CSS GRID

Tartalom

[Bevezető 2](#_Toc128984041)

[1. A CSS Grid alapfogalmai 2](#_Toc128984042)

[2. Grid létrehozása 4](#_Toc128984043)

[3. Rácsvonalak és egységek 9](#_Toc128984044)

[4. Automatikus rácselem elhelyezés 13](#_Toc128984045)

[5. Kézi rácselem elhelyezés 14](#_Toc128984046)

[6. Sorok elnevezése 17](#_Toc128984047)

[7. Grid Template Area 19](#_Toc128984048)

[8. Grid gap – rács rés 20](#_Toc128984049)

[9. CSS GRID legfontosabb tulajdonságainak áttekintése 21](#_Toc128984050)

[10. Különbség a grid és az inline grid között 24](#_Toc128984051)

# Bevezető

Az elkövetkezendő időszakban mélyrehatóan belemerülünk a CSS Grid tanulmányozásába, amely lehetővé teszi, hogy valódi reszponzív, kétdimenziós rácselrendezéseket hozzunk létre a Pure CSS használatával.

Tanmenetünk a következő lesz:

- CSS Grid Layout modul specifikációja

- CSS-rács-elrendezések tervezésére és végrehajtására az éles környezetben

- CSS Grid speciális elrendezései (amelyek kiterjednek a teljes elrendezésekre, a többoszlopos elrendezésekre, a kártyaelrendezésekre, a beágyazott elrendezésekre és még az interaktív vászonmenükre is.)

A CSS Grid Layout modul nagyjából mindent megváltoztat az eddig tanult webes elrendezésekkel kapcsolatban, és új lehetőségeket fog nyitni előtettek a layoutok tervezésre.

A CSS Grid egy alapvető CSS-elrendezési modul, akárcsak a blokk, az inline, a flex, a táblázat és így tovább, amelyekkel már megismerkedtetek, és ezek az előzetes ismeretek szükségesek is ahhoz, hogy megértsétek a css grid működését.

# A CSS Grid alapfogalmai

A **Grid Container** bármely olyan elem, amelynek megjelenítési tulajdonsága rácsra, vagyis grid-re van állítva, tehát amikor egy konténer tulajdonságának és értékének megadjuk: display:grid.

Egyetlen nézetben vagy egyetlen dokumentumban a böngészőben annyi vagy annyi rácstárolód lehet tetszés szerint, amennyire csak szükséged van, és rácstárolókat is beágyazhatsz más rácskonténerekbe.

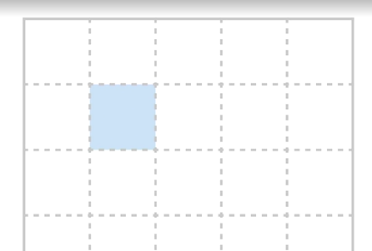
A rácstároló közvetlen első szintű leszármazott elemei rácselemekké válnak. Ezek automatikusan a rácsra kerülnek, hacsak másként nem adod meg, és minden rácselemet pontosan elhelyezhetsz a rácson belül, pontosan oda, ahol meg szeretnéd megjeleníteni.

Fontos megjegyezni, hogy a Flexboxhoz hasonlóan csak ***a rácskonténer első szintű leszármazottai kerülnek a rácsra***. Ezen leszármazottak gyermekeit egyáltalán nem érinti a rács.

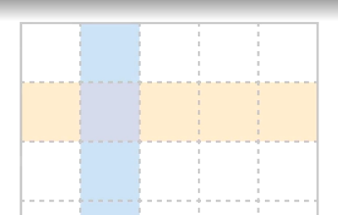
A rácstartály négy rácsvonallal rendelkezik a dobozonként, ezek tulajdonképpen a rács külső szélei. Általában további sorokat és oszlopokat ad hozzá sor- és oszlopsorokkal a rács cellákba történő szegmentálásához.

A rácsvonalakra egy szám hivatkozik, amely a rácstároló külső éleivel kezdődik és végződik, így a bal és a felső élek általában 1. sornak számítanak, majd minden sor kap egy növekményes számot, ahogyan a fenti képen is láthattátok.

A CSS Grid tiszteletben tartja az aktuális dokumentum szövegirányát, vagy akár az aktuális elemet is, így balról jobbra haladva a bal külső él az 1. sor , az utolsó sor pedig a jobb külső él. A jobbról balra haladó beállításoknál – arab szövegű oldalaknál - pedig a számozás megfordul, kezdve a jobb külső éllel 1. sorként, majd a bal külső éllel végződve.

Hivatkozhatsz egy rácsvonalra a számával, vagy a rácsvonalaknak egyedi neveket is adhatsz, majd az egyéni nevükkel hivatkozhatsz rájuk.

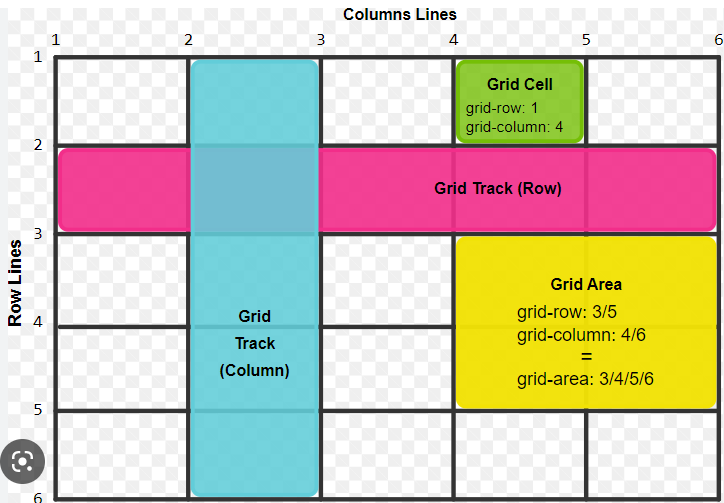
A **rácscella** két szomszédos vízszintes és függőleges vonal közötti tér, ahol a rácsoszlop és a rácssor metszi egymást. Tehát nagyjából ugyanaz, mint az Excelben egy táblázatcella.

A **rácspálya** két vagy több szomszédos, vízszintes vagy függőleges rácsvonal közötti tér. A sorsávok vízszintesek, az oszlopok függőlegesek.

A rácsterület egy vagy több cellát fed le, és a rácsterület célja egy olyan terület meghatározása a rácson belül, amelyre hivatkozni lehet a rácselemek elhelyezésekor.

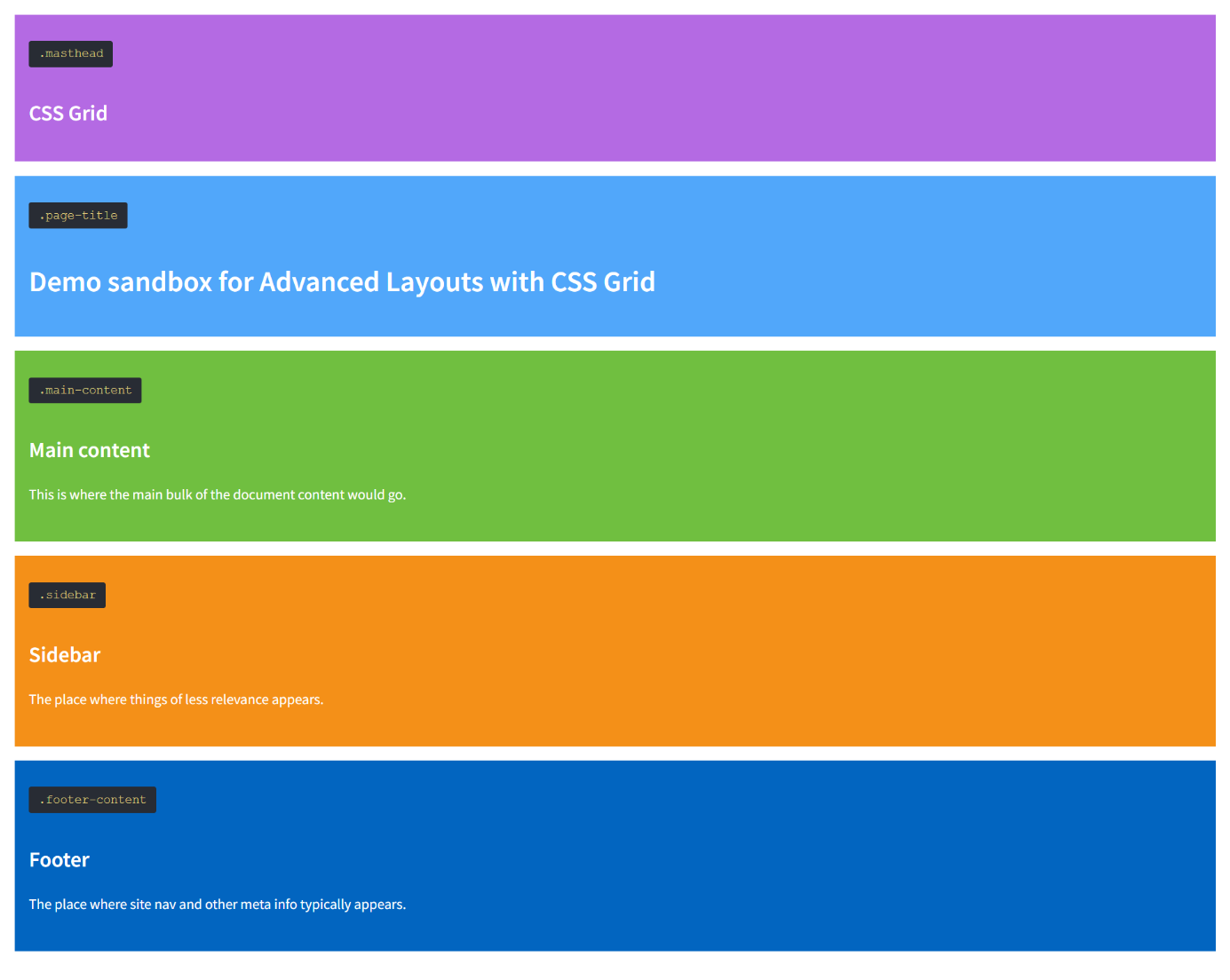
Az egyes rácssávok között helyet adhatsz hozzá egy rácsrés vagyis „grid gap” hozzáadásával. A rácsrés egységes távolságokat hoz létre az egyes sávok között, de megadhatsz csak oszlopközt, vagy csak sorközt az oszlopokhoz és a sorokhoz egyaránt.

