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

Mik a css szelektorok?

A CSS-selector a CSS-szabálykészlet része, amely valójában kiválasztja a stílust kívánó tartalmat.

i) Univerzális selector (Universal selector): Az univerzális selector helyettesítő karakterként működik, kiválasztva az oldal összes elemét

```
* {
   color: green;
   font-size: 20px;
   line-height: 25px;
}
```

ii) Elemtípus selector(**Element Type Selector)**: Ez a választó egy vagy több azonos nevű HTML elemhez illeszkedik.

```
ul {
   list-style: none;
   border: solid 1px #ccc;
}
```

```
<ul>
  <li>Fish</li>
  <li>Apples</li>
  <li>Cheese</li>
</ul>

<div class="example">
  <p>Example paragraph text.</p>
</div>

<ul>
  <li>Water</li>
  <li>Juice</li>
  <li>Maple Syrup</li>
</ul>
```

iii) ID-selector: Ez a selector illeszkedik minden olyan HTML-elemre, amelynek azonosító-attribútuma van, és ugyanaz az értéke, mint a selector-nak.

```
#container {
   width: 960px;
   margin: 0 auto;
}

<div id="container"></div>
```

iv) Osztály selector (Class selector): Az osztály selector az oldal összes olyan elemére illeszkedik, amelynek osztályattribútuma ugyanazon értékre van állítva, mint az osztály.

```
.box {
   padding: 20px;
   margin: 10px;
   width: 240px;
}
```

```
<div class="box"></div>
```

v) Descendant Combinator: A descendant selector, vagy pontosabban a Descendant Combinator lehetővé teszi két vagy több selector kombinálását, így konkrétabbá válhat a kiválasztás.

```
#container .box {
   float: left;
   padding-bottom: 15px;
}
```

```
<div id="container">
  <div class="box"></div>

  <div class="box-2"></div>
</div>

<div class="box"></div>
```

vi) Child Combinator: A Child Combinator-t használó selector hasonló egy Descendant Combinator selector-hoz, azzal a különbséggel, hogy csak a közvetlen gyermekelemeket célozza meg.

```
#container > .box {
   float: left;
   padding-bottom: 15px;
}
```

A selector minden olyan elemre illeszedik, amelynek van egy box osztálya, és amelyek a #container elem közvetlen gyermekei. Ez azt jelenti, hogy a descendant combinator –tól eltérően nem lehet más elem. .box – nak közvetlen gyermekelemnek kell lennie.

```
<div id="container">
  <div class="box"></div>

  <div>
    <div class="box"></div>
  </div>
</div>
```

vii) General Sibling Combinator: Az a selector, amely General Sibling Combinator-t használ, megfelel a testvérkapcsolatokon alapuló elemeknek. A kiválasztott elemek egymás mellett vannak a HTML-ben.

```
h2 ~ p {
   margin-bottom: 20px;
}
```

Ebben a példában az összes bekezdési elem (<p>) a megadott szabályok szerint fog stílusozni, de csak akkor, ha <h2> elemek testvérei. A <h2> és a <p> között más elemek is lehetnek, és a stílusok továbbra is érvényesek.

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
  <p>Paragraph example.</p>
</div>
```

viii) Adjacent Sibling Combinator: A Adjacent Sibling Combinator-t használó választó a pluszjelet (+) használja, és majdnem megegyezik az General Sibling Combinator-ral. A különbség az, hogy a megcélzott elemnek közvetlen testvérnek kell lennie, nem csak általánosan testvérnek.

```
p + p {
   text-indent: 1.5em;
   margin-bottom: 0;
}
```

Ebben a példában a megadott stílusok csak azokra a bekezdéselemekre lesznek alkalmazva, amelyek azonnal követik a többi bekezdéselemet. Ez azt jelenti, hogy az oldal első bekezdésének elemei nem kapják meg ezeket a stílusokat. Továbbá, ha két bekezdés között megjelenik egy másik elem, akkor a kettő második bekezdésében nem alkalmazzák a stílusokat.

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>

<div class="box">
  <p>Paragraph example.</p>
  <p>Paragraph example.</p>
</div>
```

ix) Attribútumválasztó (Attribute Selector): Az attribútumválasztó elemeket céloz meg a HTML-attribútumok jelenléte és / vagy értéke alapján, és szögletes zárójelet használva.

```
input[type="text"] {
   background-color: #444;
   width: 200px;
}

<input type="text">
```

Az attribútumválasztó deklarálható csak maga az attribútum használatával, érték nélkül:

```
input[type] {
   background-color: #444;
   width: 200px;
}
```

x) Pseudo-class: Egy Pseudo-class kettőspont karaktert használ annak az pseudo-állapotnak az azonosítására, amelyben egy elem lehet - például a hovered vagy az activated állapot.

```
a:hover {
   color: red;
}
```

xi) Pseudo-element: CSS Pseudo-elemet használunk az elem meghatározott részeinek stílusához.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p::first-line {
        color: #ff0000;
        font-variant: small-caps;
```

```
      }

      p::first-letter {
        color: #ff0000;
        font-size: xx-large;
      }

      h1::before {
        content: url(smiley.gif);
      }

      h1::after {
        content: url(smiley.gif);
      }

      ::selection {
        color: red;
        background: yellow;
      }
    </style>
  </head>
<body>
  <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.
  Lorem Ipsum has been the industry\'s standard dummy text ever since the 1500s,
  <h1>when an unknown printer took a galley of type and scrambled it to make a
  type specimen book.<h1></p>
</body>
</html>
```

Mi a contextual selector?

A kontextuális választó (contextual selector) egy adott elem adott előfordulására vonatkozik. Ez egy egyedi szelektorok sora, amelyet white space választ el (keresési minta), ahol csak a minta utolsó eleme van címezve, feltéve, hogy megfelel a megadott kontextusnak.

Ezenkívül ellenőrzi az osztály kontextusát a html fában, hozzárendelve a stílust az elemhez egy adott útvonalon keresztül, figyelembe véve a fa mélységi sorrendjét

```
table p { property: value; }
```

Mi a különbség az Pseudo-classes és pseudo-elements között?

A pseuso-osztály olyan szelektor, amely segít kiválasztani valamit, amit nem lehet egyszerű szelektorral kifejezni, például :hover. A pseudo-elem azonban lehetővé teszi számunkra, hogy olyan elemeket hozzunk létre, amelyek általában nem léteznek a dokumentumfában, például :: after.

Pseudo-classes

Az Pseudo-classes elemeket választanak ki, de bizonyos feltételek mellett, például amikor a testvérekhez viszonyított helyzetük van, vagy amikor egy adott állapotban vannak. Itt található a CSS3 pseudo- osztályainak listája:

a) Dynamic pseudo-classes

- :link
- :visited

- :hover
- :active
- :focus

b) UI element states pseudo-classes

- :enabled
- :disabled
- :checked

c) Structural pseudo-classes

- :first-child
- :nth-child(n)
- :nth-last-child(n)
- :nth-of-type(n)
- :nth-last-of-type(n)
- :last-child
- :first-of-type
- :last-of-type
- :only-child
- :only-of-type
- :root
- :empty

d) Other pseudo-classes

- :not(x) :target :lang(language)

Pseudo- elemek

A pseudo-elemek hatékonyan hoznak létre olyan új elemeket, amelyek nincsenek megadva a dokumentum jelölésében, és hasonlóan kezelhetők, mint egy normál elem.

- ::before
- ::after
- ::first-letter
- ::first-line
- ::selection

Mi az a Combinator választó?

A Combinator az a választó karakter, amely két választót kapcsol össze. Négyféle combinator létezik.

a) Descendant Combinator (space): A leszármazó választó megfelel minden elemnek, amely egy meghatározott elem leszármazottja.

A következő példa az összes <p> elemet kiválasztja a elemeken belül:

```
div p {
  background-color: yellow;
}
```

b) Child Combinator (>): A gyermekválasztó kiválaszt minden elemet, amely egy adott elem gyermeke.

A következő példa kiválasztja az összes olyan <p> elemet, amely egy elem gyermeke:

```
div > p {
  background-color: yellow;
}
```

c) Adjacent Sibling Combinator (+): A szomszédos testvérválasztó kiválaszt minden elemet, amely egy adott elem szomszédos testvére.

A következő példa kiválasztja az összes olyan <p> elemet, amely közvetlenül a elemek után helyezkedik el:

```
div + p {
  background-color: yellow;
}
```

d) General Sibling Combinator (~): Az általános testvérválasztó kiválaszt minden elemet, amely egy meghatározott elem testvére.

A következő példa kiválasztja az összes olyan <p> elemet, amely a elemek testvére:

```
div ~ p {
  background-color: yellow;
}
```

Mi a különbség az osztályválasztók (class selector) és az idválasztók (id selector) között?

A CSS-ben az osztályválasztó egy név, amelyet egy pont („.”) előz meg, az ID-választó pedig egy név, amelyet hash karakter („#”) előz meg. Az azonosító és az osztály közötti különbség az, hogy egy azonosító használható egy elem azonosítására, míg egy osztály többre is.

```
#top {
    background-color: #ccc;
    padding: 20px
}

.intro {
    color: red;
    font-weight: bold;
}
<div id="top">

<h1>Chocolate curry</h1>
```

```
<p class="intro">This is my recipe for making curry purely with chocolate</p>
<p class="intro">Mmm mm mmmmm</p>

</div>
```

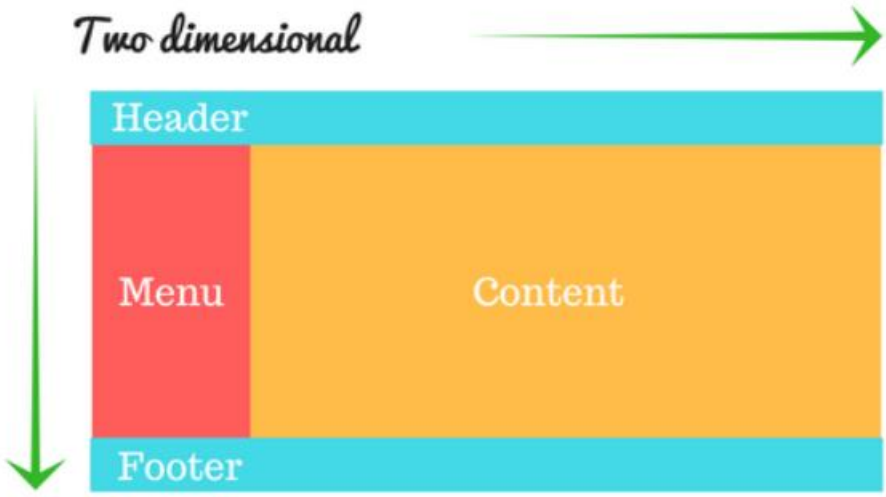
Mikor kell használni a css grid-et és a flexboxot?

A CSS Grid Layout kétdimenziós rendszer, vagyis oszlopokat és sorokat is képes kezelni, ellentétben a flexbox-szal, amely nagyrészt egydimenziós rendszer (vagy oszlopban, vagy sorban).

Alapvető különbség a CSS Grid és a Flexbox között az, hogy - a CSS Grid megközelítése layout-first (elrendezés oriental), míg a Flexbox megközelítése content-first (tartalom oriental). Ha jól ismeri tartalmát az elrendezés elkészítése előtt, akkor vakon válassza a Flexbox alkalmazást, ha pedig nem, akkor válassza a CSS Grid szolgáltatást.

A Flexbox elrendezés a legmegfelelőbb az alkalmazás (mivel többségük alapvetően lineáris) és a kisméretű elrendezés elemeihez, míg a Grid elrendezés nagyobb méretű elrendezésekhez készült, amelyek nem lineárisak.

Ha csak elrendezést kell megadnia sorként vagy oszlopként, akkor valószínűleg flexboxra van szüksége. Ha meg akar határozni egy rácsot, és két dimenzióban illeszteni bele a tartalmat - szüksége van a grid-re.



Mi a CSS BEM?

A BEM (Block Element Modifier) módszertan a CSS osztályok elnevezési konvenciója, annak érdekében, hogy a CSS fenntarthatóbb legyen a névterek definiálásával a hatókörrel kapcsolatos kérdések megoldására. A blokk egy önálló komponens, amely többször is felhasználható a projektekben, és "névtérként" működik az alkomponensek (Elements) számára. A módosítókat flag-ként használják, ha egy blokk vagy elem egy bizonyos állapotban van, vagy szerkezete vagy stílusa eltér.

