel kevert szöveget jelent.

Mi a HTML? (1)

- Eredetileg “Hiperszöveg Jelölőnyelv” (*HyperText Markup Language*) volt a HTML jelentése.
- A jelenlegi HTML szabvány azonban már nem tekinti a HTML-t rövidítésnek, hanem egy feloldás nélküli önálló kifejezésként használja.

Mi a HTML? (2)

- “A HTML a Web elsődleges leíró nyelve.”
- “[...] egy szemantikai szintű leíró nyelv és a kapcsolódó szemantikai szintű alkalmazásprogramozási interfészek a Weben elérhető oldalak készítéséhez, amelyek a statikus dokumentumoktól a dinamikus alkalmazásokig terjednek.”
    - Lásd: HTML Living Standard

Mi a HTML? (3)

- A HTML egy jelölőnyelv, amelynek célja tartalom publikálása a weben.
- Statikus weboldalak és dinamikus webalkalmazások létrehozásához is használható.
- A “hiperszöveg” kifejezés dokumentumok közötti navigálható kapcsolatok, úgynevezett hiperlinkek ábrázolásának képességét jelenti.

A HTML születése

- Tim Berners-Lee (TBL) találta fel és alkotta meg a HTML-t.
- Az első nyilvános webhely: http://info.cern.ch/ (indulás: 1991. augusztus 6.)
  - Lásd: Restoring the first website
- Az első nyilvános műszaki dokumentáció a HTML-ről: https://info.cern.ch/hypertext/WWW/MarkUp/Tags.html
- A HTML korai verziói mind az SGML-en alapultak.

Főbb HTML verziók (1)

- HTML 4.01 (hatálytalanítva)
- XHTML 1.0 (hatálytalanítva)
- XHTML 1.1 (hatálytalanítva)
- HTML5 (az aktuális verzió)

Főbb HTML verziók (2)

Verzió statisztikák:

- Usage statistics and market share of HTML for websites (W3Techs)
  HTML5 is used by 97.8% of all the websites who use HTML.

HTML 4.01 (1)

HTML 4.01 Specification (W3C ajánlás, 1999. december 24.; hatálytalanítva: 2018. március 27.)

- Az utolsó SGML-alapú HTML verzió.
- Médiatípus: `text/html`

HTML 4.01 (2)

Dokumentumtípus-deklarációk:

- **Strict**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

- **Transitional**:

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

- **Frameset**:

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML

- Az XML alkalmazásként definiált HTML szigorúbb szabályokat ír elő a dokumentumok számára, így azok feldolgozása egyszerűbb.
- Különösen lényeges ez a hagyományos asztali gépekhez képest korlátozott lehetőségekkel bíró eszközöknél (például mobil eszközöknél).
- Az XHTML illetve annak modularizációja lehetővé teszi az XHTML kombinálását más XML alkalmazásokkal.
  - Például MathML és SVG beágyazás XHTML dokumentumokba – ezek a dokumentumok a továbbiakban azonban már nem XHTML dokumentumok.

XHTML 1.0 (1)

XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0 (W3C ajánlás, 2000. január 26.; hatálytalanítva: 2018. március 27.)

- A HTML 4 újrafogalmazása XML alkalmazásként.
- Médiatípus: `application/xhtml+xml`

XHTML 1.0 (2)

Dokumentumtípus-deklarációk:

- **Strict**:

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

- **Transitional**:

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

- **Frameset**:

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

XHTML 1.0 (3)

XHTML 1.0-ben írt weboldalak:

- IANA protokoll regiszterek, például:
  - Media Types
  - Hypertext Transfer Protocol (HTTP) Status Code Registry

XHTML 1.1 (1)

XHTML Modularization 1.1 – Second Edition (W3C ajánlás, 2010. július 29.; hatálytalanítva: 2018. március 27.)

- A modularizáció lehetővé teszi az XHTML résznyelveinek definiálását és az XHTML kiterjesztését.
- Megvalósítható a DTD vagy XML Schema felhasználásával is.
- Szabványos modulok egy készletét biztosítja.
  - Például: *Frames* (`frame`, `frameset`, `noframes` elemek), *Hypertext* (`a` elem), *Text* (`div`, `h1`, `p`, ... elemek), ...
- Több modul kombinálása révén úgynevezett hibrid dokumentumtípusok létrehozását teszi lehetővé.

XHTML 1.1 (2)

XHTML Basic 1.1 – Second Edition (W3C ajánlás, 2010. november 23.; hatálytalanítva: 2018. március 27.)

- Az XHTML egy olyan részhalmaza, amely számos különböző eszköz számára alkalmas (mobiltelefonok, PDA-k, elektronikus könyvolvasók, tv-készülékek, ...).
- A dokumentumtípus-definíció megvalósítása modulok segítségével az *XHTML Modularization* ajánlásban foglaltak szerint.
- Dokumentumtípus-deklaráció:

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML Basic 1.1//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
```

- Médiatípus: `application/xhtml+xml`

XHTML 2.0

XHTML 2.0 (W3C munkacsoport feljegyzés, 2010. december 16.)

- Egy új nyelv, amely nem volt kompatibilis visszafelé a korábbi HTML és XHTML nyelvekkel.
- A W3C úgy döntött, hogy nem folytatják tovább a fejlesztést, így nem lesz XHTML 2.0 ajánlás.
- A HTML5 vette át eredetileg kitűzött szerepét.
- Lásd: Frequently Asked Questions (FAQ) about the future of XHTML (W3C)

HTML5: történet (1)

Lásd: HTML Living Standard – Introduction – History

HTML5: történet (2)

- Eredetileg a WHATWG fejlesztette ki a HTML5 specifikációt.
- A W3C 2007-ben kapcsolódott be a HTML5 fejlesztésébe.
- Lásd:
  - Ian Hickson. The WHATWG Blog – W3C restarts HTML effort. 7 March 2007.
  - W3C Relaunches HTML Activity. 7 March 2007.

HTML5: történet (3)

- 2012 júliusa és 2019 júniusa között a WHATWG és a W3C is külön specifikációt fejlesztett, amelynek során eltérő fejlesztési modellt követtek.
- A W3C a HTML 5.0 specifikáció ajánlásként való kiadása után a következő verzión dolgozott (HTML 5.1).
- A WHATWG specifikációja soha nem lesz lezárt, folyamatosan fejlesztik ("élő szabvány").

HTML5: történet (4)

- 2019 júniusáig a W3C-n belül a Web Platform Munkacsoport fejlesztette a HTML nyelvhez kapcsolódó specifikációkat.
- A W3C specifikációi a WHATWG specifikációján alapultak.
  - A W3C bizonyos részeket külön dokumentumokba emelt ki.

HTML5: történet (5)

- A két szervezet 2019. május 28-án aláírt egy megállapodást arról, hogy együttműködnek a HTML és DOM specifikációk egyetlen verziójának kifejlesztésén.
  - Lásd: Jeff Jaffe. W3C and WHATWG to work together to advance the open Web platform. 28 May 2019.
- Együttműködési megállapodás:
  - A HTML-t és a DOM-ot elsősorban a WHATWG fejleszti.
  - A W3C ajánlásként szándékozik jóváhagyni és kiadni a WHATWG specifikációkat.
  - Lásd: Memorandum of Understanding Between W3C and WHATWG. May 28, 2019.
- A továbbiakban a W3C-n belül a HTML Munkacsoport felelős a HTML fejlesztéséért.

HTML5 szabvány

- **WHATWG**:
  - HTML Living Standard
  - HTML: The Living Standard – Edition for Web Developers
    - Nem tartalmazza a csupán a böngészőgyártók számára szóló információkat.
- **W3C**: a https://www.w3.org/TR/html URI jelenleg átirányít a WHATWG specifikációra.

HTML dokumentumok felépítése

- A HTML az XML-hez hasonlóan az SGML-ből származik.
- Az XML- és a HTML-dokumentumok is elemekből állnak.
- Néhány HTML konstrukció furcsának vagy elavultnak tűnhet, mint például a dokumentumtípus-deklaráció.
    - Ezek a régi lehetőségek a SGML-ből származnak, amelyen a HTML korai verziói alapultak, és elsősorban a visszafelé kompatibilitás miatt tartották meg őket.

Elemek (1)

- A HTML-dokumentumok **nyitócímkék** (***start tags***) és **zárócímkék** (***end tags***) által határolt elemekből állnak.
  - A dokumentum egy kezdőcímke és a neki megfelelő zárócímke közötti részét az elem **tartalmának** nevezzük.
    - A tartalom magában foglalhat szöveget és más elemeket is, vagyis az elemek egymásba ágyazhatók.
  - Az elemekhez megadhatók metaadatokat hordozó és **attribútumoknak** nevezett név-érték párok.
    - Az attribútumokat az elemek nyitócímkéiben kell megadni.

Elemek (2)

Példák:

- ```
  <h2>Table of contents</h2>
  ```
- ```
  <a href="https://html.spec.whatwg.org/" target="_blank">HTML
    Standard</a>
  ```
- ```
  <ul class="links">
    <li><a href="https://whatwg.org/">WHATWG</a></li>
    <li><a href="https://www.w3.org/">W3C</a></li>
    <li><a href="https://developer.mozilla.org/">MDN Web Docs</a></li>
  </ul>
  ```
  ```

A HTML-dokumentumok fák (1)

- A HTML szabvány szerint egy HTML-dokumentum egy faszerkezet, amelynek csomópontjai elemeket, szöveget vagy egyéb HTML konstrukciókat (például megjegyzéseket) ábrázolnak.
- Egy HTML-dokumentum ezen fa ábrázolásmódját **DOM-fának** (***DOM tree***) nevezzük.
- Amikor egy böngésző egy HTML-dokumentumot tölt be, egy DOM-fát épít fel belőle a memóriában.
- A DOM-fa a böngészőkben a fejlesztői eszköztárral (DevTools) tekinthető meg.

A HTML-dokumentumok fák (2)

- A DOM-fa csomópontjai egy API-val rendelkező objektumok, amelyen keresztül manipulálható a dokumentum.
  - Ez képezi alapját a dinamikus weboldalak létrehozásának.
- A HTML szabvány két konkrét szintaxist biztosít a DOM-fák sorosításához és perzisztens tárolásához, a HTML- és az XML-szintaxist.

Karakterek

- A HTML-dokumentumok Unicode-karakterekből állnak.
- A dokumentumokat az UTF-8 karakterkódolással kell sorosítani.

HTML-dokumentumok építőelemei

- Az alábbi építőelemek állnak rendelkezésre, amelyek mindegyikét egy csomópont ábrázolja a DOM-fákban:
  - Szöveg
  - CDATA-szakaszok
  - Elemek
  - Dokumentumtípus-deklaráció (DOCTYPE)
  - Megjegyzések
- Ezek nem mindegyike áll rendelkezésre mindkét konkrét szintaxisban.
- A konkrét szintaxisokban eltérhet az ábrázolásmódjuk.

Szöveg (1)

- Szöveg elemek tartalmaként, megjegyzésekben és attribútumértékekben fordulhat elő.
- Az elemtartalomként előforduló szöveget egy szövegcsomópont ábrázolja a DOM-fákban.
    - A DOM-fa létrehozásakor egyetlen szövegcsomópont ábrázol egy elemben megjelenő minden folytonos és összefüggő szövegrészt.

Szöveg (2)

- Vegyük figyelembe, hogy a < és & karakterek speciális karakterek.
- Ökölszabályként szövegben le kell védeni őket.
  - Literálisan használhatók megjegyzésekben és CDATA-szakaszokban.
- Példa:

```
<!-- Fish emoticon, i.e., <>< -->
<code>&lt;>&lt;</code>
```

Szöveg (3)

- Unicode-karakterek mindkét szintaxisban megadhatók decimális vagy hexadecimális **numerikus karakterhivatkozásokkal** (***numeric character references***), mint például `&#8364;` és `&#x20AC;`, mindkettő az eurójelre (U+20AC) hivatkozik.
- A HTML-szintaxisban sok Unicode-karakter adható meg `&név;` formájú **nevesített karakterhivatkozásokkal** (***named character references***), mint például `&euro;`.
  - A karakterhivatkozás nevek listája: https://html.spec.whatwg.org/multipage/named-characters.html
  - Az XML-ben **egyedhivatkozásoknak** (***entity references***) nevezik ezeket a hivatkozásokat és csak öt áll belőlük rendelkezésre: `&`, `&apos;`, `>`, `<` és `&quot;`.

Szöveg (4)

- A karakterhivatkozások minden formája kifejtésre kerül a DOM-fa létrehozása előtt, azaz nem jelennek meg a DOM-fában.

CDATA-szakaszok (1)

- CDATA-szakaszok elemtartalomban előforduló szövegben használhatók nem levédett szöveg megadásához.
  - CDATA-szakaszokban a < és & karakterek elveszítik speciális jelentésüket, tehát literálisan is előfordulhatnak.
- A CDATA-szakaszok korlátozások nélkül csak az XML-szintaxisban használhatók.
- A HTML-szintaxisban csak nagyon korlátozottan használhatók.
  - Csak a `math` és az `svg` elemekben állnak rendelkezésre.
- Minden egyes CDATA-szakaszt egy adott fajta csomópont ábrázol a DOM-fákban.

CDATA-szakaszok (2)

- A CDATA-szakaszok hasznosak számítógépes programkódok beágyazásához, amelyekben a < és & karakterek gyakran fordulnak elő.
- Példa CDATA-szakaszra:

```
<pre><![CDATA[unsigned getbits(unsigned x, int p, int n) {
  return (x >> (p+1-n)) & ~(~0 << n);]]></pre>
}
```

Elemek (1)

- Az elemek a HTML-dokumentumok elsődleges építőelemei, amelyek egymásba ágyazása határozza meg dokumentum felépítését.
- Egy elem egy dokumentum egy jelentéssel bíró olyan része, amely az alábbiakkal rendelkezik:
  - A célját és jelentését világosan jelző név
  - Nulla vagy több attribútum (opcionális)
  - Tartalom (opcionális)
- Minden egyes elemet egy elemcsomópont ábrázol a DOM-fában.

Elemek (2)

- A HTML definiálja a rendlekezésre álló elemeket, meghatározva a hozzájuk megadható attribútumokat és a bennük megengedett tartalmat.