Fontos, ha rácssávot – grid gap-et használsz, gyakorlatilag üres teret hozol létre az egyes rácssávok között, és ez azt jelenti, hogy bármilyen szín vagy elem jelenik meg a rács mögött, át fog látszani ezen a rácsrésen.



# Grid létrehozása

A következő elrendezést fogjuk létrehozni az első lépésben:



Hozzatok létre egy üres mappát, és egy index.html dokumentumot. A dokumentum body részébe a következő html tagek kerüljenek:

<body class="site">

<a class="skip-link screen-reader-text" href="#content">Skip to content</a>

<header class="masthead">

<code class="identifier"><pre>.masthead</pre></code>

<h2>CSS Grid</h2>

</header><!-- .masthead -->

<div class="page-title">

<code class="identifier"><pre>.page-title</pre></code>

<h1>Demo sandbox for Advanced Layouts with CSS Grid</h1>

</div><!-- .page-title -->

<main id="content" class="main-content">

<code class="identifier"><pre>.main-content</pre></code>

<h2>Main content</h2>

<p>This is where the main bulk of the document content would go.</p>

</main><!-- #content .main-content -->

<aside class="sidebar">

<code class="identifier"><pre>.sidebar</pre></code>

<h2>Sidebar</h2>

<p>The place where things of less relevance appears.</p>

</aside><!-- .sidebar -->

<footer class="footer-content">

<code class="identifier"><pre>.footer-content</pre></code>

<h2>Footer</h2>

<p><span class="bg">The place where site nav and other meta info typically appears.</span></p>

</footer><!-- .footer-content -->

</body>

Másold be a kódot a dokumentumodba.

Hozz létre egy css fájlt és csatold az oldalhoz.

A fejlécbe húzd be a következő betűtípust:

<link href="https://fonts.googleapis.com/css?family=PT+Serif:400,400i,700,700i|Source+Sans+Pro:400,400i,600,600i" rel="stylesheet">

Ha megnézzük ennek a dokumentumnak a HTML-kódját, láthatjuk, hogy van egy törzsünk, azaz body egy „site” osztállyal, amely egy sor ***első szintű leszármazottat tartalmaz***, olyan osztályokkal, mint „skip-link”, „masthead” és „page- title" és "maincontent". A böngészőben ezek az elemek egymás után, egyedi blokkként jelennek meg.

Most a CSS Grid segítségével fogunk kísérletezni különböző elrendezésekkel ezekhez a dobozokhoz, vagyis blokkokhoz.

Az első lépés egy grid-tároló deklarálása, és ebben az esetben a tároló a törzselem a "site" osztállyal, mert ez az összes többi elemet tartalmazza. 2 db külső css-fájlt hozzunk létre ehhez, az egyik neve az legyen: grid.css a másik main.css. Csatold először a main-t majd a grid.css-t az index.html oldaladhoz. Először megformázzuk együtt a main.css-t.

Mivel már kiválóan tudtok egyedi dobozokat színezni, szövegeket formázni, ezért a main.css-tartalmát most megkapjátok, hogy egy kis időt spóroljunk, és az új tananyagra tudjunk koncentrálni:

/\* Az alábbi formázás az első sor szövegére vonatkozik, amelyet a képernyő felolvasók tudnak értelmezni. \*/

.screen-reader-text {

clip: rect(1px, 1px, 1px, 1px);

position: absolute !important;

height: 1px;

width: 1px;

overflow: hidden;

}

.screen-reader-text:hover,.screen-reader-text:active,.screen-reader-text:focus {

background-color: #f1f1f1;

border-radius: 3px;

box-shadow: 0 0 2px 2px rgba(0, 0, 0, 0.6);

clip: auto !important;

color: #21759b;

display: block;

font-size: 14px;

font-weight: bold;

height: auto;

left: 5px;

line-height: normal;

padding: 15px 23px 14px;

text-decoration: none;

top: 5px;

width: auto;

z-index: 100000;

}

/\*--------------------------------------------------------------

Oldal formázása

--------------------------------------------------------------\*/

body {

margin: 0;

}

body,button,input,select,textarea {

font-family: 'Source Sans Pro', 'Helvetica', 'Arial', sans-serif;

font-size: 18px;

line-height: 1.5;

}

h1,h2,h3,h4,h5,h6 {

clear: both;

}

p {

margin-bottom: 1.5em;

}

b,

strong {

font-weight: bold;

}

dfn,cite,em,i {

font-style: italic;

}

blockquote {

margin: 0 1.5em;

}

address {

margin: 0 0 1.5em;

}

pre {

display: inline-block;

font-family: "Courier 10 Pitch", Courier, monospace;

color: #ddca7e;

background: #282c34;

padding: .5em .8em;

border-radius: .2em;

}

code,kbd,tt,var {

font: 15px Monaco, Consolas, "Andale Mono", "DejaVu Sans Mono", monospace;

}

abbr,acronym {

border-bottom: 1px dotted #666;

cursor: help;

}

mark,ins {

background: #fff9c0;

text-decoration: none;

}

sup,sub {

font-size: 75%;

height: 0;

line-height: 0;

position: relative;

vertical-align: baseline;

}

sup {

bottom: 1ex;

}

sub {

top: .5ex;

}

small {

font-size: 75%;

}

big {

font-size: 125%;

}

/\* Kisebb kijelzőre definiálva \*/

@media screen and (min-width: 700px) {

.two-column {

column-count: 2;

column-gap: 2.4em;

}

}

/\* Global buttons \*/

.content-button {

padding: .5em 1em;

border: 3px solid #B51C35;

/\*border-radius: 10px;\*/

font-weight: 600;

}

a.content-button {

color: black;

text-decoration: none;

}

a.content-button:focus,a.content-button:hover {

background: #B51C35;

color: #FFFCED;

}

/\*--------------------------------------------------------------

Általános stílusok

--------------------------------------------------------------\*/

.site {

margin: 1em;

}

.masthead,.page-title,.main-content,.sidebar,.footer-content {

margin-bottom: 1em;

padding: 1em;

color: white;

}

.masthead {

background-color: #b46ae3;

}

.page-title {

background-color: #51a7fa;

}

.main-content {

background-color: #70bf40;

}

.sidebar {

background-color: #f49018;

}

.footer-content {

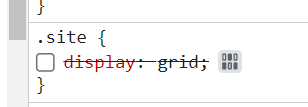
background-color: #0265c0;

}

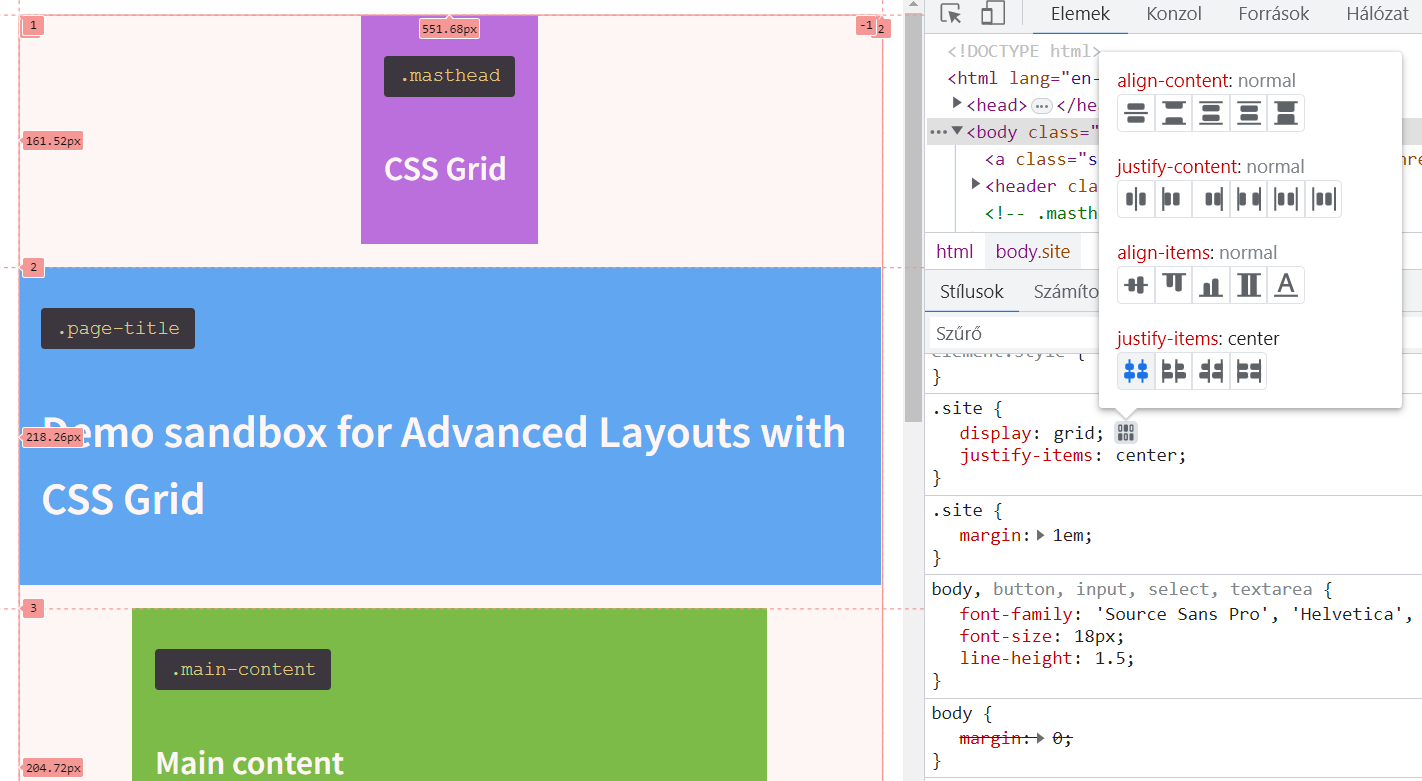
A grid.css-ben egyetlen szelektort választunk ki a body-hoz, azaz a dokumentum törzséhez kapcsolt site osztályt, melyet .site-al jelölünk. A tulajdonság display lesz az értéke pedig gird.