```
/* block component */
.block {
}

/* element */
.block__element {
}

/* modifier */
.block__element--modifier {
}
```

Példa:

```
.button {
          display: inline-block;
          border-radius: 3px;
          padding: 7px 12px;
          border: 1px solid #D5D5D5;
          background-image: linear-gradient(#EEE, #DDD);
          font: 700 13px/18px Helvetica, arial;
}
.button--state-success {
          color: #FFF;
          background: #569E3D linear-gradient(#79D858, #569E3D) repeat-x;
          border-color: #4A993E;
}
.button--state-danger {
          color: #900;
}
<button class="button">
          Normal button
</button>
<button class="button button--state-success">
          Success button
</button>
<button class="button button--state-danger">
          Danger button
</button>
```

Előnyök:

- Modularitás: A blokkstílusok soha nem függenek az oldal más elemeitől, így soha nem fogsz problémát tapasztalni a lépcsőzetes működésből.
- Újrafelhasználhatóság: A független blokkok különböző módon történő összeállítása és intelligens újrafelhasználása csökkenti a fenntartandó CSS-kód mennyiségét.
- Felépítés: A BEM módszertana szilárd struktúrát ad a CSS-kódnak, amely továbbra is egyszerű és könnyen érthető.

Milyen előnyei vannak a CSS-sprite használatának?

A CSS-sprite több képet egyetlen nagyobb képpé egyesítenek. Ez az ikonok számára gyakran használt technika.

Előnyök:

Csökkentse a több képre vonatkozó HTTP-kérelmek számát (munkalaponként csak egyetlen kérelem szükséges). De a HTTP2 használatával több kép betöltése már nem jelent problémát.

Előre le kell tölteni azokat az eszközöket, amelyeket csak szükség esetén töltenek le, például :hover.

Ha több kép / ikon van, a böngésző mindegyikükhöz külön kérést indít a szerverhez. A sprite egy olyan módszer, amellyel mindegyiket / némelyiket (a kép típusát tekintve általában hasonlót hasonlítjuk össze. Például a jpg-t egy sprite-be fogja tenni) egy képbe kombinálni. Az ikon megjelenítéséhez állítsa be a magasságot, a szélességet és a háttér helyzetét.

Alternatívák:

Adat URI-k - lehetővé teszik a képadatok közvetlen stíluslapba ágyazását. Ezzel elkerülhetők a képek további HTTP-kérései, lényegében ugyanaz, mint a sprite, a képzeletbeli pozicionálás nélkül.

Ikon betűtípusok

SVG-k

Mi az a float tulajdonság és mit csinál?

A float CSS tulajdonság egy elemet helyez el a tároló bal vagy jobb oldalán, lehetővé téve a szöveges és beillesztett elemek körbevételét.

```
/* Keyword values */
float: left;
float: right;
float: none;
float: inline-start;
float: inline-end;

/* Global values */
float: inherit;
float: initial;
float: unset;
```

Sl.No	Value	Description
01.	none	Az elem nem lebeg, (ott jelenik meg, ahol a szövegben előfordul).
02.	left	Az elem a konténertől balra lebeg
03.	right	Az elem a konténer jobb oldalán lebeg

Sl.No	Value	Description
04.	initial	Ezt a tulajdonságot az alapértelmezett értékre állítja.
05.	inherit	Örökli ezt a tulajdonságot a szülő eleméből.

```
section {
  border: 1px solid blue;
  width: 100%;
  float: left;
}

div {
  margin: 5px;
  width: 50px;
  height: 150px;
}

.left {
  float: left;
  background: pink;
}

.right {
  float: right;
  background: cyan;
}
<section>
  <div class="left">1</div>
  <div class="left">2</div>
  <div class="right">3</div>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
     Morbi tristique sapien ac erat tincidunt, sit amet dignissim
     lectus vulputate. Donec id iaculis velit. Aliquam vel
     malesuada erat. Praesent non magna ac massa aliquet tincidunt
     vel in massa. Phasellus feugiat est vel leo finibus congue.</p>
</section>
```

Mi a tweening a css-ben?

A Pose-to -Pose lehetőség az, hogy néhány kulcsképet (keyframes) hozzon létre az egész sorrendben, majd később töltse ki a hiányosságokat. Ezeknek a hiányosságoknak a kitöltése tweening néven ismert. Ez két kép közötti köztes képkockák létrehozásának folyamata. Azt a benyomást kelti, hogy az első kép simán átalakult a másodikká. A CSS3-ban az ransforms (matrix, translate, rotate, scale etc.) module használható a tweenelés elérésére.

```
p {
  animation-duration: 3s;
  animation-name: slidein;
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
```

```
}
```

Magyarázza el a különbséget a visibility: hidden; és display: none; között? Milyen előnyei és hátrányai vannak a display:none használatának?

visibility:hidden egyszerűen elrejti az elemet, de ez helyet foglal és befolyásolja a dokumentum elrendezését.

display: none távolítja el az elemet a normál elrendezési folyamatból (DOM reflow okoz). Ez nem befolyásolja a dokumentum elrendezését, és nem foglal helyet.

Mi a DOM reflow?

A Reflow annak a webböngésző folyamatnak a neve, amelynek segítségével újraszámolhatja a dokumentumban lévő elemek helyzetét és geometriáját, a dokumentum egy részének vagy egészének újrarendezése céljából.

Reflow akkor következik be, amikor

- új elemet ad hozzá, távolít el vagy frissít egy elemet a DOM-ban
- az oldal tartalmának módosítása, pl. a beviteli mező szövegét
- áthelyez egy DOM elemet
- animálja a DOM elemet
- végezzen olyan elem mérését, mint például az offsetHeight vagy a getComputedStyle
- CSS stílus módosítása
- változtassa meg az elem className elemét
- stíluslap hozzáadása vagy eltávolítása
- méretezze át az ablakot
- scroll

Minimalizálja a böngésző reflow-t az alábbiakkal:

- Csökkentse a felesleges DOM mélységet. A DOM-fa egy szintjén történő változások a fa minden szintjén változásokat okozhatnak - egészen a gyökérig, és egészen a módosított csomópont gyermekeihez. Ez azt eredményezi, hogy több időt fordítanak a reflow végrehajtására.
- Csökkentse a CSS-szabályokat, és távolítsa el a fel nem használt CSS-szabályokat.
- Ha összetett renderelési változásokat hajt végre, például animációkat, akkor ezt a folyamaton kívül végezze el. Ennek eléréséhez használja az position-absolute vagy position-fixed
- Kerülje a felesleges komplex CSS-szelektorokat - különösen a leszármazott szelektorokat (descendant selectors) -, amelyek nagyobb CPU-energiát igényelnek a szelektorillesztéshez.

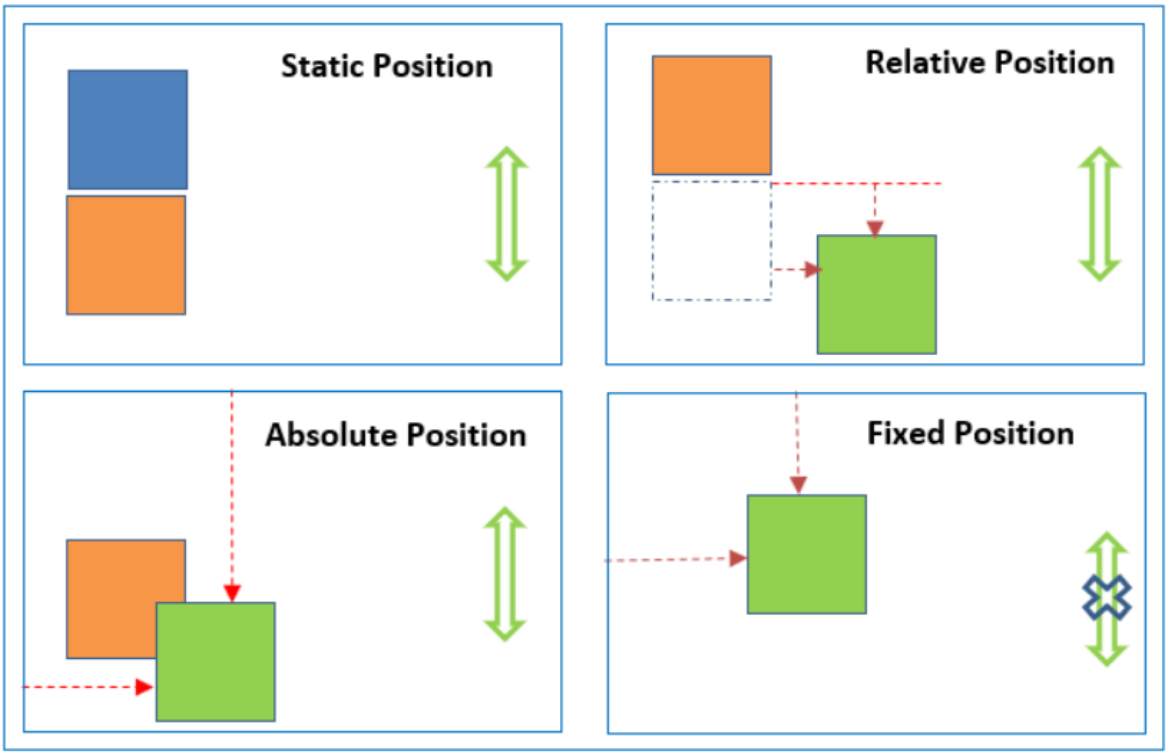
Mi a célja a z-indexnek és hogyan használják?

A z-index segít meghatározni az egymással átfedő elemek elhelyezésének sorrendjét. A z index alapértelmezett értéke nulla, és akár pozitív, akár negatív számot vehet fel. A magasabb z-indexű elem mindig magasabbra kerül, mint egy alacsonyabb index.

A z-index a következő értékeket veheti fel:

- Auto: Sets the stack order equal to its parents.
- Number: Orders the stack order.
- Initial: Sets this property to its default value (0).
- Inherit: Inherits this property from its parent element.

Magyarázza a CSS position property-t?



- absolute, helyezzen el egy elemet pontosan oda, ahová el szeretné helyezni. az abszolút pozíciót valójában az elem szülőjéhez viszonyítva állítjuk be. ha nincs elérhető szülő, akkor magához az oldalhoz viszonyít.
- relative, jelentése "önmagához viszonyítva". Beállítási helyzet: relatív; egy elemen és nincs más pozicionálási attribútum, ez nem lesz hatással az elhelyezésére. Lehetővé teszi a z-index használatát az elemen, és korlátozza az abszolút pozícionált gyermek elemek körét. Bármely gyermekelem abszolút pozicionált, ami a blokkon belül helyezkedik el.
- fix, az elem a magához a böngészőablakhoz képest helyezkedik el. nem változik az elem helye, ha görget, és így a fix elem ugyanazon a helyzeten marad.
- static: ez az alapértelmezett az elemekre. Az egyetlen ok, amiért bármikor beállítanánk egy elemet a static-ra az, hogy eltávolítsunk olyan pozícionálást, amelyet nem szeretnénk, hogy tartalmazzon az elem.

- sticky - A sticky pozícionálás a relatív és a fixed pozícionálás hibridje. Az elemet relatív helyzetben kell kezelni, amíg átlép egy meghatározott küszöböt, és ekkor fix helyzetűként kezeljük.

A különbség a blokk, az inline és az inline-block elem között?

a) Block Elements

A blokk elemek mindig egy új sorban kezdődnek. Ezenkívül egy teljes sor vagy szélesség helyet foglalnak el. A blokkelemek listája: <p>, <h1>, , <header>.