Elemek (3)

- Az elemeknek lehet tartalma.
  - **Nem üres elemeknek** (***non-void elements***) nevezzük azokat az elemeket, amelyeknek lehet tartalma.
    - A legtöbb HTML elem (például `a`, `div`, `p`, `table`) nem üres.
  - **Üres elemeknek** (***void elements***) nevezzük azokat az elemeket, amelyeknek nem lehet tartalma.
    - Az üres elemek közé tartoznak például a `base` és az `img`.
- Egy elem tartalmát ábrázoló csomópontok az elemcsomópont gyermekei a DOM-fában.

Elemek (4)

- Minden egyes elemnek van egy **tartalommodellje** (***content model***): az elem szükséges tartalmának egy leírása.
- Egy HTML elem tartalma meg kell, hogy feleljen a tartalommodelljében leírt követelményeknek.

Elemek (5)

- Az elemeket címkék határolják a konkrét szintaxisokban.
- A címkék két fő fajtája a **nyitócímke** (***start tag***) és a **zárócímke** (**end tag**).
  - A címkék egy harmadik fajtája a `<név/>` formájú **önlezáró címke** (***self-closing tag***), amely csak az XML-szintaxisban használható.
- Az elemek nem keverendőek össze a címkékkel: az utóbbiak az elemeket a konkrét szintaxisokban határoló jelölők.
  - Az elem és a címke kifejezéseket néha egymás szinonimáiként használják, noha ez pontatlan.

Attribútumok (1)

- Az elemek attribútumai a nyitócímkékben vagy – az XML-szintaxisban – az önlezáró címkékben adhatók meg.
- Általános formájuk `név="érték"` vagy `név='érték'`, ahol a határoló karakter nem fordulhat elő közvetlenül az értékben.
- Az attribútumokat általában nem közvetlenül jelenítik meg a böngészők.
  - Egy kivétel a `title` attribútum, amelynek értéke felugró tippként (*tooltip*) kerül megjelenítésre.
- Az attribútumok nem csomópontokként jelennek meg a DOM-fában, hanem az elemcsomópontok tulajdonságaiként kerülnek tárolásra.

Attribútumok (2)

- A konkrét szintaxisokban tetszőleges sorrendben adhatók meg az attribútumok.
- Egy adott nevű attribútum legfeljebb egyszer adható meg egy elemhez.
- A HTML-szintaxis néhány extra lehetőséget biztosít az attribútumok használatához, lásd például a logikai attribútumokat.

Attribútumok (3)

Az attribútumok többféle célra szolgálhatnak:

- Metaadatokat hordozhatnak:

```
<blockquote cite="https://whatwg.org/faq">
  <p>The Web Hypertext Application Technology Working Group
  (WHATWG) is a community of people interested in evolving
  the web through standards and tests.</p>
</blockquote>
```

- Az elemek jelentését finomíthatják:

```
<input type="text">
<input type="file">
```

- Az elemek viselkedését vagy megjelenését vezérelhetik:

```
<textarea name="review" readonly>Great for the price!</textarea>
```

Whitespace az elemekben (1)

- Amikor egy elem tartalmaként elemek jelennek meg, akkor a gyermekelemek előtt, között és után megengedettek *whitespace* karakterek.
- Az ilyen *whitespace* karakterek gyakran az elemek egymásba ágyazását mutatják.
- Ezek a *whitespace* karakterek a DOM-fában is megjelennek, azonban általában nem lényegesek, ezért a DevTools nem mutatja őket.
- A gyakorlatban a méret csökkentésének érdekében a weboldalakból – főleg az automatikusan generáltakból – gyakran hiányoznak az elemek egymásba ágyazását mutató *whitespace* karakterek.

Whitespace az elemekben (2)

Példák “egysoros” weboldalakra:

- https://lodash.com/
- https://alvarotrigo.com/fullPage/

DOCTYPE

- A DOCTYPE (`<!DOCTYPE html>`) a HTML-dokumentum legelején, a gyökérelem előtt jelenhet meg.
- Nem szolgál verzióazonosítóként!
- Az SGML-től örökölt régi maradványnak tekinthető.
- A célja mindössze az, hogy a böngészőt a megfelelő megjelenítési módba kapcsolja.
- A dokumentumtípus-deklarációt egy adott fajta csomópont ábrázolja a DOM-fákban.

Megjegyzések (1)

- Egy megjegyzés egy olyan szöveget ábrázol, amelyet a felhasználói ágensek figyelmen kívül hagynak.
- A megjegyzéseket mindkét konkrét szintaxisban `<!--` és `-->` határolók között kell megadni.
- A megjegyzések többféle célra szolgálhatnak:
  - **Dokumentálás**: A fejlesztők számára a megértést segítő magyarázatot fűzhetnek a HTML-kódhoz.
  - **Emlékeztetők**: Jelezhetik, hogy a kód egy része hiányzik vagy javításra szorul (TODO megjegyzések).
  - **Kód megjegyzésbe zárása**: Kódrészek megjegyzésbe zárása azok figyelmen kívül hagyására utasítja a felhasználói ágenseket.
- Minden egyes megjegyzést egy megjegyzés csomópont ábrázol a DOM-fákban.

Megjegyzések (2)

Példák megjegyzésekre:

- ```
  <!-- Intentionally left empty -->
  ```

- ```
  <!-- Google Tag Manager -->
  <script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
  new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
  j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
  'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j
  })(window,document,'script','dataLayer','GTM-XXXXXX');</script>
  <!-- End Google Tag Manager -->
  ```

- ```
  <!-- TODO: add navigation links -->
  ```

- ```
  <!--
  <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
  -->
  ```

Megjegyzések (3)

Feltételes megjegyzések:

- A feltételes megjegyzések (*conditional comments*) egy böngésző-specifikus lehetőség volt, amelyek kizárólag az Internet Explorer támogatott a 6–9 verziókban.
- Lehetővé tették olyan HTML kód használatát, amelyet csak az Internet Explorer bizonyos verziói értelmeztek és rendereltek.
- Lásd: Peter-Paul Koch. Conditional comments.

Megjegyzések (4)

Feltételes megjegyzések:

- ```
  <!--[if IE]>
  IE-specifikus HTML-kód
  <![endif]-->
  ```

- ```
  <!--[if IE 6]>
  IE 6-specifikus HTML-kód
  <![endif]-->
  ```

- ```
  <!--[if !IE]> -->
  bármely IE verzió által figyelmen kívül hagyott HTML-kód
  <!-- <![endif]-->
  ```
  ```

Megjegyzések (5)

Feltételes megjegyzések:

- Noha az Internet Explorer ma már egy nem támogatott termék, néhány webhely még mindig használ feltételes megjegyzéseket.
- Példa feltételes megjegyzéseket használó webhelyekre:
  - https://www.ietf.org/
  - https://kernel.org/
  - https://www.mozilla.org/
  - https://www.python.org/

Az elemek jelentése (1)

- Az elemeknek, attribútumoknak és attribútumértékeknek meghatározott jelentése (szemantikája) van.
  - Például az `ol` elem egy rendezett listát ábrázol, a `lang` attribútum pedig a tartalom nyelvét ábrázolja.
- A szerzők számára tilos az elemek, attribútumok és attribútumértékek a megfelelő rendeltetésüktől eltérő jelentésbeli céllal történő használata.

Az elemek jelentése (2)

- A HTML előző verzióiban rendelkezésre álló prezentációs lehetőségek többsége többé nem megengedett.
- A prezentációs jelölők problémái:
  - A prezentációs elemek használata rontja a hozzáférhetőséget.
  - Magasabb karbantartási költségek.
  - Nagyobb dokumentumméret.
- Csak a `style` attribútum és a `style` elem maradt meg, mint prezentációs jelölési lehetőség.

Az elemek jelentése (3)

Médiafüggetlenként lettek újrafogalmazva a következő, korábban prezentációs elemek:

Elem	HTML 4.01, XHTML 1.0	HTML5
`b`	Félkövér betű	Kulcsszavak
`i`	Kurzív betű	Hangnembeli változás
`hr`	Vízszintes választóvonal	Témaváltás
`small`	Kisebb betűméret	Lapszéli megjegyzés
`s`	Áthúzás	Pontatlan szöveg
`u`	Aláhúzás	Artikulálatlan jelölés (például hibásan írt szöveg)

Szemantikus HTML (1)

Példa:

- ```
  <h1>Ez egy felső szintű címsor</h1>
  ```
- ```
  <span style="display: block; font-size: 2em;
      font-weight: bold; margin: 0.67em 0;">
    Ez nem egy felső szintű címsor, noha annak látszik
  </span>
  ```

Forrás: Semantics in HTML (MDN)

Szemantikus HTML (2)

Előnyök:

- Jó a keresőoptimalizálás (*search engine optimization* – SEO) számára
- Javítja a karbantarthatóságot
- Akadálymentesség

Lásd: HTML: A good basis for accessibility (MDN)

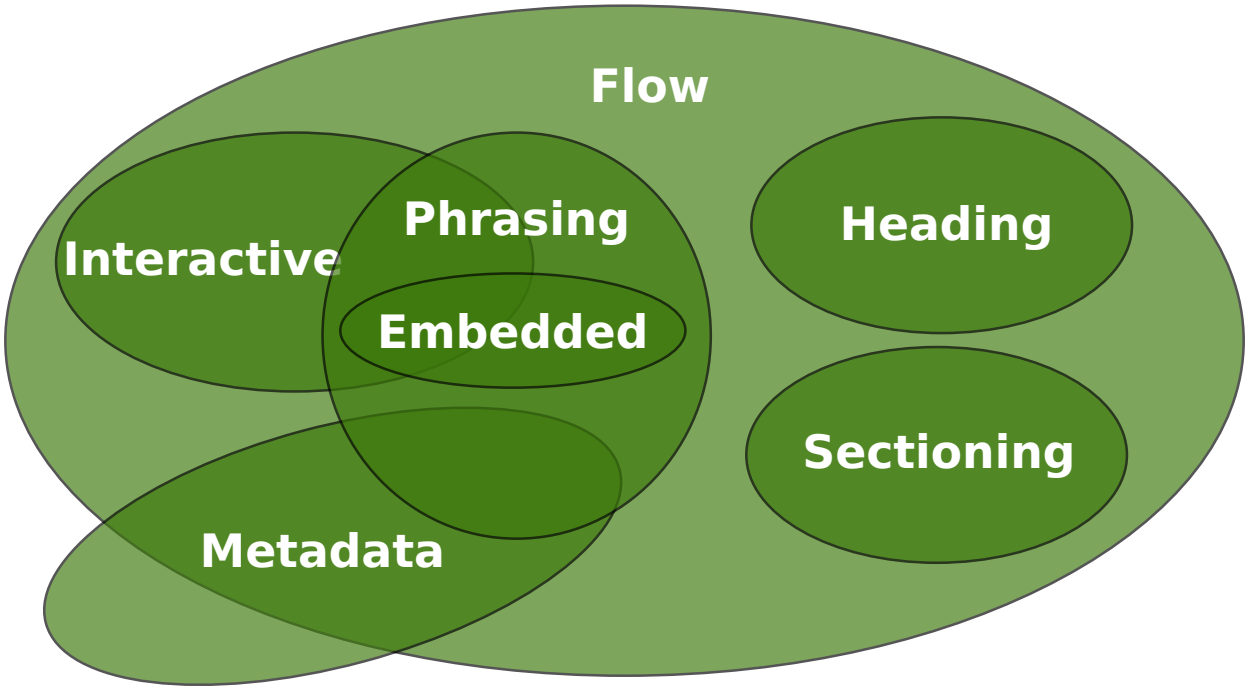
Szemantikus HTML (3)

Kapcsolódó fogalom: **akadálymentesség** (***web accessibility***)

- Az akadálymentesség azt jelenti, hogy a webhelyeket, eszközöket és technológiákat úgy tervezik és fejlesztik, hogy azokat fogyatékkal élő embertársaink is használhassák.
- Lásd: https://www.w3.org/mission/accessibility/

HTML elemek fajtái (1)

Minden egyes HTML elem nulla vagy több kategóriába sorolható az alábbiak közül:



1. Ábra. Forrás: https://html.spec.whatwg.org/images/content-venn.svg

HTML elemek fajtái (1)

Idegen elemek (***foreign elements***): A MathML és SVG névterekbe tartozó elemek.

A HTML5 új elemei (1)

A HTML5-ben a jobb tagoláshoz bevezetett elemek:

- `article`
- `aside`
- `figure`
- `footer`
- `header`
- `hgroup`
- `main`
- `nav`
- `section`

A HTML5 új elemei (2)

A HTML5-ben bevezetett további új elemek:

- `audio`
- `canvas`
- `data`
- `dialog`
- `math`
- `meter`
- `picture`
- `progress`
- `summary/details`
- `time`
- `video`
- …

Referencia a HTML elemekhez

Az összes HTML elem:

- HTML Standard – Index – Elements
- MDN Web Docs – HTML Elements Reference

Globális attribútumok

Valamennyi HTML elemhez megadható attribútumok:

- `class`
- `dir`
- `id`
- `lang`
- `style`
- `title`
- `xml:lang` (csak az XML szintaxisban ajánlott használni)
- Egyéni adat attribútumok (*custom data attributes*)
- ...

Lásd: Global attributes (MDN)

Globális attribútumok: `id` attribútum (1)

- Az `id` attribútum egy egyéni azonosítót szolgáltat az elemhez, amelyen előfordul.
- Értéke nem tartalmazhat *whitespace* karaktereket és legalább egy karakterből áll. Egy adott `id` érték legfeljebb egyszer fordulhat elő egy dokumentumban.
- Felhasználások:
  - Egy `id` érték erőforrásrészként használható egy olyan URI alkotásához, amely az azzal az azonosítóval rendelkező elemere hivatkozik.
  - Az `id` kiválasztóként használható CSS-stíluslapokon az azzal az azonosítóval rendelkező elem stilizálásához.

Globális attribútumok: `id` attribútum (2)

- Példa a használatra:

```
<nav id="toc">
  <h1>Table of Contents</h1>
</nav>
```

  - Az `id` használata erőforrásrészként:

```
<a href="#toc">Jump to Table of Contents</a>
```

  - Az `id` használata CSS-kiválasztóként:

```
#toc {
  background-color: lavender;
}
```

Globális attribútumok: `class` attribútum

- A szerzők az elemek kiterjesztéséhez használhatják a `class` attribútumot, gyakorlatilag saját elemeket létrehozva úgy, hogy közben a legalkalmasabb létező HTML elemet használják.
- A HTML elemeken megadott `class` attribútum értéke olyan tokenek egy szóközökkel elválasztott listája, amelyek azokat a különféle osztályokat ábrázolják, amelyekhez az elem tartozik.
- Példák a használatra:

```
<p class="author">...</p>
<p class="note">...</p>
<p class="warning">...</p>
<p class="note important">...</p>
```

Globális attribútumok: egyéni adat attribútumok (1)

- Egy **egyéni adat attribútum** (***custom data attribute***) egy olyan attribútum, amelynek neve a `data-` karakterlánccal kezdődik, amelyet legalább egy karakter követ a kötőjel után.
- Olyan egyéni adatok, állapot, annotációk és más hasonlók az oldal vagy alkalmazás számára történő privát tárolására szolgálnak, amelyekhez nincsenek megfelelőbb elemek vagy attribútumok.
- Minden HTML elemhez tetszőleges számú egyéni adat attribútum adható meg tetszőleges értékkel.

Globális attribútumok: egyéni adat attribútumok (2)

- Példa a használatra:

```
<form>
  <p>
    <label for="house">Ház:</label>
    <select name="house" id="house" required>
      <option value="">--Válassz egy lehetőséget--</option>
      <option value="gryffindor"
        data-color-primary="red"
        data-color-secondary="gold">Griffendél</option>
      <!-- ... --->
    </select>
  </p>
</form>
```

Globális attribútumok: egyéni adat attribútumok (3)

- Példa a használatra:

```
document.querySelectorAll('a').forEach(function (element) {
  element.addEventListener('mouseenter', function (event) {
    const a = event.currentTarget;
    const timer = setTimeout(function () {
      window.location.href = a.href;
    }, 3000);
    // Időzítő tárolása a data-timer attribútumban:
    a.dataset.timer = timer;
  });
  element.addEventListener('mouseleave', function (event) {
    // Időzítő lekérése a data-timer attribútumból:
    const timer = event.currentTarget.dataset.timer;
    clearTimeout(timer);
  });
});
```

Dokumentum metaadatok (1)

- A `head` elem metaadatokat tartalmaz a dokumentumról.
- Minden dokumentumnak pontosan egy `head` elemet kell tartalmaznia.
- Az alábbi elemek állnak rendelkezésre a `head` elemben:
  - `base`
  - `link`
  - `meta`
  - `noscript`
  - `script`
  - `style`
  - `title`
- A `title` elem kivételével a böngészők nem jelenítik meg a `head` elem tartalmát.

Dokumentum metaadatok (2)

További információk:

- Josh Buchea. HEAD: A simple guide to HTML <head> elements.

Dokumentum metaadatok: a base elem

- A base elem a relatív hivatkozások feloldásához a bázis-URI-t szolgáltató üres elem.
- Egy dokumentum legfeljebb egy base elemet tartalmazhat.
- A base elemet ritkán használják nagy nyilvános webhelyeken.
- Lásd: <base>: The Document Base URL element (MDN)
- Példa a használatra:

```
<base href="https://www.eg.com/">
```

Dokumentum metaadatok: a `link` elem (1)

- A `link` elem egy külső erőforrást társít a dokumentumhoz.
- A `rel` attribútum jelzi a kapcsolat fajtáját, megengedett értékei közé tartoznak például az `icon`, `stylesheet` vagy `license`.
- A leggyakrabban külső CSS-stíluslapok a dokumentumhoz való kapcsolására használják.
- Egy dokumentum nulla vagy több `link` elemet tartalmazhat a `head` elemben.
- Lásd: `<link>`: The External Resource Link element (MDN)

Dokumentum metaadatok: a `link` elem (2)

Példák a használatra:

- ```
  <link rel="stylesheet" href="screen.css">
  ```
- ```
  <link rel="icon" href="favicon.png">
  ```
- ```
  <link rel="license"
    href="https://creativecommons.org/licenses/by/4.0/">
  ```
  ```

Dokumentum metaadatok: a `meta` elem (1)

- A `meta` elem olyan metaadatok ábrázolására szolgál, amelyek nem fejezhetők ki a többi metaadat elemmel, mint például a `title` vagy a `link`.
- A `name` és a `content` attribútumok név-érték párok formájában adnak meg metaadatokat.
- Egy dokumentum nulla vagy több `meta` elemet tartalmazhat a `head` elemben.
  - Azonban bizonyos `meta` elemek (például a `<meta charset="UTF-8">`) legfeljebb egyszer fordulhatnak elő a `head` elemben.
- Lásd:
  - `<meta>`: The metadata element (MDN)
  - Meta Tags and Attributes that Google Supports (Google Search Central)

Dokumentum metaadatok: a `meta` elem (2)

Példák a használatra:

- `<meta charset="UTF-8">`
- `<meta name="author" content="Tim Berners-Lee">`
- `<meta http-equiv="Refresh" content="300">`

Dokumentum metaadatok: a `noscript` elem (1)

- A `noscript` elem olyan helyettesítő tartalmat ábrázol a felhasználói ágens számára, amelyet az akkor kell, hogy használjon, ha nem támogatja a szkripteket vagy a szkriptek le vannak tiltva.
- Ha a `head` elemben fordul elő, csak `link`, `meta` és `style` elemeket tartalmazhat.
- A `noscript` elemet figyelmen kívül hagyják a böngészők, ha támogatják a szkriptelést.
- Egy dokumentum nulla vagy több `noscript` elemet tartalmazhat a `head` elemben.
- Csak a HTML-szintaxisban áll rendelkezésre.
- Lásd: `<noscript>`: The Noscript element (MDN)

Dokumentum metaadatok: a `noscript` elem (2)

Példa a használatra:

```
<noscript>
  <link rel="stylesheet" href="noscript.css">
</noscript>
```

Dokumentum metaadatok: a `script` elem (1)

- A `script` elem végrehajtható (JavaScript) kódot ágyaz be vagy ilyenre hivatkozik.
- Egy dokumentum nulla vagy több `script` elemet tartalmazhat a `head` elemben.
- Lásd: `<script>`: The Script element (MDN)

Dokumentum metaadatok: a `script` elem (2)

Példák a használatra:

- ```
  <script>
    const startTime = performance.now();
    document.addEventListener("DOMContentLoaded", () => {
      const loadTime = (performance.now() - startTime).toFixed(2);
      console.log(`Page loaded in ${loadTime} ms`);
    });
  </script>
  ```

- ```
  <script src="js/metadata.min.js"></script>
  ```

Dokumentum metaadatok: a `style` elem (1)

- A `style` elem a dokumentumra alkalmazandó CSS-szabályokat tartalmaz.
- Egy dokumentum nulla vagy több `style` elemet tartalmazhat a `head` elemben.
- Kerülendő a `style` elem használata, a dokumentumszerzők számára inkább külső stíluslapok használata javasolt, amelyek `link` elemekkel társíthatók a dokumentumhoz.
- Lásd: `<style>`: The Style Information element (MDN)

Dokumentum metaadatok: a `style` elem (2)

Példa a használatra:

```
<style>
  ol, ul {
    padding: 0;
    margin: 0 0 1em 2em;
  }
  li {
    margin: 0 0 0.25em 0;
  }
</style>
```

Dokumentum metaadatok: a `title` elem

- A `title` elem a dokumentum címét szolgáltatja.
- Egy dokumentum legfeljebb egy `title` elemet tartalmazhat.
- Lásd: `<title>`: The Document Title element (MDN)
- Példa a használatra:

```
<title>HTML Standard, Edition for Web Developers</title>
```