Ha most megnyitod a fejlesztői eszközöket, és kijelölöd a body-t látni fogsz mellette egy kis rács mintát, kattints rá, és meg fogja jeleníteni Neked az oldalon a rács nézetet.

És most, ha megnézed a rácsot, látni fogod, hogy van ott egy körvonal, ami maga a rács, és, hogy minden elem, az egy rács blokk. Valójában ezek egyes rácscellákban vannak elhelyezve. Csak éppen most ezek a rácscellák történetesen függőlegesen vannak elrendezve, és egymás után helyezkednek el. Tehát gyakorlatilag nincs különbség, a rács beállítása előtti állapot és a mostani között még. ☺

Most átkattinthatunk az Elrendezés lapra, a jobb oldalon, és látni fogod, hogy a Rács alatt van egy Overlay Grid a body.site számára. Jelenleg ez be van kapcsolva. Ki-be kapcsolhatod.

Ezen az eszközön belül pedig most választhatunk, pontosan mit szeretnénk vizualizálva látni. Így végtelenségig kiterjesztheted a rácsvonalakat. Itt láthatod, hogy a rácsvonalak túlmutatnak magán a rácstárolón.

Itt már láthatod azokat a rácsvonalakat, amelyekről beszéltem. Megjelenítheti a sorszámokat, itt van az első sor, a második, a harmadik és így tovább. Jelenleg nincsenek még oszlopaink, ezért csak az első és a második sor van az oszlopokhoz. És végül, meg tudjuk jeleníteni a rácsterületeket, de még nincsenek rácsterületeink, így nincs értelme ezt bekapcsolni. Ez az eszköz azt is lehetővé teszi, hogy lássuk, mely rácsterületekre fókuszálunk jelenleg úgy, hogy egyszerűen lebegtetjük a kijelzőt.

Innentől kezdve a gyakorlatokhoz ugyanezt a mappát fogjuk használni, csak a grid.css-ben fogunk mindig változásokat végezni.

# Rácsvonalak és egységek

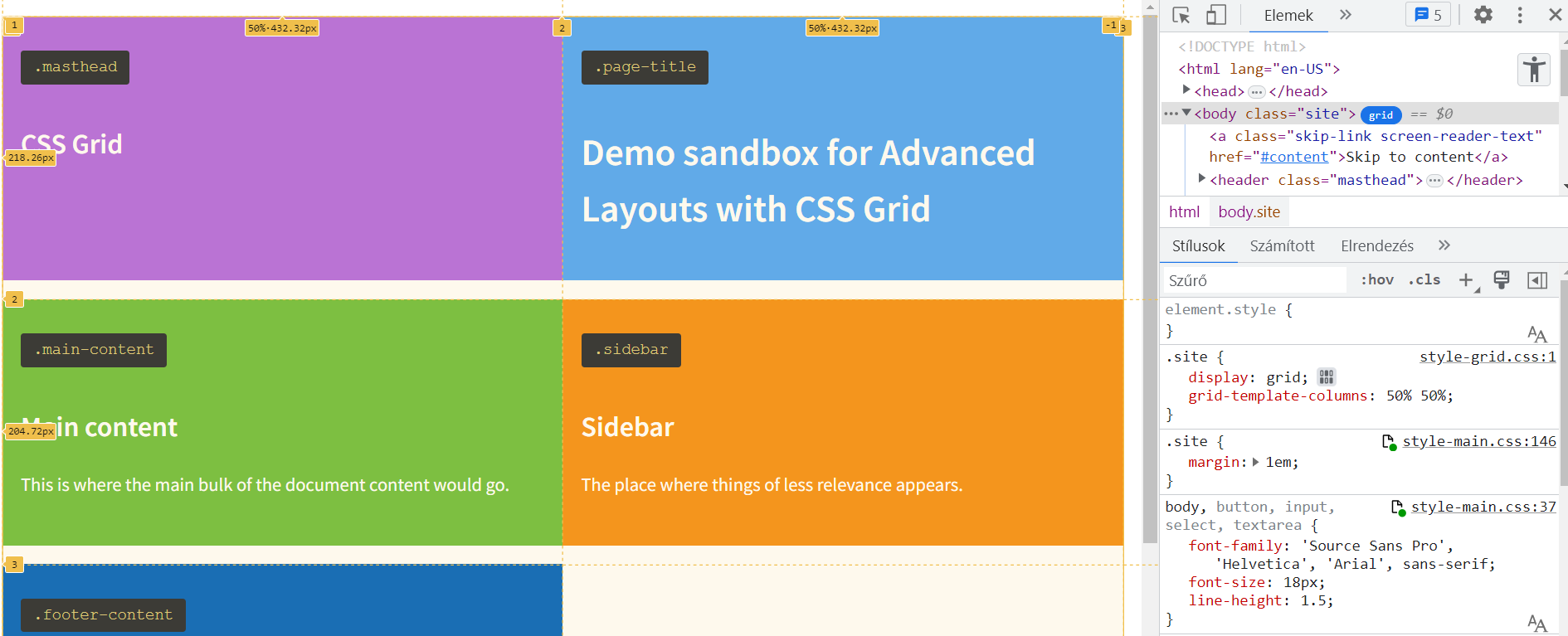
A grid konténer definiálásával megkezdhetjük a rácsvonalak deklarálását oszlopok és sorok létrehozásához. Ez a rácssablon oszlopainak és a rácssablon sorainak tulajdonságai segítségével történik.

Ezek a tulajdonságok mindegyike szóközökkel elválasztott értéklistákat vesz fel, amelyek beállítják az oszlop vagy a sor szélességét vagy magasságát az egyik sor és a következő sor közötti tér szélességének vagy magasságának deklarálásával.

A böngészőben jelenleg a rácsnak van egy oszlopa, amely átfedi a rács teljes szélességét, az oszloprács első sora a bal széle, a második oszloprácsvonal pedig a jobb széle.

A .site osztályhoz adjunk hozzá egy új tulajdonságot: grid-template-columns: 50% 50%

Mentsed el, és a böngészőben látni fogod, hogy létre jött egy további rácsvonal. Tehát van egy rácsvonal, a bal él, majd a második rácsvonal, ami most középen van, és a harmadik rácsvonal a jobb szélen.

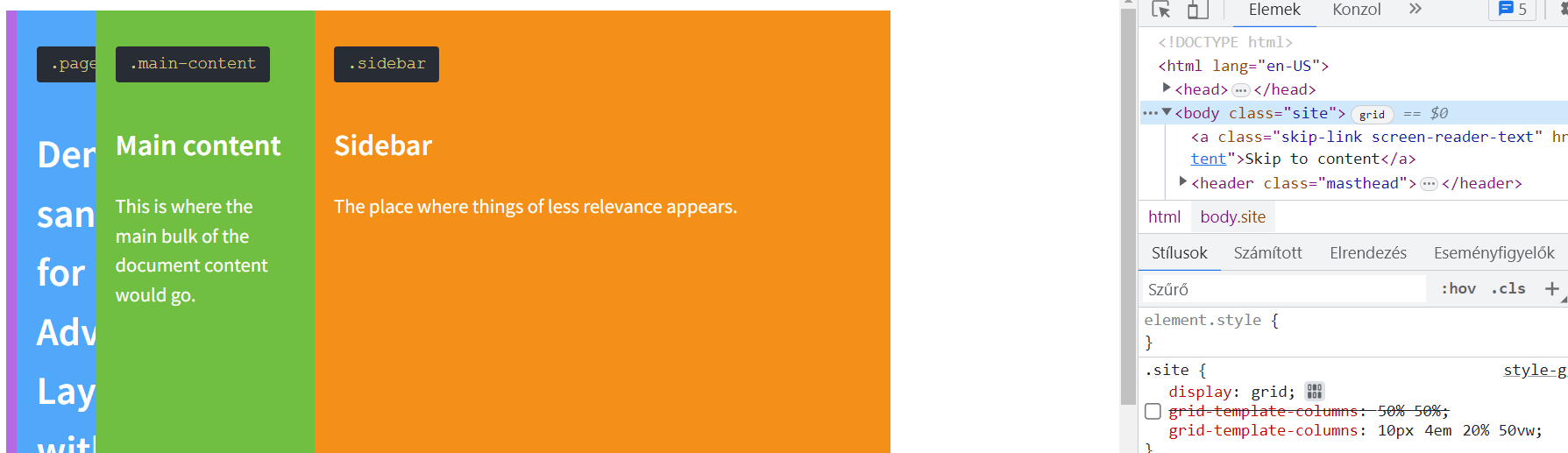


*7. számú kép*

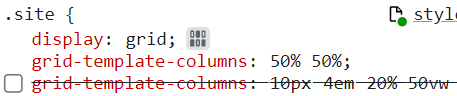
Ez a tulajdonság amit hozzá adtunk azt mondja, hogy keresse meg az első rácsvonalat, tehát a bal szélét, majd helyezzen egy második rácsvonalat a rácstartály teljes szélességének 50%-ára, és helyezze az utolsó rácsvonalat a maradék 50% után. Ennek eredményeként két egyenlő szélességű oszlopot kapunk a rácsunkon belül. A létrehozott sorok vagy oszlopok alapértelmezett szélessége automatikus, vagyis olyan széles, amennyire a tartalomnak szüksége van, a rács rendelkezésre álló területén belül.

Ha azt szeretnéd, hogy egy oszlop vagy sor a benne lévő tartalomtól függően bővüljön vagy szűküljön, az oszlopszélességeket és a sormagasságokat is megadhatod, méghozzá a mértékegységek tetszőleges keverékével. pl:

grid-template-columns: 10px 4em 20% 50vw



Így a fenti példából láthatod, hogy ugyanabban a deklarációban keverheted a pixelszélességeket és az em-eket és a százalékokat , a nézetablak szélességét és magasságát és minden mást . Tehát ha egy nagyon furcsa rácsot akarunk készíteni, akkor mondhatok 10 pixelt és 4 EM-et és 20%-ot és 50 VW-t.