```
<p>
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Unde autem,
  consequatur deleniti nobis beatae quo dolore nemo corporis. Ad delectus
  dignissimos pariatur illum eveniet dolor rem eius laborum sed iure!
</p>

<p>
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Unde autem,
  consequatur deleniti nobis beatae quo dolore nemo corporis. Ad delectus
  dignissimos pariatur illum eveniet dolor rem eius laborum sed iure!
</p>
```

b) Inline elemek

Az inline elemek nem egy új sorban kezdődnek, hanem ugyanazon a soron jelennek meg, mint a mellettük levő tartalom. Néhány példa az inline elemekre: <a>, , és <img>.

Ha a margin-ről és padding-ről van szó, a böngészők másként kezelik az inline elemeket. Üres helyet adhatunk balra és jobbra egy inline elemnek, de nem adhat magasságot egy inline elem felső vagy alsó padding-jéhez vagy margójához.

```
<a href="#">Link</a>
<img src="https://picsum.photos/30" />
<span>Span</span>
<strong>Strong Player</strong>
```

c) Inline-Block elemek

Az inline-block elemek hasonlóak az inline elemekhez, csakhogy mind a négy oldalukhoz padding-et és margókat adhatnak. Az inline-block használatának egyik általános használata a navigációs hivatkozások vízszintes létrehozása. Néhány példa az inline-block elemekre: <input>, <button>, <select>, <textarea> stb.

```
input {
  width: 300px;
  height: 50px;
}

button {
  width: 100px;
  height: 50px;
  margin-top: 20px;
}
<input type="text" /> <button>Submit</button>
```

Mik a számlálók (counter) a CSS3-ban?

A CSS számlálók segítségével beállíthatja a tartalom megjelenését a dokumentumban való elhelyezkedése alapján. A CSS számláló használatához először inicializálni kell egy értéket a counter-reset tulajdonsággal (alapértelmezés szerint 0). Ugyanez a tulajdonság felhasználható annak értékének bármely meghatározott számra történő megváltoztatására is. Az inicializálás után a számláló értéke növelhető vagy csökkenthető az ellenszám növelésével. A számláló neve nem lehet "none", "inherit", or "initial", különben a deklarációt figyelmen kívül hagyják.

```
body {
  counter-reset: section;   /* Set a counter named 'section', and its initial value is 0. */
}

h3::before {
  counter-increment: section;   /* Increment the value of section counter by 1 */
  content: "Section " counter(section) ": ";  /* Display the word 'Section ', the value of
                                                 section counter, and a colon before the content
                                                 of each h3 */
}


<h3>Introduction</h3>
<h3>Body</h3>
<h3>Conclusion</h3>
```

Property	Description
content	A ::before and ::after pseudo-elemekkel együtt használják a létrehozott tartalom beillesztésére
counter-increment	Növel egy vagy több számláló értéket
counter-reset	Létrehoz vagy visszaállít egy vagy több számlálót

Melyiket preferálná a px, em% vagy pt között, és miért?

- A px finom szemcsézést biztosít és megtartja az összehangolást, mert garantáltan 1 képpont (vagy 1 képpont többszöröse) éles lesz. A px nem kaszkád (cascade), ez azt jelenti, ha a szülő betűmérete 20 képpont, a gyermek pedig 16 képpont. gyermek 16 képpontos lenne.
- em fenntartja a relatív méretet. reszponzív betűkészleteid lehetnek. em az 'm' betű szélessége a kiválasztott betűtípusban. Ez a koncepció azonban trükkös. Az 1em egyenlő az elem aktuális betűméretével vagy a böngésző alapértelmezett értékével. ha betűméretet küldött 16px-re, akkor 1em = 16 px. Az általános gyakorlat az alapértelmezett törzs betűméret beállítása 62,5% -ra (egyenlő 10px). emk kaszkádol (cascade)
- A% betűméretet állít be a törzs betűméretéhez képest. Ennélfogva ésszerű méretre kell állítania a törzs betűméretét. ez könnyen használható és kaszkádolható (cascade). például ha a szülő betűmérete 20 képpont, a gyermek betűmérete pedig 50%. gyerek 10px lenne.

- A pt (pontokat) hagyományosan nyomtatásban használják. 1pt = 1/72 hüvelyk, és rögzített méretű egység.

Mi a pseudo elem? Mi a pseudo osztály?

1. Pszeudo elem: CSS pseudo-elemet használunk az elem meghatározott részeinek stílusához.

Például:

Stílusozza az elem első betűjét vagy vonalát

Tartalom beillesztése az elem tartalma elé vagy után

Sl.No	Selector	Example	description
01.	::after	p::after	Helyezzen valamit az egyes elemek tartalma után
02.	::before	p::before	Helyezzen valamit az egyes tartalma elé
03.	::first-letter	p::first-letter	Kiválasztja az egyes elemek első betűjét
04.	::first-line	p::first-line	Kiválasztja az egyes elemek első sorát
05.	::selection	p::selection	Kiválasztja az elemnek a felhasználó által kiválasztott részét

2. Pseudo-osztályok: A pseudo-osztály egy elem speciális állapotának meghatározására szolgál.

Például:

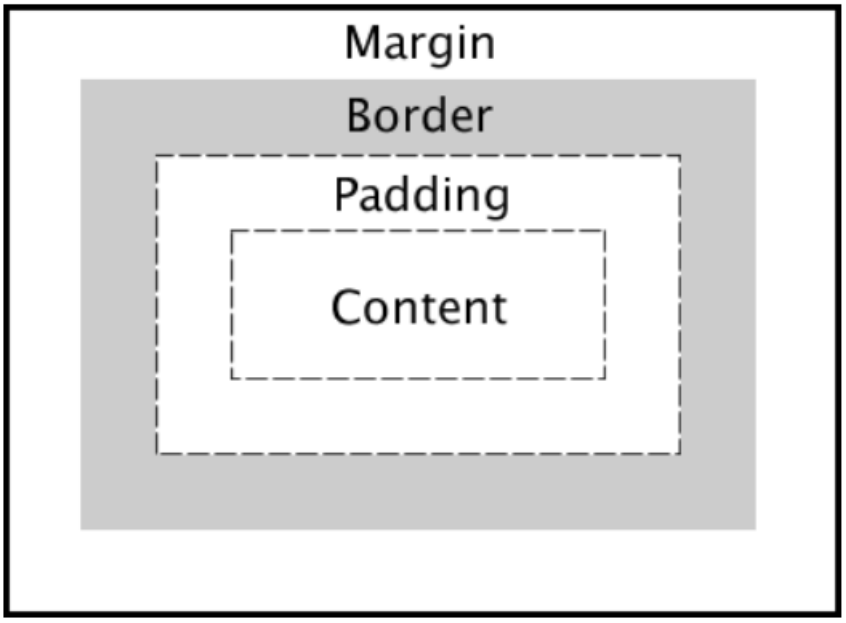
- Stílusoljon egy elemet, amikor a felhasználó fölé húzza az egeret
- A látogatott linkek és a nem látogatott linkek másként jelenjenek meg

Sl.No	Selector	Example	description
01.	:active	a:active	Kiválasztja az aktív linket
02.	:checked	input:checked	Minden bejelölt <input> elemet kiválaszt
03.	:disabled	input:disabled	Minden letiltott <input> elemet kiválaszt
04.	:empty	p:empty	Kiválaszt minden <p> elemet, amelynek nincs gyermeke

Sl.No	Selector	Example	description
05.	:enabled	input:enabled	Kiválasztja az összes engedélyezett <input> elemet
06.	:first-child	p:first-child	Kiválaszt minden <p> elemet, amely a szülő első gyermeke
07.	:first-of-type	p:first-of-type	Kiválaszt minden <p> elemet, amely a szülő első <p> eleme
08.	:focus	input:focus	Kiválasztja a fókuszált <input> elemet
09.	:hover	a:hover	Kiválasztja az egér fölött található hivatkozásokat
10.	:in-range	input:in-range	Kiválasztja az <input> elemeket egy megadott tartományon belüli értékkel
11.	:invalid	input:invalid	Az összes <input> elemet érvénytelen értékkel választja ki
12.	:lang(language)	p:lang(it)	Kiválaszt minden <p> elemet, amelynek lang attribútumértéke „it” kezdetű
13.	:last-child	p:last-child	Kiválaszt minden <p> elemet, amely a szülő utolsó gyermeke
14.	:last-of-type	p:last-of-type	Kiválaszt minden <p> elemet, amely a szülő utolsó <p> eleme
15.	:link	a:link	Kiválasztja az összes meg nem látogatott linket
16.	:not(selector)	:not(p)	Minden elemet kiválaszt, amely nem <p> elem
17.	:nth-child(n)	p:nth-child(2)	Kiválaszt minden <p> elemet, amely a szülő második gyermeke
18.	:nth-last-child(n)	p:nth-last-child(2)	Kiválaszt minden <p> elemet, amely a szülő második gyermeke,
19.	:nth-last-of-type(n)	p:nth-last-of-type(2)	Kiválaszt minden <p> elemet, amely a szülő második <p> eleme, az utolsó gyermektől számítva

Sl.No	Selector	Example	description
20.	:nth-of-type(n)	p:nth-of-type(2)	Kiválaszt minden <p> elemet, amely a szülő második <p> eleme
21.	:only-of-type	p:only-of-type	Kiválaszt minden <p> elemet, amely a szülőjének egyetlen <p> eleme
22.	:only-child	p:only-child	Kiválaszt minden <p> elemet, amely a szülője egyetlen gyermeke
23.	:optional	input:optional	Kiválasztja az <input> elemeket, nincs "required" attribútum
24.	:out-of-range	input:out-of-range	Kiválasztja az <input> elemeket, amelyek értéke egy megadott tartományon kívül esik
25.	:read-only	input:read-only	Kiválasztja az <input> elemeket egy "read-only" attribútummal
26.	:read-write	input:read-write	Kiválasztja az <input> elemeket "read-write" attribútum nélkül
27.	:required	input:required	Kiválasztja az <input> elemeket, amelyekhez "required" attribútum van megadva
28.	:root root		Kiválasztja a dokumentum gyökérelemét
29.	:target	#news:target	Kiválasztja az aktuális aktív #news elemet (rákattintott a horgony nevét tartalmazó URL-re)
30.	:valid	input:valid	Kiválasztja az összes érvényes értékű <input> elemet
31.	:visited	a:visited	Kiválasztja az összes meglátogatott linket

Magyarázza el a CSS „box modelljét” és az elrendezés összetevőit, amelyekből áll?



A CSS box modell a HTML elemek téglalap alakú elrendezési paradigmája, amely a következőkből áll:

- Content: A mező tartalma, ahol a szöveg és a képek megjelennek
- Padding: A tartalmat körülvevő átlátszó terület (azaz a border és a tartalom közötti tér nagysága)
- Border: A kitöltést (ha van) és a tartalmat körülvevő keret
- Margin: A border körülvevő átlátszó terület (vagyis a keret és a szomszédos elemek közötti tér nagysága)