Dokumentum metaadatok: gyakorlati felhasználás

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>The Mysteries of HTML</title>
  <meta name="description"
    content="Everything you'd like to know about HTML, but are afraid to ask.">
  <meta name="keywords" content="faq,html,markup">
  <meta name="robots" content="index,follow">
  <meta name="googlebot" content="notranslate">
  <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min.css">
  <link rel="icon" type="image/svg+xml" href="favicon.svg">
  <script
    src="https://cdn.jsdelivr.net/npm/jquery@3.7.1/dist/jquery.min.js"
    defer></script>
</head>
```

Nincs HTML séma

- DTD-k és XML sémák nem képesek kifejezni a HTML által támasztott valamennyi szintaktikai és szerkezeti követelményt.
- Lásd például az egyéni adat (`data-*`) attribútumokat.
  - Lásd: Embedding custom non-visible data with the data-* attributes

Konkrét szintaxisok (1)

- A HTML egy absztrakt nyelv, amelyben a dokumentumokat fák ábrázolják.
- A szabvány két konkrét szintaxist biztosít a dokumentumok sorosításához: a HTML- és az XML-szintaxist.

Konkrét szintaxisok (2)

- **HTML szintaxis**:
  - Bár nagyon hasonlít az SGML-hez és az XML-hez, egy külön nyelv saját elemzési szabályokkal.
  - Kompatibilis a legtöbb ősi böngészővel.
  - Fájlkiterjesztés: `.html`, `.htm`
  - Médiatípus: `text/html`
- **XML szintaxis**:
  - Az XML 1.0 és a XML 1.0 and the *Namespaces in XML 1.0* szabványokon alapuló szintaxis.
  - Nem határoz meg további szintaktikai követelményeket az XML-hez előírtakon túl.
  - XHTML szintaxisnak is nevezik.
  - Fájlkiterjesztés: `.xhtml`, `.xht`
  - Médiatípus: `application/xhtml+xml`

HTML szintaxis: példa

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Example Page</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="screen.css">
  </head>
  <body>
    <p>Visit W3C's website by clicking on the logo:</p>
    <a href="https://www.w3.org/">
      <img src="w3c.png" width="120" height="120"
        alt="W3C logo">
    </a>
  </body>
</html>
```

HTML szintaxis: DOCTYPE

- A HTML szintaxisban a `<!DOCTYPE html>` dokumentumtípus-deklaráció szükséges, amelynek célja mindössze annak biztosítása, hogy a dokumentum megjelenítése a szabványos módban történjen.
- A fenti rövid dokumentumtípus-deklaráció előállítására nem képes szoftverek használhatják helyette a `<!DOCTYPE html SYSTEM "about:legacy-compat">` dokumentumtípus-deklarációt.
- Lásd: The DOCTYPE

HTML szintaxis: kisbetű-nagybetű érzéketlenség

Az elem- és attribútumnevek kisbetű-nagybetű érzéketlenek:

- Az elemek és attribútumok neveinek megadásakor (még az idegen elemeknél is) tetszőlegesen keverhetők a kis- és nagybetűk.
- Ugyanabban a nyitó címkében soha nem fordulhat elő két vagy több olyan attribútum, amelyek nevei kisbetű-nagybetű érzéketlen hasonlítás esetén megegyeznek.

HTML szintaxis: speciális karakterek

- Elem szövege nem tartalmazhat `'<'` karaktert vagy félreérthető `'&'` karaktert.
  - Kivételt képeznek a `script`, `style`, `textarea` és `title` elemek.
- Attribútumérték nem tartalmazhat félreérthető `'&'` karaktert.
- Félreérthető (`'&'`) karakter:
  - Egy olyan `'&'` karakter, amelyet egy vagy több ASCII alfanumerikus karakter és egy `';'` karakter követ, amelyek nem felelnek meg a szabvány által definiált nevesített karakterhivatkozások egyikének sem (például `&nosuchchar;`).
    - Lásd: Named character references

HTML szintaxis: nem idézett attribútumérték szintaxis

- Ha egy, az üres karakterlánctól különböző attribútumérték nem tartalmaz egyetlen literális *whitespace* karaktert sem, akkor megadható az attribútumérték-határolók elhagyásával.
- Ekvivalensek például az alábbiak:

```
<img src="w3c.png" width="120" height="120"
  alt="W3C logo">
<img src=w3c.png width=120 height=120 alt="W3C logo">
```

HTML szintaxis: logikai attribútumok

- Sok attribútum logikai.
- Egy logikai attribútum jelenléte egy elemen az igaz értéket ábrázolja, hiánya pedig a hamis értéket.
- Ha megjelenik az attribútum, akkor értéke az üres karakterlánc kell, hogy legyen, vagy egy olyan érték, amely kisbetű-nagybetű érzéketlen hasonlítás esetén megegyezik az attribútum nevével.
- Ekvivalensek például az alábbiak:

```
<input type=checkbox checked name=agree disabled>
<input type=checkbox checked=checked name=agree
  disabled=disabled>
<input type='checkbox' checked='' name="agree"
  disabled="">
```

HTML szintaxis: üres elemek

- Az üres (*void*) elemeknek csak nyitócímkéjük van, tilos hozzájuk zárócímke megadása.
- Példák: `br`, `img`, `input`, `link`, `meta`

HTML szintaxis: opcionális címkék (1)

- Bizonyos elemek nyitó- és zárócímkéi elhagyhatók.
- Egy elem nyitócímkéjének elhagyása az itt tárgyalt esetekben nem azt jelenti, hogy az elem nem nincs ott (feltételezetten, de ott van)!
  - Például egy HTML dokumentumnak mindig van egy `html` nevű gyökéreleme, még akkor is, ha a `<html>` karakterlánc egyáltalán nem jelenik meg benne.
- Lásd: Optional tags

HTML szintaxis: opcionális címkék (2)

- Elhagyható egy `li` elem zárócímkéje, ha az elemet közvetlenül egy másik `li` elem követi vagy nincs több tartalom a szülő elemben.
  - Ekvivalensek például az alábbiak:
    - ```
      <ul>
        <li>Apple</li>
        <li>Banana</li>
        <li>Cherry</li>
      </ul>
      ```
    - ```
      <ul>
        <li>Apple
        <li>Banana
        <li>Cherry
      </ul>
      ```

HTML szintaxis: opcionális címkék (3)

- Elhagyható a `html` elem nyitócímkéje, ha elsőként nem egy megjegyzést tartalmaz.
- Elhagyható a `html` elem zárócímkéje, ha nem egy megjegyzés követi közvetlenül.
- ...

HTML szintaxis: opcionális címkék (4)

- Ha nem lényegesek az elemek közötti *whitespace* karakterek, ekvivalensek például az alábbiak:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

```
<!DOCTYPE html>
<title>Example Page</title>
<p>Hello, World!</p>
```

HTML szintaxis: idegen elemek

- Az idegen elemeknek vagy egy nyitó- és egy zárócímkéjük van, vagy egy önlezáróként jelölt nyitócímkéjük, amely esetben nem lehet zárócímkéjük.
- Ekvivalensek például az alábbi SVG elemek:

```
<circle cx="50" cy="50" r="50"></circle>
<circle cx="50" cy="50" r="50"/>
```

HTML szintaxis: XML lehetőségek

- A HTML-ben önlezáró címkéknek nevezett üreselem címkék csak az idegen elemeknél használhatók.
- Nem támogatottak a névtér-deklarációk, még idegen elemekhez sem.
- CDATA-szakaszok csak idegen tartalomban (MathML vagy SVG) használhatók.