Ki is tudod a fejlesztőben kapcsolni a tulajdonságokat és úgy tesztelni mi történik:

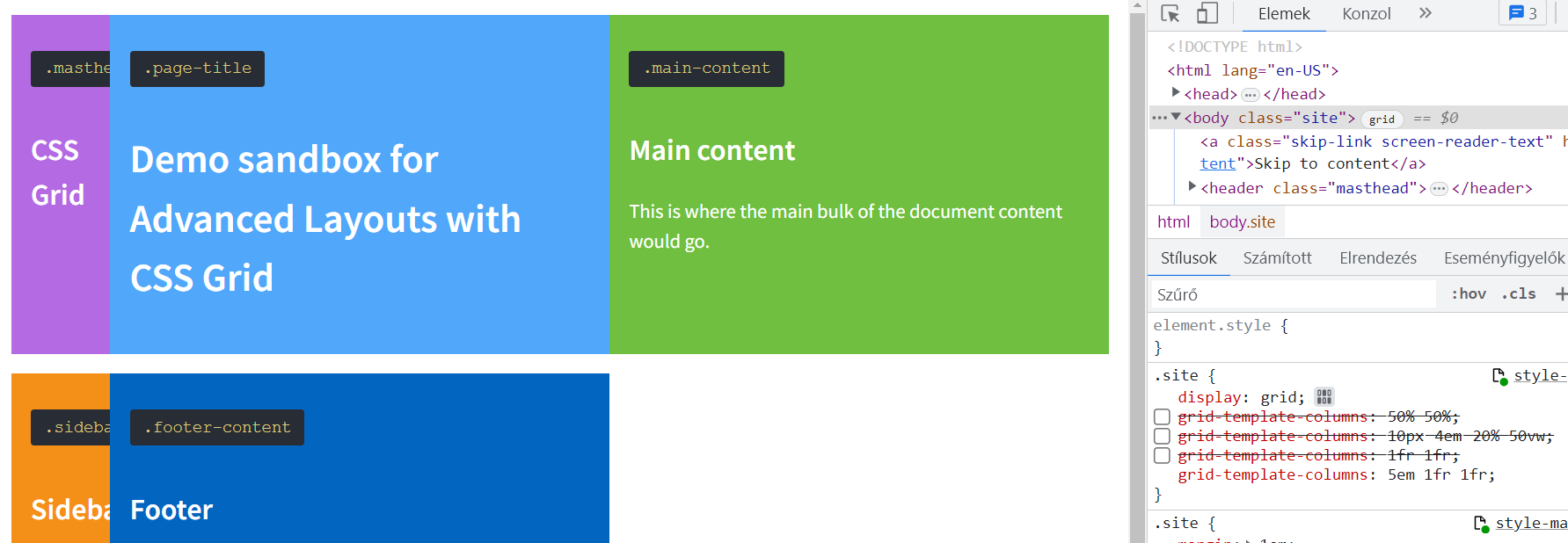
A CSS-GRID két további mérési egységgel és eszközzel rendelkezik, amelyeket használhatsz. A tört mértékegysége vagyis **Fraction (FR) Unit** és a **minmax** függvény. Az FR-ként megjelölt törtegység használata meglehetősen egyszerű. Példánkban ahelyett, hogy azt mondanánk, hogy a terület 50%-át akarjuk, csak felosztjuk a rácsot két egyenlő fraction között:

grid-template-columns:1fr 1fr;

Ha megnézed a böngészőben ugyanazt az eredményt fogjuk kapni, mint amikor kétszer 50%-ra osztottuk fel a gridet. (*7.számú kép*)

Ha ezután egy másik oszlopot is létre akarok hozni fix szélességgel, amely például 5 em szélességű, akkor a fennmaradó hely továbbra is egyenletesen oszlik el a két FR egység között.

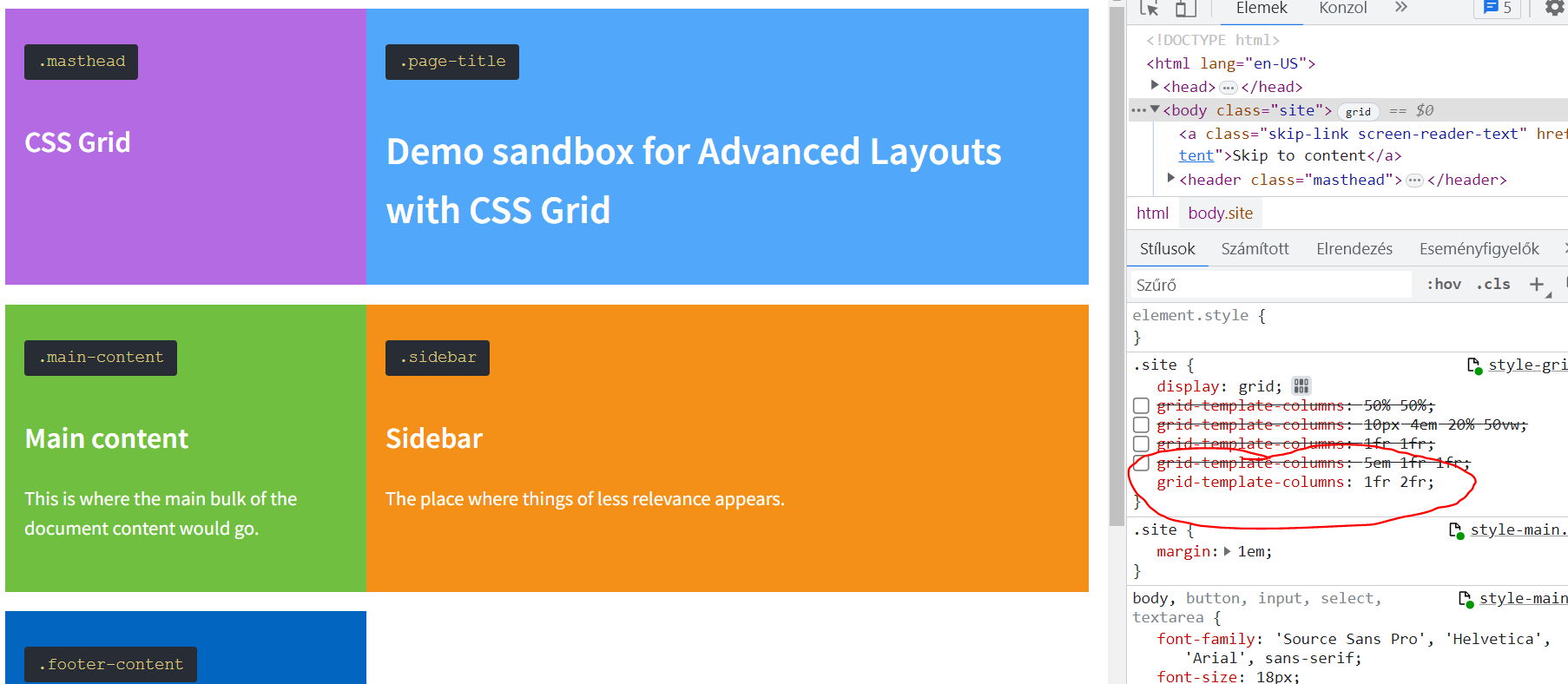
grid-template-columns: 5em 1fr 1fr;



Az FR egység lehetővé teszi, hogy törtekkel dolgozzunk. Tehát jelenleg, ha van egy FR-em, egy FR-em, akkor a két oszlop egyenlő szélességű.

De ha beállítok 1 FR-t és 2 FR-t, akkor három részre osztja:

grid-template-columns: 1fr 2fr;

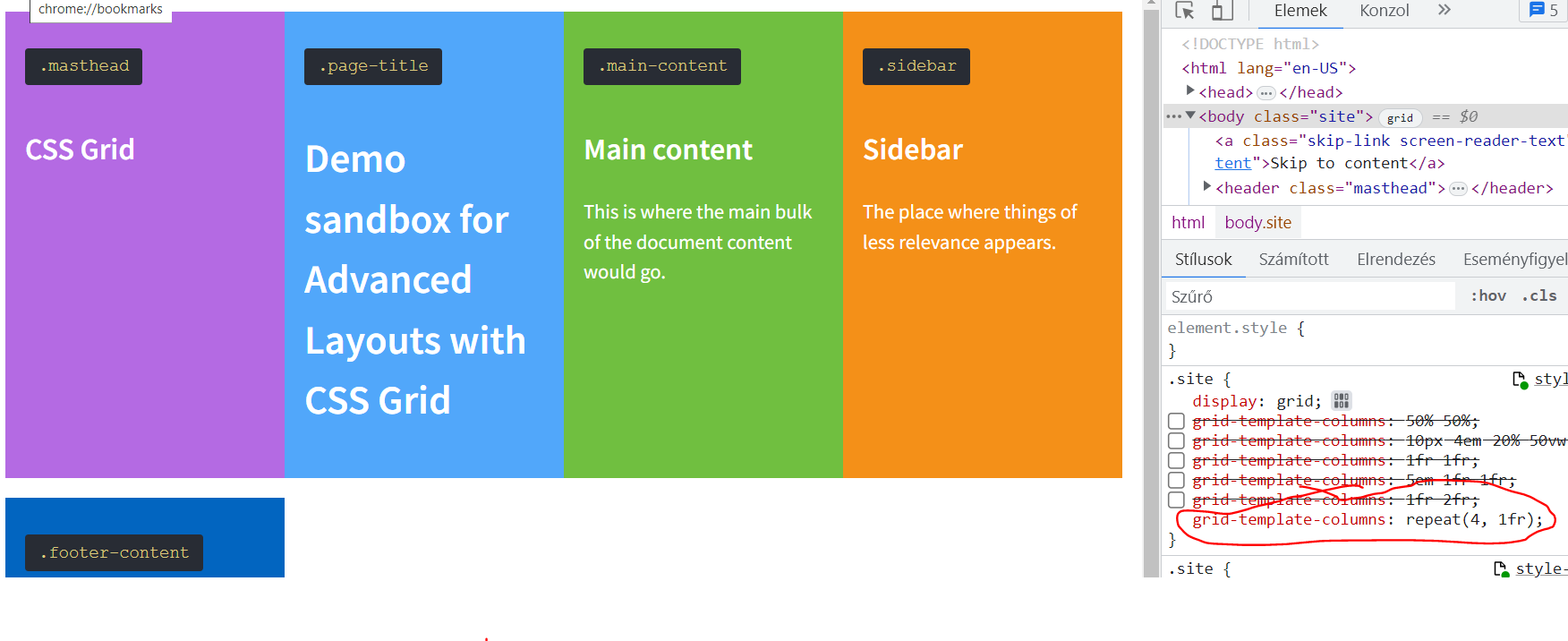


Tehát itt a végül valami pontosabbat kapunk, mintha 33,3333333%-ot próbálnánk értékeknek megadni, majd ezt háromszor megtennénk.