```
/* top   right  bottom left  */
padding: 25px  50px   75px   100px;

/* same padding on all 4 sides: */
padding: 25px;

/* top/bottom padding 25px; right/left padding 50px */
padding: 25px 50px;

/* top padding 25px; right/left padding 50px; bottom padding 75px */
padding: 25px 50px 75px;
```

Magyarázza meg ezeknek a CSS egységeknek a jelentését a hosszúság kifejezésére?

- `cm` centimeters
- `em` elements (i.e., relative to the font-size of the element; e.g., 2 em means 2 times the current font size)
- `in` inches
- `mm` millimeters
- `pc` picas (1 pc = 12 pt = 1/6th of an inch)
- `pt` points (1 pt = 1/72nd of an inch)
- `px` pixels (1 px = 1/96th of an inch)

A CSS3-ban hogyan lehet kiválasztni (+ technikák)?

Minden <a> elem, amelynek href attribútumértéke „https” -el kezdődik.

```
a[href^="https"]
```

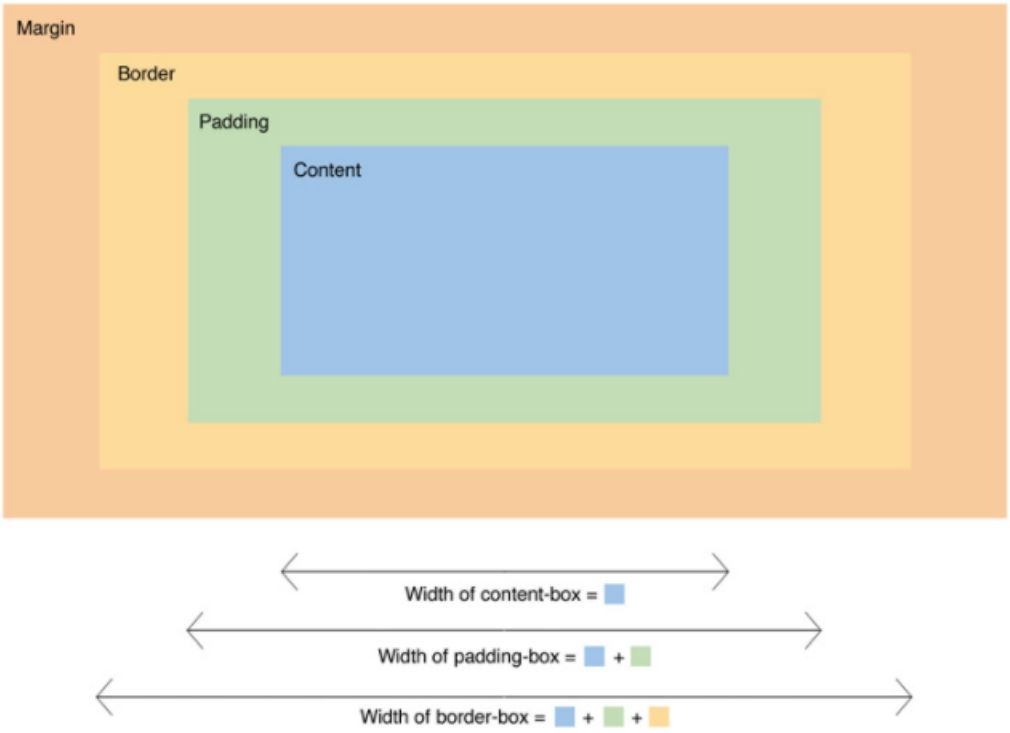
Minden <a> elem, amelynek href attribútumértéke „.pdf” -el végződik.

```
a[href$=".pdf"]
```

Minden <a> elem, amelynek href attribútumértéke tartalmazza a „css” részszöveget.

```
a[href*="css"]
```

Mi a célja a box-sizing tulajdonságának?



A box-sizing CSS tulajdonság meghatározza, hogyan kell kiszámítani az elem teljes szélességét és magasságát.

- content-box: az alapértelmezett szélesség- és magasságértékek csak az elem tartalmára vonatkoznak. A padding és a border hozzáadódik a doboz külsejéhez.
- padding-box: A szélesség és a magasság értékei az elem tartalmára és padding-ére vonatkoznak. A keret hozzáadódik a doboz külsejéhez. Jelenleg csak a Firefox támogatja a padding-box értékét.
- border-box: A szélesség és a magasság értékei a tartalomra, a padding-re és a border-re vonatkoznak.
- inherit: örökli a szülő elem box sizing-ját.

```
box-sizing: content-box;
width: 100%;
border: solid rgb(90,107,204) 10px;
padding: 5px;
```

Mi a különbség az RGBa, a HEX és a HSLa között?

- **RGB** (Red/Green/Blue) is a color model.

```
p {
  color: rgba(37, 84, 127, 1);
}
```

- **HEX (Hexadecimal color values)**

```
p {
  color: #25547f;
}
```

- **HSLa (Hue Saturation Lightness alpha)**

```
p {
  color: hsla(209, 55%, 32%, 1);
}
```

Mi az a CSS előfeldolgozó (preprocessor)?

Az előfeldolgozók kiterjesztik a CSS-t változókkal, operátorokkal, interpolációkkal, függvényekkel, mixekkel és még sok más használható eszközzel. A fejlesztés után ezeket a fájlokat CSS-be fordítják, amelyet minden böngésző megérthet. Az előfeldolgozó segít újrafelhasználható, könnyen karbantartható és bővíthető kódot írni CSS használat során.

CSS preprocessors

- SASS (SCSS)
- LESS
- Stylus
- PostCSS

Előnyök:

- A CSS karbantarthatóbbá válik.
- Könnyen beágyazott selector-ok.
- Változók a következetes theming-hez. Megoszthatja a theme fájlokat különböző projektek között.
- Mixin-ek az ismételt CSS előállításához.
- A kód felosztása több fájlra. A CSS fájlokat is fel lehet osztani, de ehhez HTTP kérésre lesz szükség az egyes CSS fájlok letöltéséhez.

Mi a különbség a CSS "resetting" és "normalizing" között?

1. Resetting: A CSS visszaállítások célja a beépített böngészőstílus eltávolítása. Például az összes elem margins, paddings, font-sizes visszaállítják.

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr,
acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong,
sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label, legend, table,
caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed,  figure, figcaption,
footer, header, hgroup,  menu, nav, output, ruby, section, summary, time, mark, audio, video {
   margin: 0;
   padding: 0;
   border: 0;
   font-size: 100%;
   font: inherit;
   vertical-align: baseline;
}
```

2. Normalizálás (Normalizing): célja, hogy a beépített böngésző stílusa következetes legyen a böngészők között. Javítja a böngésző általános függőségeinek hibáit is.

```
/*
  Correct the font size and margin on `h1` elements within `section`
  and `article` contexts in Chrome, Firefox, and Safari.
*/
 h1 {  font-size: 2em;  margin: 0.67em 0;}
```

css clear property ?

A clear tulajdonság meghatározza, hogy mely elemek lebeghetnek a float-olt elem mellett és melyik oldalán.

Sl.No	Properties	Description
01.	clear: none	Mindkét oldalon lebegő elemeket tesz lehetővé. Ez az alapértelmezett
02.	clear: left	A bal oldalon nem szabad lebegő elemeket használni
03.	clear: right	A jobb oldalon nem szabad lebegő elemeket használni
04.	clear: both	Sem a bal, sem a jobb oldalon nem szabad lebegni
05.	clear: inherit	Az elem örökli a szülő clear értékét

```
div {
  clear: left;
}
```

Hogyan közelítené meg a böngészőspecifikus stílusproblémák kijavítását?

Használjon külön stíluslapot, amely csak akkor töltődik be, ha az adott böngészőt használja. Ez a technika szerveroldali renderelést igényel.

Az autoprefixer segítségével automatikusan hozzáadhatja a vendor előtagjait a kódjához.

Használja a Reset CSS or Normalize.css fájlt.

```
/*Example: 01*/
.box-shadow {
  background-color: red;
  background-image: url(gradient-slice.png);
  background-image: -webkit-linear-gradient(top right, #A60000, #FFFFFF); /*Chrome and Safari*/
  background-image: -moz-linear-gradient(top right, #A60000, #FFFFFF);     /*Firefox*/
  background-image: -ms-linear-gradient(top right, #A60000, #FFFFFF);     /*Internet Explorer*/
  background-image: -o-linear-gradient(top right, #A60000, #FFFFFF);      /*Opera*/
  background-image: linear-gradient(top right, #A60000, #FFFFFF);
}

/*Example: 02*/
.box {
          -moz-border-radius: 15px;    /* Firefox */
          -webkit-border-radius: 15px; /* Safari and Chrome */
          border-radius: 15px;
}
```

Melyek a kedvenc képpótló technikáid, és melyiket használod mikor?