XML szintaxis: példa

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Example Page</title>
    <link rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <p>Visit W3C's website by clicking on the logo:</p>
    <a href="https://www.w3.org/">
      <img src="w3c.png" width="120" height="120"
        alt="W3C logo"/>
    </a>
  </body>
</html>
```

XML szintaxis: DOCTYPE

- Az XML szintaxisban tetszőleges dokumentumtípus-deklaráció használható, megadása nem is kötelező.
- Az `application/xhtml+xml` médiatípussal továbbított dokumentumok megjelenítése mindig a szabványos módban történik.
- Lásd: Writing documents in the XML syntax

DOM (1)

- A DOM a Dokumentum Objektum Modellt (*Document Object Model*) jelenti.
- Egy alkalmazásprogramozási interfész (API) dokumentumok (főleg HTML és XML dokumentumok) eléréséhez és manipulálásához.
- Aktuális szabvány: DOM Living Standard (WHATWG)

DOM (2)

DOM-fa:

- Egy DOM-fa egy dokumentum memóriabeli ábrázolása.
- Egy olyan fastruktúra, amely különféle típusú csomópontokból áll.
- A csomópontok főbb fajtái:
  - `Attr`
  - `CDATASection`
  - `Comment`
  - `Document`
  - `DocumentType`
  - `Element`
  - `ProcessingInstruction`
  - `Text`

DOM (3)

Példa:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Sample Page</title>
 </head>
 <body>
  <p>Hello, World!</p>
  <!-- This is a comment -->
 </body>
</html>
```

```
├ DOCTYPE: html
└ html lang="en"
  ├ head
  │ ├ #text: ⏎␣␣
  │ ├ title
  │ │ └ #text: Sample Page
  │ └ #text: ⏎␣
  ├ #text: ⏎␣
  └ body
    ├ #text: ⏎␣␣
    ├ p
    │ └ #text: Hello, World!
    ├ #text: ⏎␣␣
    ├ #comment: This is a comment
    └ #text: ⏎␣⏎
```

DOM (4)

- Minden egyes csomópontot egy API-val rendelkező objektum ábrázol, amelyen keresztül manipulálható.
- A DOM interfészek Web IDL-ben kerülnek leírásra.

DOM (5)

Web IDL:

- A Web IDL egy interfészleíró nyelv, mely böngészőkben implementálható interfészek leírására szolgál.
- Aktuális szabvány: Web IDL Living Standard (WHATWG)
- Példa a használatra: `Node` interface

DOM (6)

- A HTML specifikáció a HTML elemek ábrázolásához a DOM interfészeket kiterjesztő további interfészeket határoz meg.
- Példák:
  - `meta` element
  - `img` element

DOM (7)

- A DOM nem csupán egy API, a HTML implementációk megfelelési kritériumai is DOM műveletekkel vannak meghatározva.
  - Példa: The `Navigator` object
- A specifikációk jórészt a DOM-fák segítségével vannak megfogalmazva, nem pedig a jelölőkével.

DOM (8)

- Egy DOM-fa szkriptekből manipulálható az oldalon.
- Példa:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>DOM Example</title>
  </head>
  <body>
    <p>User agent: <span id="ua"></span></p>
    <script>
      document.getElementById('ua').innerHTML = navigator.userAgent;
    </script>
  </body>
</html>
```

DOM (9)

- Eszközök:
  - Live DOM Viewer
- További információk:
  - MDN Web Docs – Document Object Model (DOM)
  - The Modern JavaScript Tutorial – DOM tree

A DOM használata (1)

Az ablak használata:

```
// Az ablak tulajdonságainak lekérése:
console.log(window.location);
console.log(window.origin);
console.log(window.navigator.userAgent);
console.log(window.devicePixelRatio);
console.log(window.innerWidth, window.innerHeight);

// Az ablak görgetése:
window.scrollTo(0, 0);
window.scrollTo({top: 0, left: 0, behavior: "smooth"});
```

A DOM használata (2)

Dokumentum metaadatok használata:

```
// A dokumentum médiatípusának lekérése:
console.log(document.contentType);

// A dokumentum bázis-URI-jának lekérése:
console.log(document.baseURI);

// A dokumentum címének lekérése:
console.log(document.title);

// A dokumentum címének beállítása:
document.title = 'Mastering the DOM';
```

A DOM használata (3)

Elemek használata:

```
// Egy elem tartalmának lekérése:
console.log(document.querySelector('h1').textContent);

// Egy elem tartalmának módosítása:
document.getElementById('now').textContent = new Date().toLocaleString();
document.querySelector('footer').innerHTML =
  "<p><small>This content is in the public domain.</small></p>";
```

A DOM használata (4)

Attribútumok használata:

```
// Attribútum értékének lekérése:
console.log(document.documentElement.lang)
console.log(document.querySelector('img#logo').alt);

// Attribútum értékének beállítása:
document.querySelector('a').target = 'blank_';
document.querySelectorAll('a')
  .forEach((element) => element.target = 'blank_');

// Egy elem osztályainak lekérése:
console.log(document.body.className);

// Egy elem osztályainak módosítása:
document.getElementById('copyright').classList.add('important');
```

A DOM használata(5)

```
// CSS-tulajdonságok értékének beállítása:
document.body.style.backgroundColor = 'aliceblue';
document.body.style.margin = '2rem';
document.body.style.marginTop = '1em';
document.getElementById('copyright').style.fontStyle = 'italic';

// CSS-tulajdonság számított értékének lekérése:
const fontSize = window.getComputedStyle(document.body).fontSize;
console.log(`Computed font size of body: ${fontSize}`);
```

A DOM használata (6)

Elem elrejtése és megjelenítése:

```
<aside>
  <div id="help">
    <!-- ... -->
  </div>
  <button onclick="toggleHelp()">Hide/Show Help</button>
</aside>
<script>
  function toggleHelp() {
    const element = document.getElementById('help');
    element.hidden = !element.hidden;
  }
</script>
```

Az onevent attribútumok használata nem ajánlott!

A DOM használata (7)

Elem elrejtése és megjelenítése:

```
<aside>
  <div id="help">
    <!-- ... -->
  </div>
  <button id="toggle-help-btn">Hide/Show Help</button>
</aside>
<script>
  function toggleHelp(event) {
    const element = document.getElementById('help');
    element.hidden = !element.hidden;
  }
  document.getElementById('toggle-help-btn')
      .addEventListener('click', toggleHelp);
</script>
```

A DOM használata (8)

Elemek dinamikus létrehozása:

```
<div id="container"></div>
<button id="add-item-btn">Add Item</button>
<script>
  function addItem(event) {
    const container = document.getElementById('container');
    const item = document.createElement('div');
    item.className = 'item';
    container.appendChild(item);
  }
  document.getElementById('add-item-btn')
    .addEventListener('click', addItem);
</script>
```

A DOM használata (9)

Elemek dinamikus törlése:

```
<div id="container"></div>
<button id="remove-children-btn">Remove Children</button>
<script>
  function removeChildren(event) {
    const container = document.getElementById('container');
    while (container.firstChild) {
      container.removeChild(container.firstChild);
    }
  }
  document.getElementById('remove-children-btn')
    .addEventListener('click', removeChildren);
</script>
```

A DOM, a HTML- és az XML-szintaxis összehasonlítása

A DOM, a HTML-szintaxis és az XML-szintaxis nem képes mind ugyanazt a tartalmat ábrázolni.

	HTML-szintaxis	XML-szintaxis	DOM
Névterek	nem	igen	igen
`noscript`	igen	nem	nem
A `-->` sztring megjegyzésben	nem	nem	igen

Hibakezelés (1)

- A HTML korábbi verzióitól eltérően az aktuális specifikáció az érvényes dokumentumok feldolgozása mellett bizonyos szintű részletességgel azt is előírja, hogy hogyan kell feldolgozni az érvénytelen dokumentumokat.
- A böngészők nagyon elnézőek a hibákkal szemben, automatikusan kijavítják a nem érvényes tartalmat.
  - Azonban ez a HTML-szintaxisú dokumentumokra vonatkozik!
- Lásd:
  - HTML Standard – Parsing HTML documents
    - An introduction to error handling and strange cases in the parser

Hibakezelés (2)

- A példák kipróbálása:
  - Vizsgáljuk meg a dokumentumokból felépülő DOM-fákat a böngészőben a webfejlesztő eszközökkel vagy a Live DOM Viewer révén!

Hibakezelés (3)

Példa:

```
<html>
<body><b>How is <i>it</b> rendered?</i></body>
</html>
```

Az érvénytelen dokumentumból egy olyan DOM-fa épül fel, amely megegyezik az alábbi érvényes dokumentumból felépülővel:

```
<html>
<head></head>
<body><b>How is <i>it</i></b><i> rendered?</i></body>
</html>
```

Hibakezelés (4)

Példa:

```
<html>
  <foo>
  <p bar class=>Does it <b>work,<i> or</b> not</i>?
  <
  </body>
</html>
```

HTML API-k

- A HTML, mint nyelv nem csak a címkékről és az elemekről szól.
- A modern weboldalak sok olyan API-t használnak, amelyek vagy a HTML szabvány részeként, vagy a WHATWG által fejlesztett más szabványokban vannak definiálva.
  - A HTML szabvány részeként definiált API-k:
    - Canvas API
    - History API
    - Web Storage API
  - Külön specifikációkban definiált API-k:
    - Fetch API
    - Fullscreen API
    - WebSockets

A böngészők fejlesztői eszköztára

- A fő asztali böngészök egy beépített fejlesztői eszköztárat tartalmaznak, amelyet DevTools-nak is neveznek.
- Dokumentáció:
  - Chromium, Google Chrome, Opera: Chrome DevTools
  - Firefox: Firefox DevTools User Docs
  - Safari: Web development tools
  - Chromium-based Edge: Microsoft Edge DevTools documentation

Böngészők fejlesztői kiadásai

- Google Chrome: Google Chrome for developers
- Firefox: Firefox Developer Edition

HTML szerkesztés: szerkesztők

Szabad és nyílt forrású szoftverek:

- Visual Studio Code (platform: Linux, macOS, Windows; licenc: *MIT License*) https://code.visualstudio.com/ https://github.com/Microsoft/vscode
  - Lásd: https://code.visualstudio.com/docs/languages/html
  - Ajánlott kiterjesztések:
    - Live Preview https://marketplace.visualstudio.com/items?itemName=ms-vscode.live-server https://github.com/microsoft/vscode-livepreview

HTML szerkesztés: Emmet (1)

- Szövegszerkesztő bővítmények HTML és CSS-kód írásának gyorsításához.
- Programozási nyelv: JavaScript
- Licenc: *MIT License*
- Webhely: https://emmet.io/
- Tároló: https://github.com/emmetio/emmet

HTML szerkesztés: Emmet (2)

- Számos szövegszerkesztőhöz rendelkezésre áll, mint például az Eclipse, NetBeans, Notepad++, Visual Studio Code, WebStorm.
  - https://emmet.io/download/
  - Emmet in Visual Studio Code
- Dokumentáció: https://docs.emmet.io/
  - Testreszabás: https://docs.emmet.io/customization/

HTML szerkesztés: Emmet (3)

- Az Emmet rövidítéseket biztosít, amelyek HTML vagy CSS-kóddá kerülnek kifejtésre.
- A rövidítések szintaxisa a CSS-kiválasztók szintaxisán alapul.

HTML szerkesztés: Emmet (4)

Példák Emmet-rövidítésekre:

`ul>li*3`

kifejtése:

```
<ul>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

HTML szerkesztés: Emmet (5)

Példák Emmet-rövidítésekre:

`section.chapter>h1{Introduction}+p`

kifejtése:

```
<section class="chapter">
  <h1>Introduction</h1>
  <p></p>
</section>
```

HTML szerkesztés: Emmet (6)

Példák Emmet-rövidítésekre:

```
ul>li*5>a[href=#chapter$]{Chapter $}
```

kifejtése:

```
<ul>
  <li><a href="#chapter1">Chapter 1</a></li>
  <li><a href="#chapter2">Chapter 2</a></li>
  <li><a href="#chapter3">Chapter 3</a></li>
  <li><a href="#chapter4">Chapter 4</a></li>
  <li><a href="#chapter5">Chapter 5</a></li>
</ul>
```

Segédeszközök: érvényesítők (1)

- A **megfelelés-ellenőrzők** (***conformance checkers***) – más néven **érvényesítők** (***validators***) – olyan szoftvereszközök, amelyek azt vizsgálják, hogy egy adott HTML-dokumentum megfelel-e a HTML szabvány szintaktikai, strukturális és szemantikai követelményeinek.

Segédeszközök: érvényesítők (2)

Nu Html Checker: egy megfelelés-ellenőrző, amely a parancssorból vagy online egy webes interfészen keresztül használható

- Programozási nyelv: Java
- Licenc: *MIT License*
- Webhely: https://validator.github.io/validator/
- Tároló: https://github.com/validator/validator
- A W3C által üzemeltetett példány: https://validator.w3.org/nu/

Segédeszközök: HTML-tisztító eszközök

Tidy: parancssori eszköz HTML-dokumentumok kijavításához és tisztításához

- Platform: Linux, macOS, Windows
- Programozási nyelv: C
- Licenc: *Tidy License*
- Webhely: https://www.html-tidy.org/
- Tároló: https://github.com/htacg/tidy-html5

Segédeszközök: webszerverek

http-server: egy egyszerű, nulla beállítást igénylő parancssori statikus HTTP szerver

- Programozási nyelv: JavaScript
- Licenc: *MIT License*
- Tároló: https://github.com/http-party/http-server
- Telepítés és használat:

```
$ npx http-server -o index.html

$ npm install -g http-server
$ http-server -o index.html
```

- A dokumenum itt elérhető: http://localhost:8080/index.html.

Segédeszközök: hosztingszolgáltatások

Surge: ingyenes statikus HTML publikálás a parancssorból

- Webhely: https://surge.sh/
- Dokumentáció: https://surge.sh/help/
- Telepítés és használat (az aktuális könyvtár kihelyezése):

```
$ npm install --global surge
$ surge --domain example.surge.sh
```

- A kihelyezett webhely itt elérhető: https://example.surge.sh/.