A **minmax() function** ettől már valamivel fejlettebb. Lehetővé teszi egy oszlop vagy sor minimális és maximális szélességének deklarálását. Ez különösen hasznos és érzékeny a tervezésre, mert meghatározhatjuk, hogy az adott oszlopok soha nem lehetnek keskenyebbek 20 em-nél, és soha nem lehetnek szélesebbek 30 em-nél. Ez hasonló a minimális szélességhez és a maximális szélességhez, de az oszlopok és sorok kontextusában. Ez azért fontos számunkra, mert, lehetővé teszi számunkra, hogy valóban reszponzív elrendezéseket hozzunk létre CSS-rács segítségével.

Ha több hasonló oszlopra van szükséged, akkor van egy rövidítés az repeat() jelölés használatával. Tegyük fel, hogy négy oszlopot szeretnél 1 FR-ből. Ahelyett, hogy beírnád 1 FR, 1 FR, 1 FR, 1 FR, egyszerűen így jelöljük:

grid-template-columns: repeat(4, 1fr);



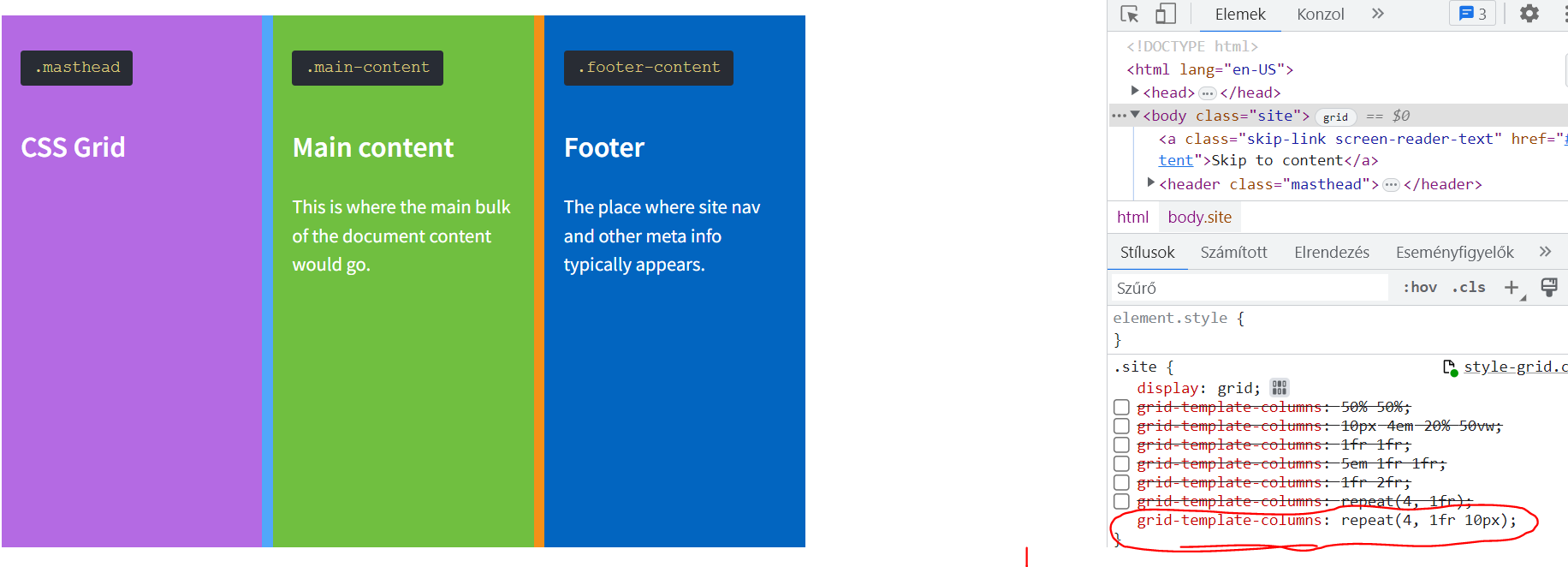
Fejlettebb kombinációkat is használhatunk pl:

grid-template-columns: repeat(4, 1fr 10px);

Fontos a szintaktika, hová teszel vesszőt és hová nem!

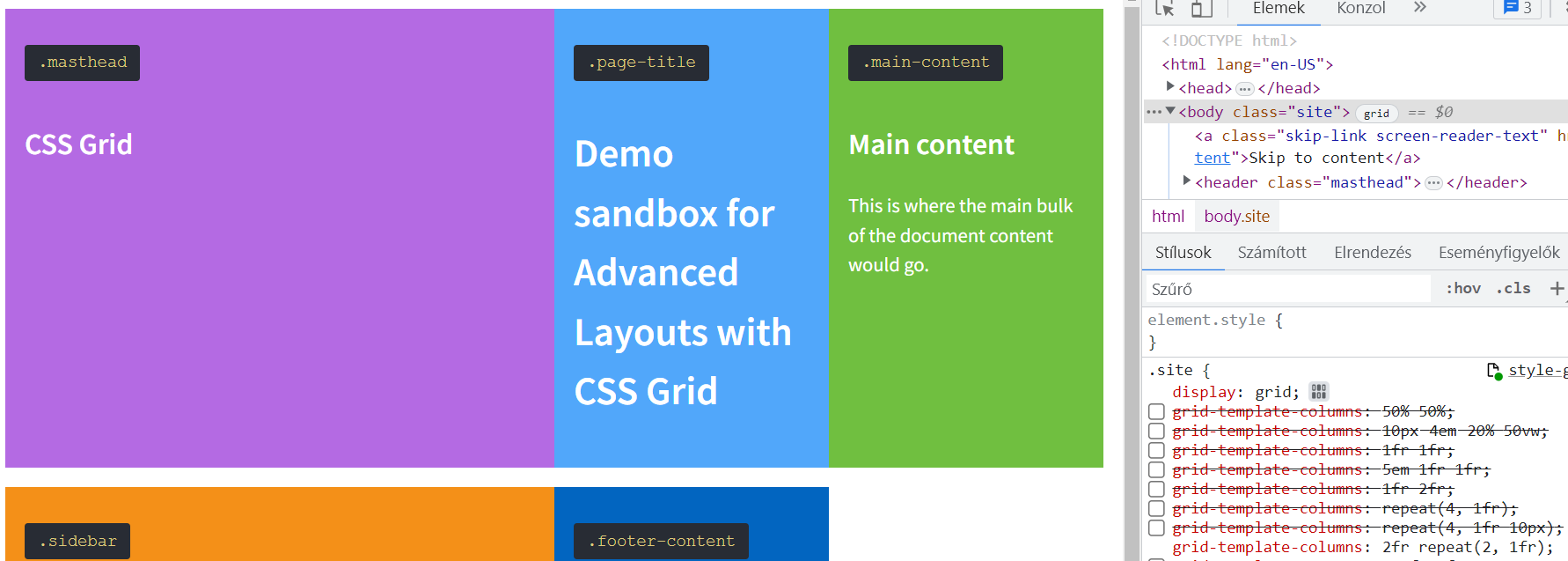
Tehát ismételje meg a négyet, vesszővel elválasztva, 1 FR 10 pixel és figyelj arra ezek közé már nem kell vessző!!!

A végeredmény valójában nyolc oszlopot ad, váltakozva egy FR és 10 pixel szélességben.



Megadhatjuk vegyesen is, fixen egy rácsot és automatikusan a többit:

grid-template-columns: 2fr repeat(2, 1fr);



Amint látod létre hoz 4 rácselemet, az első 2fr, a második és harmadik ismétlődik kétszer az 1fr.

# Automatikus rácselem elhelyezés

Hozzuk létre a következő rácspontokat:

grid-template-columns: 2fr repeat(2, 1fr);

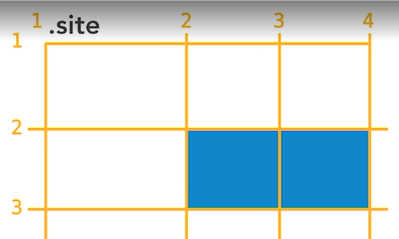
grid-template-rows: auto 1fr 3fr;

Tehát az új rácsomban három oszlopom van, 1db 2fr, 2db 1fr, és van 3 sorom is.

Az első sor automatikus magasságra van állítva, ami azt jelenti, hogy a tartalom megjelenítéséhez szükséges magasságú lesz. A második sor 1fr-nál, a harmadik sor pedig 3fr-nál van, és jelenleg üres, mert ezek az elemek automatikusan behelyezik magukat a rácsba.

Ez a rács automatikus elhelyezése egy funkció, és rendkívül hasznos, ha nagy tartalomcsoportokkal kezd valaki dolgozni, és csak azt szeretné, hogy azok automatikusan kitöltsenek egy meghatározott rácsot.