Technique: 01

```
h1#technique-one {
  width: 250px;
  height: 25px;
  background-image: url(logo.gif);
}
h1#technique-one span {
  display: none;
}
<h1 id="technique-one">
  <span>CSS-Tricks</span>
</h1>
```

Technique: 02

```
h1.technique-two {
  width: 2350px;
  height: 75px;
  background: url("images/header-image.jpg") top right;
  margin: 0 0 0 -2000px;
}
<h1 class="technique-two">
  CSS-Tricks
</h1>
```

Technique: 03

```
h1.technique-three {
  width: 350px;
  height: 75px;
  background: url("images/header-image.jpg");
  text-indent: -9999px;
}
<h1 class="technique-three">
  CSS-Tricks
</h1>
```

Technique: 04

```
h1.technique-four {
          width: 350px;
  height: 75px;
```

```
          background: url("images/header-image.jpg");
          text-indent: -9999px;
}
<h1 class="technique-four">
  <a href="#">
    <img src="images/header-image.jpg" alt="CSS-Tricks" />
  </a>
</h1>
```

Technique: 05

```
h1.technique-five {
          width: 350px;
  height: 75px;
          background: url("images/header-image.jpg");
}
h1.technique-five span {
  display: none;
}
<h1 class="technique-five">
  <img src="images/blank.gif" alt="CSS-Tricks" />
  <span>CSS-Tricks</span>
</h1>
```

Technique: 06

```
h1.technique-six {
          width: 350px;
          padding: 75px 0 0 0;
          height: 0;
          background: url("images/header-image.jpg") no-repeat;
          overflow: hidden;
}
<h1 class="technique-six">
  CSS-Tricks
</h1>
```

Technique: 07

```
h1.technique-seven {
          width: 350px;
  height: 75px;
          background: url("images/header-image.jpg") no-repeat;
}
h1.technique-seven span {
  display: block;
  width: 0;
  height: 0;
  overflow: hidden;
}
<h1 class="technique-seven">
          <span>CSS-Tricks</span>
</h1>
```

Technique: 08

```
h1.technique-eight {
          width: 350px;
  height: 75px;
          position: relative;
}
h1.technique-eight span {
  background: url("images/header-image.jpg");
  position: absolute;
  width: 100%;
  height: 100%;
}
<h1 class="technique-eight">
  <span></span>CSS-Tricks
</h1>
```

Technique: 09

```
h1.technique-nine {
  width: 350px;
  height: 75px;
  background: url("images/header-image.jpg") no-repeat;
```

```
  font-size: 1px;
  color: white;
}
<h1 class="technique-nine">
  CSS-Tricks
</h1>
```

Hogyan szolgálhatja ki oldalait a korlátozott funkciójú böngészők számára? Milyen technikákat használsz?

Graceful degradation: Az a gyakorlat, hogy a modern böngészőknek egy alkalmazást építenek, miközben biztosítják, hogy a régebbi böngészőkben is működőképes maradjon.

Progressive enhancement (Fokozatos fejlesztés): Az alkalmazás felépítése az alapszintű felhasználói élmény érdekében, de funkcionális fejlesztések hozzáadása, ha a böngésző támogatja.

caniuse: a funkció támogatásának ellenőrzése.

Autoprefixer: a vendor prefix automatikus beillesztéséhez.

Feature detection (Funkció észlelés): a Modernizr használatával.

CSS Feature queries (CSS Feature lekérdezések): a @support használatával

Milyen módon lehet vizuálisan elrejteni a tartalmat (és csak a képernyőolvasók számára elérhetővé tenni)?

Ezek a technikák kapcsolódnak az akadálymentességhez (a11y).

- visibility: hidden: Az elem azonban még mindig az oldal flow-ba van, és még mindig helyet foglal.
- width: 0; height: 0: Az elem egyáltalán ne foglaljon helyet a képernyőn, aminek eredményeként nem jelenik meg.
- position: absolute; left: -99999px: Pozíció a képernyőn kívülre.
- text-indent: -9999px: Ez csak a blokkelemek szövegén működik.
- Metadata: Például a Schema.org, az RDF és a JSON-LD használatával.
- WAI-ARIA: A W3C technikai specifikációja, amely meghatározza a weblapok akadálymentességének növelését.

Mik azok a média lekérdezések? Hogyan alkalmazhatja a médiára jellemző css szabályokat?

A média lekérdezések akkor hasznosak, ha módosítani szeretné webhelyét vagy alkalmazását az eszköz általános típusától (például nyomtatás vagy képernyő), illetve egyedi jellemzőktől és paraméterektől (például a képernyő felbontása vagy a böngésző nézetének szélessége) függően. A @media szabályt használja a CSS tulajdonságok blokkjának feltétel teljesülése esetén.

Media Types

Sl.No	Value	Description
01.	all	Alapértelmezett. Minden média típusú eszközhöz használható
02.	print	Nyomtatókhoz használják
03.	screen	Számítógép-képernyőkhöz, táblagépekhez, okostelefonokhoz stb.
04.	speech	Olyan képernyőolvasók számára használják, amelyek hangosan "olvassák" az oldalt

Media Features

Sl.No	Value	Description
01.	any-hover	A rendelkezésre álló beviteli mechanizmusok lehetővé teszik a felhasználó számára, hogy az elemek felett lebegjen?
02.	any-pointer	Van-e elérhető bemeneti mutatóeszköz, és ha igen, mennyire pontos?
03.	aspect-ratio	A nézetablak szélességének és magasságának aránya
04.	color	A kimeneti eszköz színkomponensére eső bitek száma
05.	color-gamut	A felhasználói kliens és a kimeneti eszköz által támogatott hozzávetőleges színtartomány
06.	color-index	Az eszköz által megjeleníthető színek száma
07.	grid	Az eszköz rács vagy bitkép
08.	height	A nézetablak magassága
09.	hover	Az elsődleges beviteli mechanizmus lehetővé teszi a felhasználó számára, hogy az elemek felett lebegjen?
10.	inverted-colors	A böngésző vagy az alapul szolgáló operációs rendszer megfordítja a színeket?

Sl.No	Value	Description
11.	light-level	Jelenlegi környezeti fényszint
12.	max-aspect-ratio	A megjelenítési terület szélessége és magassága közötti legnagyobb arány
13.	max-color	A kimeneti eszköz színkomponensenkénti maximális bitjeinek száma
14.	max-color-index	Az eszköz által megjeleníthető színek maximális száma
15.	max-height	A megjelenítési terület maximális magassága, például egy böngészőablak
16.	max-monochrome	A maximális bitszám "színenként" monokróm (szürkeárnyalatos) eszközön
17.	max-resolution	Az eszköz maximális felbontása dpi vagy dpcm használatával
18.	max-width	A megjelenítési terület, például egy böngészőablak maximális szélessége
19.	min-aspect-ratio	A megjelenítési terület szélessége és magassága közötti minimális arány
20.	min-color	A kimeneti eszköz színkomponensenkénti minimális bitjeinek száma
21.	min-color-index	Az eszköz által megjeleníthető színek minimális száma
22.	min-height	A megjelenítési terület minimális magassága, például egy böngészőablak
23.	min-monochrome	A minimális bitszám "színenként" monokróm (szürkeárnyalatos) eszközön
24.	min-resolution	Az eszköz minimális felbontása dpi vagy dpcm használatával
25.	min-width	A megjelenítési terület, például egy böngészőablak minimális szélessége
26.	monochrome	A bitek száma "színenként" egy monokróm (szürkeárnyalatos) eszközön
27.	orientation	A nézetablak tájolása (fekvő vagy álló módban)
28.	overflow-block	Hogyan kezeli a kimeneti eszköz a blokk tengelye mentén a nézetablakot

Sl.No	Value	Description
		túlcsordító tartalmat
29.	overflow-inline	Görgethető-e a nézetablakot a belső tengely mentén túlcsordító tartalom
30.	pointer	Az elsődleges beviteli mechanizmus mutatóeszköz, és ha igen, mennyire pontos?
31.	resolution	A kimeneti eszköz felbontása dpi vagy dpcm használatával
32.	scan	A kimeneti eszköz beolvasási folyamata
33.	scripting	Rendelkezésre állnak szkriptek (pl. JavaScript)?
34.	update	Milyen gyorsan módosíthatja a kimeneti eszköz a tartalom megjelenését
35.	width	A nézetablak szélessége

```
@media print {
  body { font-size: 10pt; }
}

@media screen {
  body { font-size: 13px; }
}

@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}

/* Nested within another conditional at-rule */
@supports (display: flex) {
  @media screen and (min-width: 900px) {
    article {
      display: flex;
    }
  }
}

/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) { }

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) { }

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) { }

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) { }

/* Extra large devices (large laptops and desktops, 1200px and up) */
```

```
@media only screen and (min-width: 1200px) { }
```

@media only screen?

Stíluslapokat rejt a régebbi böngészők elől.

A screen kulcsszó vonatkozik az eszköz fizikai képernyőjére vagy a böngésző nézetablakára?

Böngésző nézetablak (Browser's Viewport).

Hogyan valósítaná meg a nem szabványos betűtípusokat használó webdesign-t?

Használja a @ font-face -t, és adja meg a font-family-t és a különböző font-weight-eket.

Hogyan állapítja meg a böngésző, hogy mely elemek felelnek meg a CSS-selector-nak?

A böngészők a selector szerint kiszűrik az elemeket a DOM-ban, és felfelé haladnak a szülőelemek között, hogy meghatározzák az egyezéseket. Minél rövidebb a selector chain hossza, annál gyorsabban tudja a böngésző megállapítani, hogy az elem megfelel-e a selector-nak.

Például ezzel a választóval p span, a böngészők először megtalálják az összes elemet, és felfelé haladnak egészen a gyökérig, hogy megtalálják az <p> elemet. Egy adott esetében, amint talál egy <p> -ot, tudja, hogy a egyezik, és leállíthatja az egyezést.

Hogyan lehet feltételesen betölteni a css erőforrásokat?

@import: lehetővé teszi a stíluslap betöltését a fájl helyét reprezentáló elérési út (uri) használatával.

```
/* By default, include the "light" color theme for syntax highlighting */
@import "cdn.com/atom-one-light.min.css";
/* And if you’re in dark mode, have those rules superseded via a different stylesheet */
@media (prefers-color-scheme: dark) {
  @import "cdn.com/atom-one-dark.min.css";
}
```

matchMedia (): A matchMedia használatával a JavaScript-blokkokat csak akkor hajthatja végre, ha egy adott média lekérdezési feltétel teljesül. Ez azt jelenti, hogy csak akkor írja ki a CSS-t, ha a lekérdezés igaz:

```
if (window.matchMedia('screen and (min-width: 600px)')) {
  document.write('<link rel="stylesheet" href="css/small.css">');
}
```

Mit jelent * {box-sizing: border-box; } ? Milyen előnyei vannak?

- A dokumentum minden elemének tartalmaznia kell a padding-et és a border-t az elem belső méreteiben.
- Alapértelmezés szerint az elemeknek van box-sizing: content-box alkalmazva van, és csak a tartalom méretét veszik figyelembe.
- box-sizing: a border-box megváltoztatja az elemek szélességének és magasságának kiszámításának módját, a szegély és a padding is beleszámít a számításba.
- Az elem magasságát most content height + vertical padding + vertical border width határozza meg.
- Az elem szélességét most content's width + horizontal padding + horizontal border width jelenti
- A padding és a border figyelembe vétele a box modell részeként jobban mutat azzal, hogy a tervezők miként képzelik el a rácsokban a tartalmat.

List display property in CSS?

A display property megadja az elem megjelenítési viselkedését (a rendering box típusát).

```
p.ex1 {display: none;}
p.ex2 {display: inline;}
p.ex3 {display: block;}
p.ex4 {display: inline-block;}
```

Sl.No	Value	Description
01.	inline	Egy elemet inline elemként jelenít meg (például). Bármilyen magasság- és szélességi tulajdonságnak nincs hatása
02.	block	Egy elemet blokkelemként jelenít meg (például <p>). Új vonalon kezdődik, és a teljes szélességet lefedi
03.	contents	Eltünteti a tárolót, így az elem gyermekei a DOM következő szintjére emelkednek
04.	flex	Megjelenít egy elemet block-szintű flex container-ként
05.	grid	Az elem block-szintű grid container-ként jelenik meg
06.	inline-block	Egy elemet inline-szintű block container-ként jelenít meg. Maga az elem inline elemként van formázva, de alkalmazhat magasság és szélesség értékeket
07.	inline-flex	Egy elemet inline-szintű flex container-ként jelenít meg

Sl.No	Value	Description
08.	inline-grid	Egy elemet inline-level grid container-ként jelenít meg
09.	inline-table	Az elem inline szintű táblázatként jelenik meg
10.	list-item	Hagyja, hogy az elem elemként viselkedjen
11.	run-in	Az elemet blokkként vagy inline-ként jeleníti meg, a kontextustól függően
12.	table	Hagyja, hogy az elem úgy viselkedjen, mint egy <table> elem
13.	table-caption	Hagyja, hogy az elem úgy viselkedjen, mint egy <caption> elem
14.	table-column-group	Hagyja, hogy az elem úgy viselkedjen, mint egy <colgroup> elem
15.	table-header-group	Hagyja, hogy az elem <thead> elemként viselkedjen
16.	table-footer-group	Hagyja, hogy az elem úgy viselkedjen, mint egy <tfoot> elem
17.	table-row-group	Hagyja, hogy az elem úgy viselkedjen, mint egy <tbody> elem
18.	table-cell	Hagyja, hogy az elem <td> elemként viselkedjen
19.	table-column	Hagyja, hogy az elem úgy viselkedjen, mint egy <col> elem
20.	table-row	Hagyja, hogy az elem úgy viselkedjen, mint egy <tr> elem
21.	none	Az elem teljesen eltávolításra kerül
22.	initial	Ezt a tulajdonságot az alapértelmezett értékre állítja. Vegye kezdettől
23.	inherit	Örökli ezt a tulajdonságot a szülő eleméből. Vegye az öröklésről

Miben különbözik az responsive design az adaptive design?

1) Responsive design CSS média lekérdezésekkel változtatja meg a stílusokat a céleszköz, például a megjelenítés típusa, szélessége, magassága stb. alapján, és ezek közül csak egy szükséges ahhoz, hogy a webhely alkalmazkodjon a különböző képernyőkhöz.

A Responsive nem kínál annyi irányítást, mint az adaptív, de sokkal kevesebb munkát igényel mind az építkezés, mind a fenntartás érdekében.

2) Az adaptív tervezés static layouts-ot használ olyan töréspontok alapján, amelyek nem reagálnak az első betöltésük után. Az Adaptive úgy működik, hogy érzékeli a képernyő méretét és betölti a megfelelő elrendezést. Általában az adaptív webhely hat általános képernyőszélességet használ:

- 320 px
- 480 px
- 760 px
- 960 px
- 1200 px
- 1600 px

Az Adaptive hasznos egy meglévő webhely utólagos felszereléséhez, annak mobilbarátabbá tétele érdekében. Ez lehetővé teszi, hogy átvegye az irányítást a tervezésen, és továbbfejlesztheti sajátos, több nézetablakát (multiple viewports).

Mi az a retina grafika? Milyen technikákat használ a retina képernyők képeinek kezeléséhez?

Annak érdekében, hogy éles, szép megjelenésű grafikák legyenek, amelyek a lehető legjobban kihasználják a retina kijelzőit, lehetőség szerint nagy felbontású képeket kell használnunk. Mindig a legnagyobb felbontású képek használata befolyásolja a teljesítményt, mivel több bájtot kell elküldeni a hálózaton.

Ennek a problémának a leküzdéséhez reszponzív képeket használhatunk, a HTML5-ben előírtak szerint. Megköveteli, hogy a böngésző elérhetővé tegye ugyanazon kép különböző felbontású fájljait, és hagyja, hogy eldöntse, melyik kép a legjobb, például az srcset html attribútum és opcionálisan a méretek segítségével:

```
<div responsive-background-image>
  <img src="/images/foo.png" alt="bar" srcset="/images/foo.png 2x" />
</div>

<!-- It enable the browser to use a medium and large image based on
     the sizes of the image in the viewport
-->
<img
  src="/images/foo.png"
  alt="bar"
  srcset="/images/foo-medium.png 1024w,
               /images/foo-large.png 2048w,
               /images/foo.png 800w"
/>
```

Azok a böngészők, amelyek nem támogatják a HTML5 srcsetjét (azaz az IE11-et), figyelmen kívül hagyják és inkább az src-t használják. Ha valóban támogatnunk kell az IE11-et, és ezt a funkciót teljesítményi okokból szeretnénk biztosítani, használhatunk JavaScript-polyfill-t.

HTML5 picture Element

```
<picture>
   <source media="(min-width: 1024px)" srcset="foo-large.jpg  1024w, foo-medium.jpg 640w" sizes="50vw"
/>
   <source srcset="foo@2x.jpg 2x, foo.jpg 1x" />
   <img src="foo.jpg" alt="Bar" />
</picture>
```

Retina Display Media Query

```
/* 1.25 dpr */
@media (-webkit-min-device-pixel-ratio: 1.25), (min-resolution: 120dpi) {
    /* Retina-specific stuff here */
}

/* 1.3 dpr */
@media (-webkit-min-device-pixel-ratio: 1.3), (min-resolution: 124.8dpi) {
    /* Retina-specific stuff here */
}

/* 1.5 dpr */
@media (-webkit-min-device-pixel-ratio: 1.5), (min-resolution: 144dpi) {
    /* Retina-specific stuff here */
}
```

Van valami oka annak, hogy a translate ()-t szeretné használni az abszolút pozicionálás helyett, vagy fordítva?

Az translate() hatására a böngésző GPU-réteget hoz létre az elem számára, de az abszolút pozicionálási tulajdonságok megváltoztatása a CPU-t használja. Ezért a translete () hatékonyabb, és rövidebb renderelési időket eredményez a simább animációk számára.

A translate () használatakor az elem továbbra is elfoglalja eredeti terét (hasonló position: relative-hoz), ellentétben az abszolútpozícionálással.

Példa:

```
.thing {
  position: relative;
  top: 100px;
  left: 50px;
}
```

A fenti példában az elem 100 képpontos távolságra kerül felülről és 50 képpont távolságra az eredeti pozíció bal oldalától.

A transzformáció használata esetén: translate(x,y) használatával a relatív pozíció használatához nagyon hasonló vizuális eredményt kapunk. A fentiekkel megegyező eredmény érhető el a következővel:

```
.thing {
  transform: translate(50px, 100px);
}
```

Ebben az esetben az elem koordinátáit az x tengely mentén 50px-rel, az y-tengely mentén 100px-szel translating. A végeredmény vizuálisan megegyezik az előző pozíció példával.

Mondja el, mit csinálnak ezek a tag.ek és melyik előnyössebb ?

: A HTML a tartalom hangsúlyozására.

```
<em>Emphasized content...</em>
```

<b>: A <b> félkövér szöveget határoz meg

```
<p>This is normal text - <b>and this is bold text</b>.</p>
```

<abbr>: A HTML rövidítés elem (<abbr>) rövidítést jelent; az opcionális title attribútum kibővítést vagy leírást adhat a rövidítéshez.

```
The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.
```

<nav>: A <nav> meghatározza a navigációs hivatkozások halmazát.

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

<i>: A <i> tartalma általában dőlt betűvel jelenik meg.

```
<p>I looked at it and thought <i>This can't be real!</i></p>
```

<link>: A HTML <link> címkét egy külső erőforráshoz való hivatkozás definiálására használják. A HTML dokumentum <head> szakaszába kerül.

```
<head>
  <link rel="stylesheet" type="text/css" href="theme.css">
</head>
```

: Az elemet a környező szövegnél nagyobb jelentőségű szöveg azonosítására használják. Alapértelmezés szerint minden böngésző félkövér betűtípussal jeleníti meg a szöveget.

```
<strong>Strong text</strong>
```

<article>: Az <article> címke független, önálló tartalmat határoz meg.

```
<article>
  <h1>Google Chrome</h1>
  <p>Google Chrome is a free, open-source web browser developed by Google, released in 2008.</p>
</article>
```

Mi az At-Rule?

Az at-rule CSS utasítások, amelyek utasítják a CSS-t, hogyan kell viselkedni. Elkezdődnek egy előjellel, (@), @ után egy azonosítóval, és dentifier és includes-t tartalmaznak a következő pontosvesszőig,; vagy a következő CSS-blokk, amelyik előbb következik be.

```
/* General structure */
@IDENTIFIER (RULE);

/* Example: tells browser to use UTF-8 character set */
@charset "utf-8";
```

Sl.No	at-rules	Description
01.	@charset	Meghatározza a stíluslap által használt karakterkészletet.
02.	@import	Azt mondja a CSS motornak, hogy tartalmazzon egy külső stíluslapot.
03.	@namespace	Elmondja a CSS motornak, hogy minden tartalmát egy XML névtér előtagnak kell tekinteni.
04.	@media	Feltételes csoportszabály, amely akkor alkalmazza a tartalmát, ha az eszköz megfelel a média lekérdezéssel meghatározott feltétel feltételeinek.
05.	@supports	Feltételes csoportszabály, amely akkor fogja alkalmazni a tartalmát, ha a böngésző megfelel az adott feltétel kritériumainak.
06.	@page	Leírja az elrendezés változásainak szempontját, amelyet a dokumentum nyomtatása során alkalmaznak.
07.	@font-face	Leírja a letöltendő külső betűtípus aspektusát.
08.	@keyframes	Leírja a CSS animációs sorozat közbenső lépéseinek aspektusát.

Hogyan lehet eltávolítani a kép alatti rést?

Mivel a beillesztett elemekként szereplő képeket ugyanúgy kezeljük, mint a szövegeket, így van egy rés, amelyet a következők segítségével lehet eltávolítani:

```
<figure>
  <img style="display: block" src="..." alt="">
</figure>
```

Mi a progresszív renderelés (progressive rendering)?

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

Példák:

- 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 (vagy above-the-fold rendering) rangsorolása - Csak a minimális CSS / tartalom / szkriptek szerepeltetése szükséges ahhoz az oldalmennyiséghez, amelyet böngészőben a lehető leggyorsabb megjelenítenek, majd használhat halasztott szkripteket vagy a DOMContentLoaded / load esemény-t, hogy más erőforrásokba és tartalmakba töltse be a többit.
- Async HTML fragments - A HTML egyes részeinek átmosása a böngészőbe, amikor az oldal felépül a backend-ben.

Mi az mobile-first stratégia? Meg tudja magyarázni a különbséget a weboldal reszponzívra kódolása és a mobile-first stratégia használata között?

Ha egy weboldalt adaptívvá tesz, az azt jelenti, hogy egyes elemek reagálnak a méretének vagy más funkcióinak az eszköz képernyőméretéhez, jellemzően a nézetablak szélességéhez igazításával, CSS-média lekérdezések révén, például kisebb betűk mérete kisebb eszközökön.

```
@media (min-width: 601px) {
  .my-class {
    font-size: 24px;
  }
}
@media (max-width: 600px) {
  .my-class {
    font-size: 12px;
  }
}
```

A mobile-first stratégia szintén reszponzív, azonban egyetért abban, hogy alapértelmezés szerint meg kell határoznunk a mobileszközök összes stílusát, és csak később adjunk hozzá speciális reagálási szabályokat más eszközökhöz. Az előző példát követve:

```
.my-class {
  font-size: 12px;
}

@media (min-width: 600px) {
  .my-class {
    font-size: 24px;
  }
}
```

A mobile-first stratégiának két fő előnye van:

- A mobileszközökön jobban teljesít, mivel a rájuk vonatkozó összes szabályt nem kell semmilyen média-lekérdezéssel érvényesíteni.
- Tisztább kód megírására kényszerül a responsive CSS-szabályok miatt.