# Kézi rácselem elhelyezés

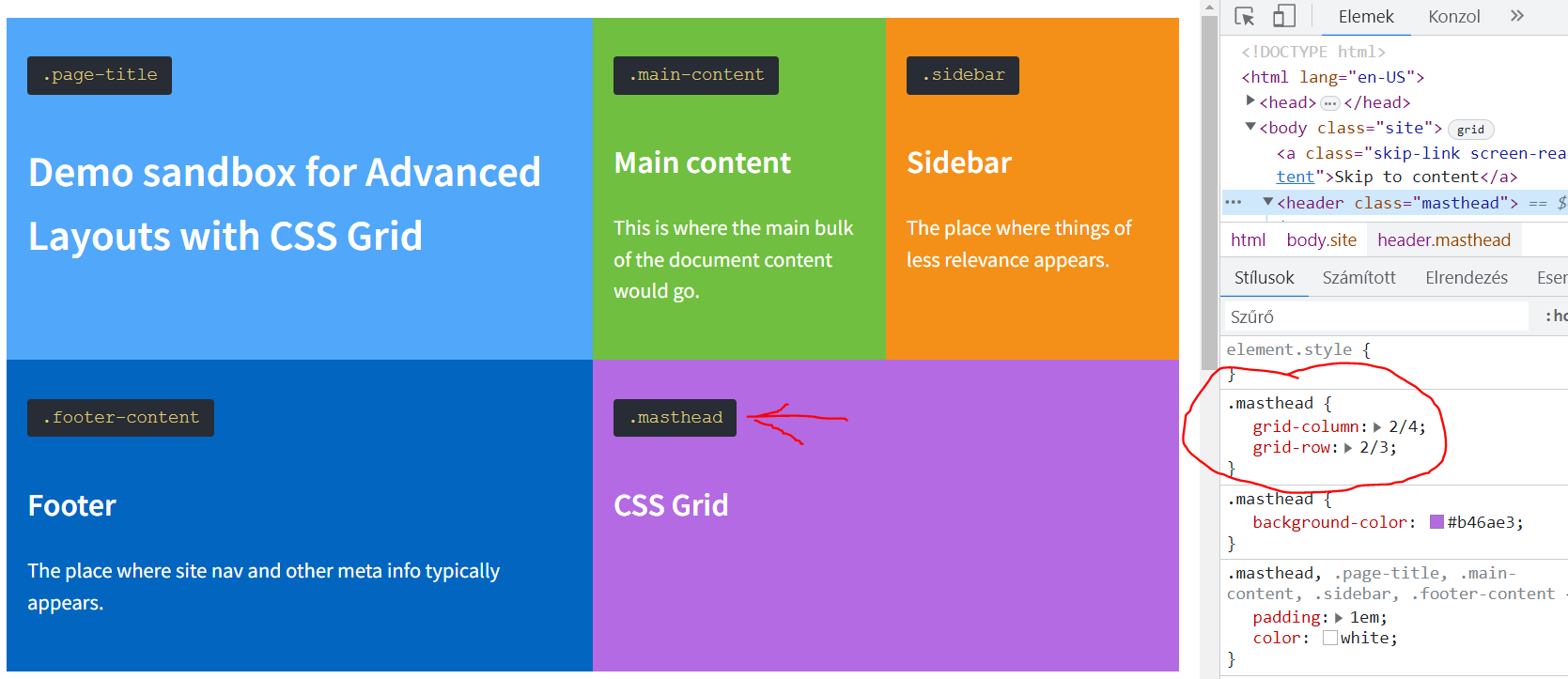
Ami a CSS Grid-et olyan hatékony elrendezési eszközzé teszi, az az, hogy lehetővé teszi számunkra, hogy meghatározzuk, hol jelenjen meg a rács bármely eleme. Más szavakkal, a CSS Grid használata lehetővé teszi, hogy tetszőleges elemet bárhová elhelyezzünk a rácsban, csak a CSS használatával.

A rácselemek rácsra helyezésének legalapvetőbb módja az, hogy minden elemhez deklaráljuk a rácsoszlop és a rácssor értékekét. A rács-oszlop- grid-column és a rácssor- grid-row a rácson azaz a griden lévő soroknak megfelelő értékeket vesz fel, tehát ha egy elemet a második és a negyedik oszlopsor közé szeretne helyezni , akkor :

grid-column:2/4;

grid-row:2/3;

Ezzel a technikával minden rácselemet bármilyen elrendezésben elhelyezhetsz a rácsban a megfelelő helyre.



Ha a jelenlegi példát vesszük, ahol az automatikusan elhelyezett tartalmak vannak, akkor ezeket az elemeket áthelyezhetjük a kívánt elrendezés létrehozásához. A conténert hagyjuk változatlanul:

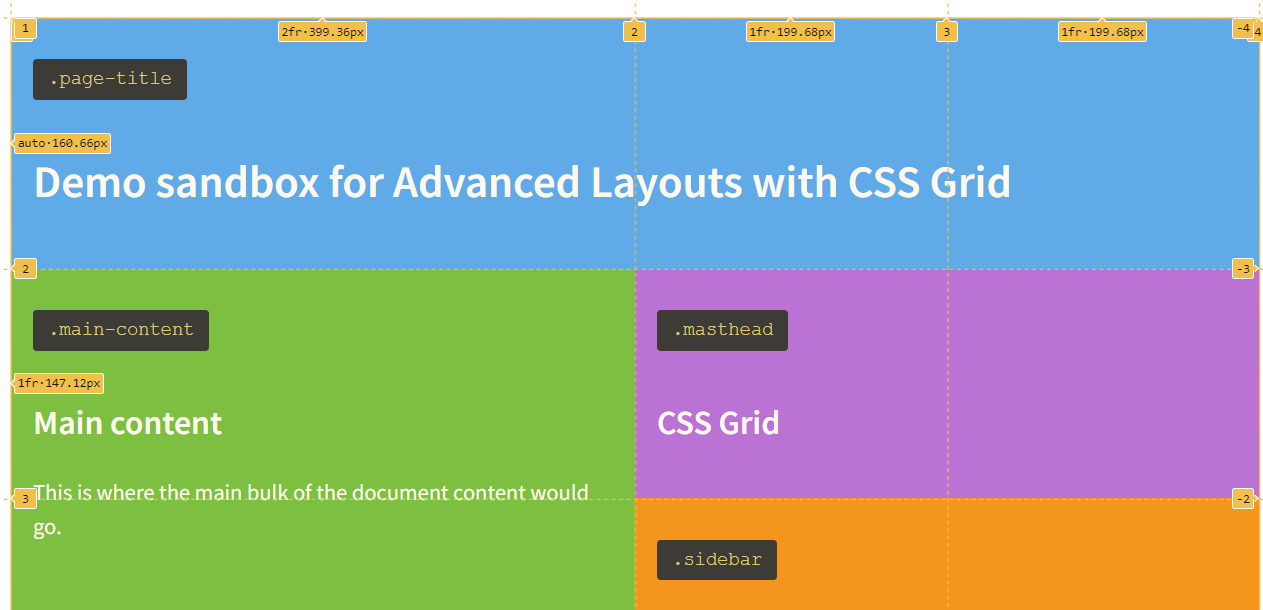
.site {

display: grid;

grid-template-columns: 2fr 1fr 1fr;

grid-template-rows: auto 1fr 3fr;

}



.masthead {

grid-column: 2/4;

grid-row: 2/3;

}

Tehát ha a fenti képen lévő mintát szeretnénk követni, akkor figyeld meg a grid által létrehozott számokat, a rács oszlop esetében megkeressük a 2-ik oszlop kezdetét, és azt mondom a 4-ig tartson, a sorok esetében, megkeressük a 2ik sort, és azt mondjuk a 3ik-ig tartson.

Elemezzük a többi blokk elhelyezését is:

.page-title {

grid-column: 1/4;

grid-row: 1/2;

}

Ha egy rácselemet elmozdítunk az útból, a többi rácselem odafolyik, hogy elfoglalja az új, rendelkezésre álló helyet. Így az oldal címe felmegy az első pozícióba , és minden más befolyik, és kitölti számomra a rácsot.

.main-content {

grid-column: 1/2;

grid-row: 2/4;

}

Ha most megnézzük az elrendezést mielőtt tovább mennénk, láthatod a láblécben ez a felső rész több helyet foglal el, mint egy fr az aktuális elrendezésben. Ennek eredményeként a rács automatikusan megnöveli ezt az oszlopot, hogy illeszkedjen ehhez a minimális szélességű tartalomhoz. Ezt mindig szem előtt kell tartani, amikor CSS Grid elrendezésekkel dolgozunk, de ha nagyon hosszú szavakat, esetleg meghatározott szélességű képeket írunk be, vagy valami hasonlót, akkor a rács oszlopait szélesebbre vagy magasabbra állíthatjuk be, mint amilyennek eredetileg kellett volna. Persze ki lehet törölni a túl hosszú szót, vagy el lehet rejteni a kilógást, le lehet vágni, de erre a legjobb megoldás a megfelelő reszponzív rácsok létrehozása. (ha összetoljuk az oldalt akkor látszik)



.sidebar {

grid-column: 2/4;

grid-row: 3/4;

}

.footer-content {

grid-column: 1/4;

grid-row: 4;

}

Így a mi forgatókönyvünkben valószínűleg megragadnánk a láblécet , és lejjebb helyeznénk a tartalom alá. De van itt egy kis probléma. Ha visszamegyünk a fejlesztői eszközökhöz és nézd meg a rácsot, látni fogod, hogy itt három sorunk van. Egy, kettő és három, és jelenleg a lábléc az alsó sorban található. Tehát hogyan mozgathatom le a láblécet egy olyan sorba, amely nincs ott? Nos, itt jutunk el valami úgynevezett implicit vonalhoz. Ha egy elemet a meglévő sorokon vagy oszlopokon kívül helyezünk el a rácsban , a böngésző új sorokat generál, akár sorokat, akár oszlopsorokat, hogy a rács szerkezetileg szilárd maradjon. És ez azt jelenti, hogy ha akarjuk, egyszerűen elhelyezhetünk dolgokat a rácson kívül, és a böngésző azt mondja: "Rendben. "Nagyobb rácsot akarsz? "Rendben van. "Csak hozzáadom a további sorokat, amelyek szükségesek ahhoz, hogy ez működjön."



# Sorok elnevezése

A rácsvonal számok használata az elemek rácsra helyezésére jól működik, de ez egy nagyon elvont módja az elrendezések megközelítésének, ha jobban belegondolunk. Ha ezt az elrendezést nézzük itt fent, nem az első, a második, a harmadik és a negyedik sorra gondolunk. Valószínűleg valami olyasmire inkább, hogy ez a fejléc, aztán a fő terület, és akkor itt van egy oldalsáv, és van egy lábléc az alján.

Ha valakinek ez logikusabb és ezért ezt a fajta terminológiát szeretné használni a CSS létrehozásakor, a jó hír az, hogy megteheti. Ennek két különböző megközelítése van.