Mi az akadálymentesség (a11y) egy webalkalmazásban?

Az akadálymentesség arra vonatkozik, hogy a szoftver vagy hardver kombinációkat hogyan tervezik a rendszer hozzáférhetővé tétele a fogyatékossággal élő személyek számára, például:

- Látás károsodás
- Halláskárosodás
- Korlátozott kézügyesség

Például egy, az akadálymentességet szem előtt tartva kifejlesztett webhely szöveg-beszéd képességekkel vagy kimenettel rendelkezhet speciális Braille-hardverekhez, amelyek látássérültek számára készültek.

Mi az a UI / UX?

1) UI or User Interface (felhasználói felület): a termék vagy a weboldal elrendezése és a velük való interakció: az, hogy hol vannak a gombok, mekkorák a betűtípusok, és hogyan vannak a menük rendezve, mind a felhasználói felület elemei.

2) UX or User Experience (UX vagy felhasználói élmény): hogyan érzi magát egy termék vagy egy weboldal használatával kapcsolatban. Szóval, az új Apple Watch kinézete iránti szeretete vagy az izgalma, hogy végre van egy tablet méretű iPhone pl az UX.. Tehát a Facebook hírcsatorna új megjelenése a felhasználói felület megváltoztatását vonja maga után, és az adott oldalon való navigálás módja, ez a UX-t jelenti.

Melyik tulajdonsággal változtatják meg a betűtípust?

A font-family tulajdonság a betűtípus megváltoztatására szolgál.

Melyik tulajdonság segítségével növelhető vagy csökkenthető a betűtípus félkövér vagy világos megjelenése?

A font-weight tulajdonság arra szolgál, hogy növelje vagy csökkentse a betű félkövér vagy világos megjelenését.

Melyik tulajdonságot használjuk a szóösszetétel betűi közötti szóköz hozzáadásához vagy kivonásához?

A letter-spacing tulajdonság a szó alkotó betűk közötti szóköz hozzáadására vagy kivonására szolgál.

Melyik tulajdonsággal lehet space-t állítani egy mondat szavai között?

A word-spacing tulajdonsággal a szó szavai közötti szóközt lehet hozzáadni vagy kivonni.

Mely tulajdonságot használjuk a bekezdés szövegének behúzására?

A text-indent tulajdonság a bekezdés szövegének behúzására szolgál.

Melyik tulajdonságot használja a dokumentum szövegének igazításához?

A text-align tulajdonság a dokumentum szövegének igazítására szolgál.

Melyik tulajdonságot használják a szöveg aláhúzására, áthúzására (underline, overline, and strikethrough) ?

A text-decoration tulajdonság a szöveg aláhúzására, áthúzására és áthúzására szolgál.

Melyik tulajdonság használatos a csupa nagybetűs íráshoz (capitalize text) vagy a szöveg nagy- vagy kisbetűkké (uppercase or lowercase) alakításához?

A text-transform tulajdonság.

Melyik tulajdonság segítségével szabályozhatja a lista jelölőjének alakját vagy megjelenését?

A list-style-type lehetővé teszi a marker alakjának vagy megjelenésének szabályozását.

Hogyan állíthatom vissza egy tulajdonság alapértelmezett értékét?

A initial kulcsszóval visszaállíthatja az alapértelmezett értékre, amelyet az adott tulajdonság CSS specifikációja határoz meg.

Mi a specifitás (specificity)? Hogyan lehet kiszámítani a specifitást (specificity)?

Annak meghatározása, hogy mely css szabályt alkalmazzák egy elemre. Ez tulajdonképpen meghatározza, hogy mely szabályok élveznek elsőbbséget. A Inline style általában fontosabb, mint az

id, osztály, az univerzális választó (*). Az ID-választók specifitása magasabb, mint az attribútum-választóké.

Selector típusok

A selector típusok prioritása sorrendben növekszik:

Típusválasztók (pl. H1) és pseudo-elemek (pl .: :: before).

Osztályválasztók (pl. . example), attribútumválasztók (pl. [Type = " radio"]) és pseudo-classes (pl . :hover).

Azonosítóválasztók (például # example).

```
/*wins*/
a#a-02 { background-image : url(n.gif); }
a[id="a-02"] { background-image : url(n.png); }
```

A kontextusválasztók specifikusabbak, mint egy elemválasztó. A beágyazott stíluslap közelebb van a stílushoz. Az utolsó definiált szabály felülír minden korábbi, ütköző szabályt.

```
p { color: red; background: yellow }
p { color: green } // wins
```

Az osztályválasztó tetszőleges számú elemválasztónál fontosabb.

```
.introduction {} //wins
html body div div h2 p {}
```

Mit tud a CSS átmenetekről?

A CSS Transitions lehetővé teszi effektus hozzáadását, miközben egyik stílusról a másikra vált. Beállíthatja, hogy melyik tulajdonságra szeretné a transition-t, az időtartamát, módját (linear, ease, ease-in, ease-out, cubic-bezier) és késleltetheti a transition kezdetét.

Sl.No	Property	Description
01.	transition	Rövid tulajdonság a négy átmeneti tulajdonság egyetlen tulajdonságba állításához
02.	transition-delay	Megad egy késleltetést (másodpercben) az átmeneti effektushoz
03.	transition-duration	Megadja, hogy hány másodperc vagy milliszekundum alatt fejeződjön be egy átmenet
04.	transition-property	Megadja annak a CSS-tulajdonságnak a nevét, amelyre az átmeneti

Sl.No	Property	Description
		effektus vonatkozik
05.	transition-timing-function	Megadja az átmeneti effektus sebességgörbéjét

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* Safari prior 6.1 */
  transition: width 2s;
}
```

Melyek a különféle css filter, amelyeket használhat?

A CSS filter tulajdonság olyan grafikus effektusokat alkalmaz, mint az elmosódás vagy a színeltolás egy elemre. A filtereket általában a képek, a háttér és a szegélyek megjelenítésének beállítására használják.

```
img {
  -webkit-filter: brightness(200%); /* Safari 6.0 - 9.0 */
  filter: brightness(200%);
}
```

Sl.No	Filter	Description
01.	none	Alapértelmezett érték. Nincs hatás
02.	blur(px)	Homályos hatást alkalmaz a képre. Egy nagyobb érték több elmosódást eredményez.
03.	brightness(%)	Beállítja a kép fényerejét.
04.	contrast(%)	A kép kontrasztjának beállítása.
05.	drop-shadow(h-shadow v-shadow blur spread color)	Árnyékhatás alkalmazza a képet.
06.	grayscale(%)	A képet szürkeárnyalattá alakítja.
07.	hue-rotate(deg)	Színárnyalatot alkalmaz a képen. Az érték meghatározza a színkör körüli fokok számát

Sl.No	Filter	Description
08.	invert(%)	Megfordítja a képen szereplő mintákat.
09.	opacity(%)	Beállítja a kép fedettségi szintjét. Az átlátszatlansági szint az átláthatósági szintet írja le
10.	saturate(%)	Telíti a képet.
11.	sepia(%)	A képet szépiává alakítja.
12.	url()	Az url () függvény egy XML fájl helyét veszi át, amely megadja az SVG szűrőt, és horgonyt tartalmazhat egy adott szűrőelemhez. Példa: filter: url (svg-url # element-id)
13.	initial	Ezt a tulajdonságot az alapértelmezett értékre állítja.
14.	inherit	Örökli ezt a tulajdonságot a szülő eleméből.

különböző betűtípus tulajdonságokat?

Property	Description
font-style	Megadja a betűtípust.
font-variant	Megadja a betűtípusváltozatot.
font-weight	Megadja a betű súlyát.
font-size/line-height	Megadja a betűméretet és a vonalmagasságot.
font-family	Megadja a betűtípuscsaládot. Az alapértelmezett érték a böngészőtől függ
caption	A feliratozott vezérlők által használt betűtípust használja (például gombok, legördülő menü stb.)
icon	Az ikoncímkék által használt betűtípust használja
menu	A legördülő menükben használt betűtípusokat használja

Property	Description
message-box	A párbeszédpanelek által használt betűtípusokat használja
small-caption	A felirat betűtípusának kisebb változata
status-bar	Az állapotsor által használt betűtípusokat használja
initial	Ezt a tulajdonságot az alapértelmezett értékre állítja.
inherit	Örökli ezt a tulajdonságot a szülő eleméből.

Mi a különbség az em és a rem egységek között?

Az em és a rem egységek a betűméret CSS tulajdonságon alapulnak. Az egyetlen különbség az, hogy honnan öröklik értékeiket.

em egységek öröklik értéküket a szülő elem betűméretéből

```
.parent {
  font-size: 18px;
}
.child {
  font-size: 1.5em;
}
<div class="parent">
  I'm 15px
  <div class="child">
  I'm 30px, as expected
    <div class="child">
    I'm 60px, trouble starts!
      <div class="child">
      I'm 120px, now we're really in trouble!
      </div>
    </div>
  </div>
</div>
```

a rem egységek öröklik értéküket a gyökérelem betűméretéből (html)

```
.html {
  font-size: 16px;
}
.parent {
  font-size: 15px;
}
.child-rem {
  font-size: 2rem;
}
<div class="parent">
  I'm 15px
  <div class="child-rem">
  I'm 32px, as expected
    <div class="child-rem">
    I'm 32px, yep!
      <div class="child-rem">
      I'm 32px, like clockwork!
```

```
      </div>
    </div>
  </div>
</div>
```

Megjegyzés: A legtöbb böngészőben a gyökérelem betűmérete alapértelmezés szerint 16 px.

Mit jelent a !important a CSS-ben?

Ez felülírja a kaszkádot, és az adott stílusszabálynak adja a legnagyobb elsőbbséget.

```
p {
    color: red !important;
}
#thing {
    color: green;
}

<p id="thing">Will be RED.</p>
```

A style2.css betöltődik, mielőtt az (1) bekezdés megjelenik az oldalon?

```
<head>
    <link href="style1.css" rel="stylesheet">
</head>
<body>
    <p>Paragraph 1</p>
    <p>Paragraph 2</p>
    <link href="style2.css" rel="stylesheet">
</body>
```

igen

Be kell-e tölteni a style1.css fájlt, mielőtt a style2.css fájl betötődne?