Az első megközelítés a vonalak elnevezése és itt több dolgot is megtehetsz. Ha elképzeled, láthatod, hogy van egy vonal itt, a 2 fr és van egy sor 2 fr után, van egy sor az 1 fr után, majd itt van egy sor a végén. Az egyes sorok neve szögletes zárójelbe kerül, és itt tetszőleges nevet adhatsz a soroknak, ha szóköz helyett kötőjelet használsz. Tehát ez az első sor itt a start-edge (kezdő él) , akkor megvan a felezőpont, majd megvan a negyedpont és akkor megvan a végél.

grid-template-columns:[start-edge] 2fr [half-way] 1fr [negyed] 1fr [end-edge];

Most, hogy a soroknak van neve, használhatjuk ezeket a neveket a számok helyett.

.masthead {

grid-column: half-way/end-edge;

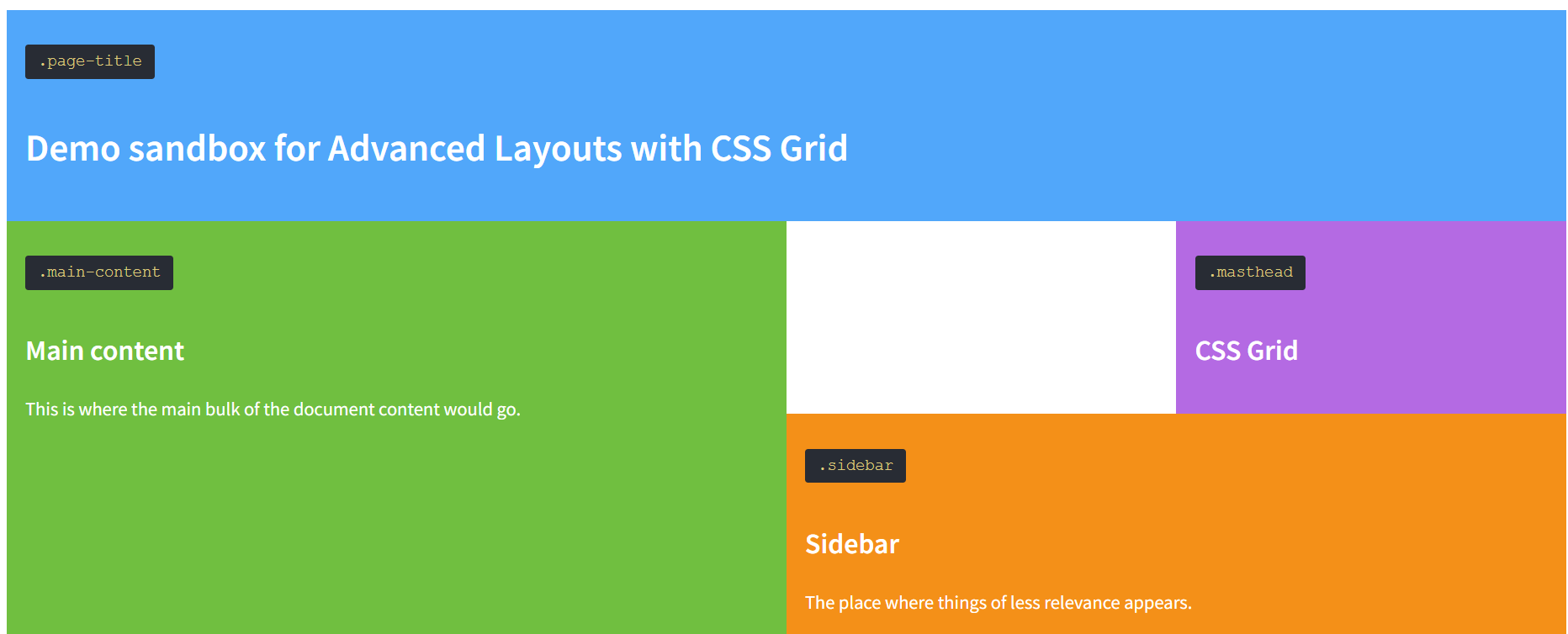
grid-row: 2/3;

}

Ha megnézed a böngészőbe az eredményt, láthatod, hogy ugyanazt kapod.

de most javítsd át így:

grid-column: negyed/end-edge;



Üres marad a kitöltetlen terület.

Akinek ez a módszer sem jön be, létezik egy beépített rövid kézi elnevezési konvenció az elnevezett sorokhoz, amellyel leegyszerűsíthetjük ezt az elnevezéses dolgot. Ahhoz, hogy lássuk, hogyan működik, még egyszer át kell néznünk az elrendezésünket. Nevezzük tehát ezt a fő szakaszt itt main-nek, ezt a jobb szélső oszlopot pedig oldalsávnak.

.site {

display: grid;

grid-template-columns:[main-start] 2fr [main-end] 1fr [sidebar-start] 1fr [sidebar-end];

grid-template-rows: auto 1fr 3fr;

}

.masthead {

grid-column: main-end/sidebar-end;

grid-row: 2/3;

}

.page-title {

grid-column: main-start/sidebar-end;

grid-row: 1/2;

}

.main-content {

grid-column: main-start/main-end;

Ha most megnézed, az eredményt ismét láthatod, hogy az elrendezés még mindig ugyanaz.

# Grid Template Area

A rácstárolón lévő rácssablon terület tulajdonságának használatával a rácsterületek lehetővé teszik számunkra, hogy a rácson belül egy vagy több cellát átívelő téglalap alakú területeket definiáljunk, és elnevezzük őket. Ezután ezeket a neveket felhasználhatjuk elemek elhelyezésére a rácson belül, a rácselemek rácsterület tulajdonságának használatával. Más szavakkal, a rácsot térképpé alakítjuk, és elemeket helyezünk el a térképen, ami kinyitja az ajtót mindenféle klassz dolog előtt.

Hozzáadhatunk rácsterületeket a rácsunkhoz, majd a rácsterületek alapján elemeket helyezhetünk el. Ehhez a webhelyszabályomhoz hozzáadok egy új tulajdonságot, a grid-template-areas-t. Ez a tulajdonság nem hasonlít más CSS-tulajdonságokhoz. Ahelyett, hogy hozzáadnánk egy beállítást vagy valamilyen értéket, valójában egy térképet rajzolunk rácsunkról, szöveget használva az egyes cellák helyett. Tehát ehhez vissza kell térnem a rácsomhoz, meg kell néznem, és pontosan ki kell találnom, mit akarok először csinálni.

.site {

display: grid;

grid-template-columns: 1fr 1fr;

grid-template-rows: auto 1fr 3fr;

**grid-template-areas:**

**"title title"**

**"main masthead"**

**"main sidebar"**

**"footer footer";**

}

Ezzel tulajdonképpen felvázolom melyik részét hová szeretném elhelyezni a rácson, és a blokkokat már csak el kell nevezni a megfelelő néven:

.masthead {

grid-area: masthead;

}

.page-title {

grid-area: title;

}

.main-content {

grid-area: main;

}

.sidebar {

grid-area: sidebar;

}

.footer-content {

grid-area: footer;

}

Amikor adaptív elrendezést hozol létre, át kell gondolnod, hogy hol van minden egyes elem, és ehhez át kell írni az összes CSS-t. A grid-template-area segítségével mindössze annyit kell tenni, hogy meghatározzuk a rácsot, majd beállítjuk a grid-tempalate-area-t, majd létrehozunk egy új médialekérdezést, és módosítjuk a grid-template-area-t, és a blokkok oda fognak ugrani, ahol szükség van rájuk. . Tehát ez az, amely minden 600 képpont alatti böngésző nézetre fog vonatkozni.

@media screen and (min-width: 600px) {

.site {

grid-template-columns: 2fr 1fr 1fr;

grid-template-areas:

"title title title"

"main masthead masthead"

"main sidebar footer";

}

}

Tegyük fel, hogy a fejléc és az oldalsáv helyet akarok cserélni. Nos, nem kell mást tennem, mint megnézni, hová hivatkozik az oldalsáv és kicserélni őket.

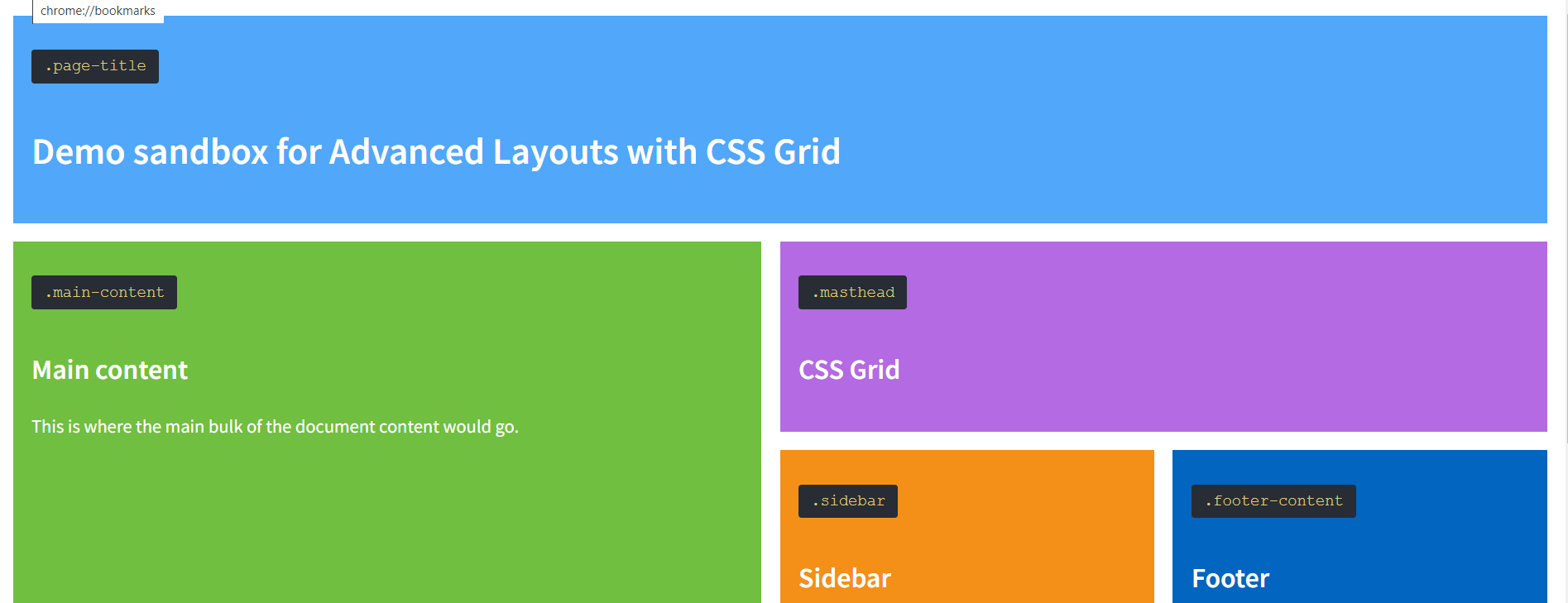
# Grid gap – rács rés

Ha megnézzük a jelenlegi rácsot, akkor látható, hogy teljesen egymásra tapadnak a rácsban lévő blokkok. Ez így most jól is néz ki, de ha az elrendezés úgy kívánja van amikor szükség van egy kis térre az egyes elemek között.