```
<head>
    <link href="style1.css" rel="stylesheet">
    <link href="style2.css" rel="stylesheet">
</head>
```

- No

Mi a CSS positioning?

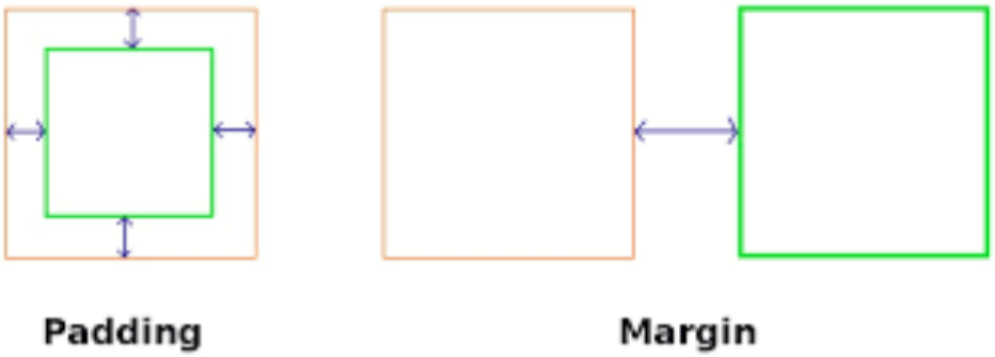
Keyword	Value	Description
position	static	Az alapértelmezett mód, a blokk elem a dokumentum flow-ban helyezkedik el.

Keyword	Value	Description
position	relative	A blokk elem a dokumentum flow-ban elfoglalt helyzetéhez képest helyezkedik el.
position	absolute	A blokkelem a konténeréhez (befoglaló elemzek képest) képest helyezkedik el.
position	fixed	A blokk elem az ablakhoz képest helyezkedik el, és mindig látszódik, ha götgetünk akkor is.
top	Number [px, cm, in...]	Helyezze a blokkot a referenciaponttól a megadott távolságra.
bottom	Number [px, cm, in...]	Helyezze a blokkot a referenciaponttól a megadott távolságra.
left	Number [px, cm, in...]	Helyezze a blokkot a referenciaponttól a megadott távolságra.
right	Number [px, cm, in...]	Helyezze a blokkot a referenciaponttól a megadott távolságra.

Mi a különbség a padding és a margó között?

1) A margót a külső elemekre alkalmazzák, így befolyásolva, hogy az elem milyen távolságra van a többi elemtől.

2) A padding-et az elem belsejére alkalmazzák, így befolyásolva, hogy az elem tartalma mennyire távol van a border-től.



Magyarázza a CSS3 új szolgáltatásait?

1. CSS3 szelektorok

Minden olyan E elemhez illeszkedik, amelynek attr attribútuma a val értékkel kezdődik. Más szavakkal, a val megegyezik az attribútum értékének kezdetével.

```
E[attr^=val]
/* Example */
a[href^='http://sales.']{color: teal;}
```

Bármely E elemhez illeszkedik, amelynek attr attribútuma val-val végződik. Más szavakkal, a val megegyezik az attribútum értékének végével.

```
E[attr$=val]
/* Example */
a[href$='.jsp']{color: purple;}
```

Bármely E elemhez illeszkedik, amelynek attribútuma attr egyezik az val bárhol az attribútumon belül. Hasonló az E [attr ~ = val] -hoz, azzal a különbséggel, hogy a val része lehet a szónak.

```
E[attr*=val]
/* Example */
img[src*='artwork']{
        border-color: #C3B087 #FFF #FFF #C3B087;
}
```

2. pseudo-osztályok

A CSS2 támogatja a felhasználói interakció pseudo-osztályokat, nevezetesen: :link, :visited, :hover, :active, and :focus.

A CSS3-ba még néhány pszeudo-osztályú szelektor került. Az egyik a: root választó, amely lehetővé teszi a tervezők számára, hogy rámutassanak a dokumentum gyökér elemére.

```
:root{overflow:auto;}
```

Az: first-child választó kiegészítéseként a: last-child került hozzáadásra. Ezzel kiválasztható egy szülőelem utolsó eleme.

```
div.article > p:last-child{font-style: italic;}
```

Új felhasználói interakció-pseudo-osztály választó került hozzá, a :target selector.

```
<style>
  span.notice:target { font-size: 2em; font-style: bold; }
</style>

<a href='#section2'>Section 2</a>
<p id='section2'>...</p>
```

A tagadás pszeudo osztályválasztó, :not kapcsolható szinte bármely más megvalósított választóval.

```
img:not([border]){ border: 1; }
```

3. CSS3 színek

A color kulcsszólista a CSS3 színmoduljában kibővült, és 147 további kulcsszíneket tartalmaz (amelyek általában jól támogatottak), a CSS3 számos más lehetőséget is kínál: HSL, HSLA, RGBA és Opacity.

```
div.halfopaque {
  background-color: rgb(0, 0, 0);
  opacity: 0.5;
  color: #000000;
}
div.halfalpha {
  background-color: rgba(0, 0, 0, 0.5);
  color: #000000;
}
```

4. Rounded Corners: border-radius

```
border-radius: 25px;
```

5 Drop Shadows

```
box-shadow: 2px 5px 0 0 rgba(72,72,72,1);
```

6. Text Shadow

```
text-shadow: topOffset leftOffset blurRadius color;
```

7. Linear Gradients

```
Syntax: background: linear-gradient(direction, color-stop1, color-stop2, ...);
/* Example */
#grad {
  background: linear-gradient(to right, red , yellow);
}
```

8. Radial Gradients

```
Syntax : background: radial-gradient(shape size at position, start-color, ..., last-color);
/* Example */
#grad {
  background: radial-gradient(red, yellow, green);
}//Default
#grad {
  background: radial-gradient(circle, red, yellow, green);
}//Circle
```

9. Multiple Background Images

A CSS3-ban nincs szükség elemre minden háttérképhez; lehetővé teszi számunkra, hogy egy elemhez több háttérképet adjunk, akár pseudo-elemekhez is.

```
background-image:
url(firstImage.jpg),
url(secondImage.gif),
url(thirdImage.png);
```

Mik a gradiensek a CSS-ben?

A CSS színátmeneteket a <gradient> adattípus képviseli, egy speciális típusú <image>, amely két vagy több szín közötti progresszív átmenetből áll. Háromféle színátmenet van:

lineáris (a linear-gradient () függvénnyel jön létre),

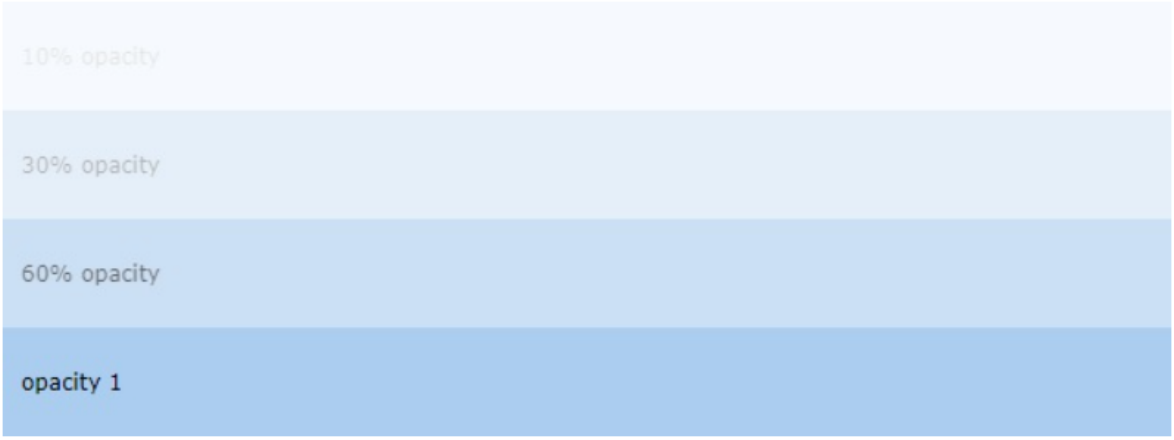
radiális (radial-gradient () létrehozva), és

kúp (a conic-gradient () függvénnyel jön létre).

Ismétlődő gradienseket is létrehozhatunk az repeating-linear-gradient(), repeating-radial-gradient(), and repeating-conic-gradient() függvényekkel.

```
/* Example - 01: A basic linear gradient */
.simple-linear {
  background: linear-gradient(blue, pink);
}

/* Example - 02: Changing the direction */
.horizontal-gradient {
  background: linear-gradient(to right, blue, pink);
}

/* Example - 03: Diagonal gradients */
.diagonal-gradient {
  background: linear-gradient(to bottom right, blue, pink);
}

/* Example - 04: Using angles */
.angled-gradient {
  background: linear-gradient(70deg, blue, pink);
}

/* Example - 05: Creating hard lines */
.striped {
   background: linear-gradient(to bottom left, cyan 50%, palegoldenrod 50%);
}
```

Mi a CSS opacity?



Az opacity CSS tulajdonság beállítja az elem átlátszatlanságát. Az átlátszatlanság az elem mögötti tartalom rejtettségének mértéke, és ellentétes az átlátszósággal.

```
div { background-color: lightblue; }
.light {
  opacity: 30%; /* Barely see the text over the background */
}
.medium {
  opacity: 60%; /* See the text more clearly over the background */
}
.heavy {
```

```
  opacity: 100%; /* See the text very clearly over the background */
}
<div class="light">You can barely see this.</div>
<div class="medium">This is easier to see.</div>
<div class="heavy">This is very easy to see.</div>
```

Hogyan alkalmazzák az öröklés fogalmát a CSS-ben?

Az öröklés olyan fogalom, amelyben a gyermekosztály örökölni fogja szülőosztályának tulajdonságait. A CSS-ben használják a hierarchia meghatározására a legfelső szinttől az alsó szintig. Az öröklött tulajdonságokat a gyermekosztály felülírhatja, ha a gyermek ugyanazt a nevet használja.

```
span {
  color: blue;
  border: 1px solid black;
}
.extra span {
  color: inherit;
}
```

Hogyan kezeli a felhasználói böngészőbeli különbségeket?

A CSS-ben található @supports lekérdezés nagyon hasznos lehet, ha a felhasználó jelenlegi böngészője rendelkezik bizonyos funkcióval. A @supports CSS at-szabály lehetővé teszi olyan deklarációk megadását, amelyek egy vagy több specifikus CSS-szolgáltatás böngészőjének támogatásától függenek. Ezt hívjuk szolgáltatás lekérdezésnek (feature query). A szabály elhelyezhető a kód legfelső szintjén, vagy beágyazható bármely más feltételes csoport-szabályba.

```
@supports (display: grid) {
  div {
    display: grid;
  }
}

@supports not (display: grid) {
  div {
    float: right;
  }
}
```

Mi a Cascade?

A Cascade az egyedi stílusszabályok súlyának (fontosságának) meghatározására szolgáló módszer, amely lehetővé teszi az ütköző szabályok rendezését, ha az ilyen szabályok ugyanazon választóra vonatkoznak.

```
P {color: white ! important} /* increased weight */
P (color: black} /* normal weight */
```

Mik a CSS vendor prefix-ek?

A vendor prefix-ek a CSS-szabványok kiterjesztései, amelyek hozzáadhatók ezekhez a funkciókhoz, hogy megakadályozzák az összeférhetetlenség kialakulását a szabvány kiterjesztésekor. Az alábbiakban felsoroljuk néhány általános platform CSS-szállítójának előtagjait.

-webkit-: Android, Chrome, iOS és Safari

-moz-: Mozilla Firefox

-ms-: Internet Explorer

-o-: Opera