Most persze meg lehet oldani ezt úgy, hogy bemegyünk egy egyedi elembe , majd hozzáadunk hozzá némi margót úgy, hogy margin: 1em vagy valami hasonló. De akkor csak annyit tesz, hogy teret hoz létre az egyes blokkok körül és nem foglalkozik a rács eredeti elrendezésével.

Sokkal tisztább módja lenne, ha tényleges teret hoznánk létre a rácsvonalak között. A CSS Grid terminológiájában az egyes oszlopok vagy sorok közötti teret grid-gap-nek nevezik, és ezt egyszerűen úgy kell beállítani, hogy a grid-gap.

Tehát próbáljunk meg pl a .site-ra egy grid-gap:1em-et beállítani, és nézzük meg mi történik.



Látható, hogy vízszintesen és függőlegesen is rés keletkezett az egyes cellák között. Abban az esetben, ha egy elem több különböző cellán átível, nem látjuk ezt a rést. Tehát csak a cellák közötti tényleges térre vonatkozik, nem területekre, semmi másra nem.

Visszatérve az elrendezési panelre ,próbáld ki mi történik, ha kiegészíted grid-column-gap: 1em , -re. majd grid-row-gap –re. Egyik csak az oszlopok, másik csak a sorok közé szúr be rést.

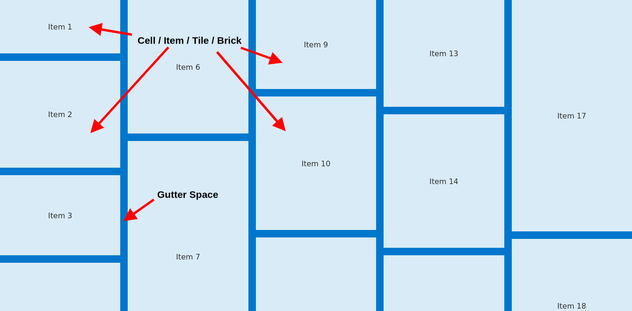
Így pontosan szabályozhatod, hogy mennyi hely legyen az egyes rácssorok vagy rácsoszlopok között.

Fontos dolgot azonban itt meg kell jegyezni, a rácsrés, amint látható, csak a rácson belül érvényesül, nem terjed ki a rács szélei köré. Tehát, ha rácsrést akarunk létrehozni, majd a rácsrés megjelenését a rácson kívülre is el akarod helyezni, akkor valamilyen margót vagy padding-et kell alkalmazni a rácstartály körül is.

# CSS GRID legfontosabb tulajdonságainak áttekintése

A CSS Grid használata egy teljesen új megközelítést vezet be a webelrendezések és webdizájnok terén. Viszont ez a technológia megkívánja, hogy alaposan átgondoljuk a különböző tartalom elemek közötti kapcsolatokat, és elrendezéseket mind a HTML szemantikai, mind pedig az elrendezés és a tervezés szempontjából.

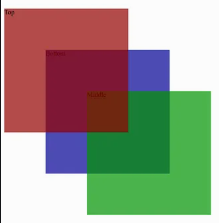
1. Csak a rácstároló közvetlen első szintű leszármazottai számítanak rácselemeknek. Ez azt jelenti, hogy ha vannak olyan leszármazottai, amelyeket el szeretnél helyezni a rácson, először létre kell hoznia egy új beágyazott rácsot. Lehet egy rács az általános elrendezéshez, egy másik a fő tartalomhoz, és egy konkrétabb az egyes tartalomblokkokhoz
2. A CSS Grid valódi rácsot hoz létre. Ez alatt azt értjük, hogy nem szigorú sorokkal és oszlopokkal dolgozik. Az egyik legelső dolog, amit az webfejlesztők megpróbáltak, - amikor a CSS Grid megjelent - , az volt, hogy falazó stílusú elrendezéseket hozzanak létre az egyes elemek magasságához, ahogyan azt azok tartalma határozza meg, és ezzel egy organikus, félig véletlenszerű szerkezetet kapjanak.



*Masonry style*

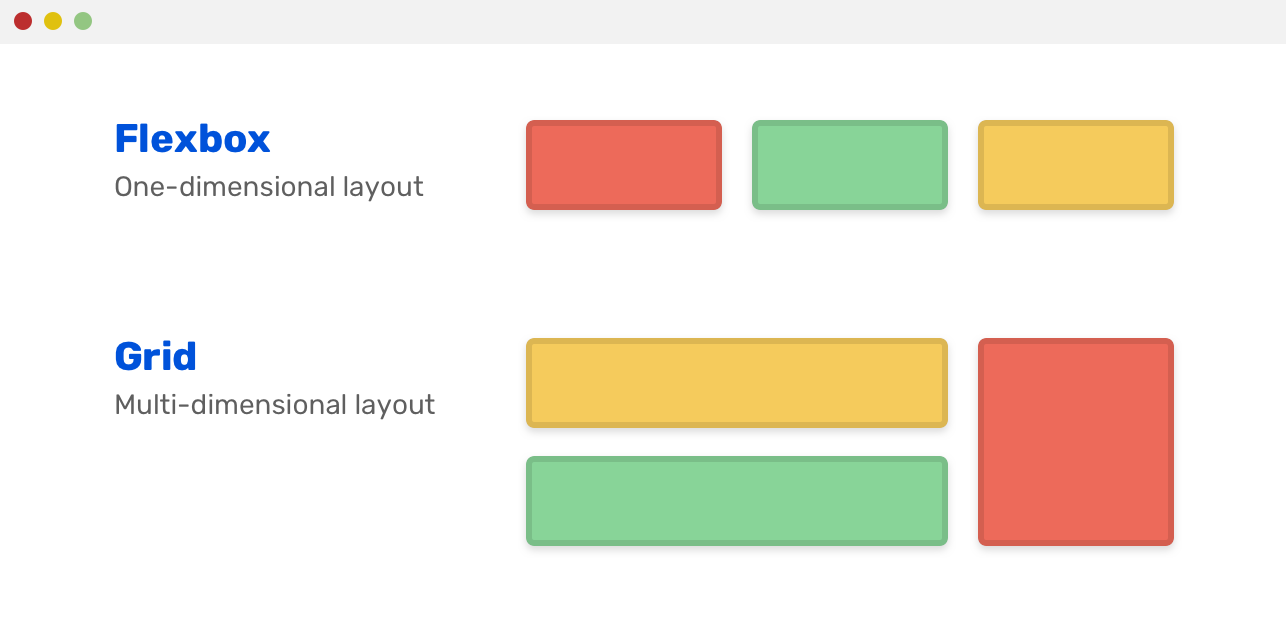
Ez nem egy rács, sőt, ez egy rács ellentéte. A rács célja, hogy egyértelműen meghatározott rácssávokat hozzon létre oszlopok és sorok formájában, és ezeken belül helyezze el a tartalmat. Ez azt jelenti, hogy alaposan át kell gondolni, hogy egy cella tartalmának hossza hogyan befolyásolja az egész sor magasságát.

Ez némi gyakorlást igényel, és azt fogod tapasztalni, hogy sok esetben úgy tűnik, hogy a rácssávok akadályozzák azt, amit meg akarsz valósítani, de ilyenkor több beágyazott rácsra van szükség.

1. A tartalom halmozása. Bármely rácselem elhelyezhető a rács bármely cellájába vagy cellacsoportjába, a böngésző lehetővé teszi több elem egymásra helyezését ugyanabban a cellában. Az elemek a HTML-forrás sorrendjében jelennek meg, ami azt jelenti, hogy minél lejjebb van egy elem a HTML-dokumentumban, annál feljebb jelenik meg a veremben. Ezután módosíthatod az egyes elemek z-index pozícióját, hogy előbbre hozd őket vagy elrejtsd őket, vagy becsúsztasd félig meddig a másik elem alá.

Ez a funkció rendkívül megkönnyíti az olyan dolgokat, mint például a szöveg elhelyezése a képek tetején, és megszünteti az általunk eddig használt összetett abszolút pozicionálást és egyéb durva megoldásokat. Hátránya viszont, hogy ha nem figyelsz oda nagyon arra, hogy hol helyezed el az elemeket, könnyen lefedhet valami fontosat, ezért ellenőrizd folyamatosan a böngészőben a megjelenést.

1. A tartalom rendezése. Köszönhetően annak, hogy bármely rácselemet bárhol elhelyezhetsz a rácson belül, könnyedén módosíthatod a tartalom sorrendjét a HTML jelölési struktúra megváltoztatása nélkül. De ugyanakkor fontos, hogy látogatónak ugyanazt a tartalmat ugyanabban a strukturált sorrendben kell megkapnia, akár a CSS-rács elrendezését nézi , akár csak felülről lefelé olvassa a HTML-t.
2. A CSS Grid tiszta CSS. Miután megfelelően megjelölted a HTML-kódot, nagyjából bármit megtehetsz az elrendezésével, mivel a rács pusztán vizuális. Bármikor be- és kikapcsolhatod a rácsokat, illetve módosíthatod az oszlopokat, sorokat, rácsterületeiket vagy bármi mást, a media query-k vagy a javascript segítségével. Ez által a CSS Grid egy igazán dinamikus tervezési felületet biztosít.



Míg a flexbox egy dimenziós elrendezést biztosított, a grid 2 dimenziós lehetőséget ad.

# Különbség a grid és az inline grid között

A dislplay:grid : blokkszintű tárolót és blokkdobozokat hoz létre a gyermekei számára.

A display: inline-grid:

Önmagukban, display: grid és display: inline-grid nem rendezik automatikusan sorokba és oszlopokba a dobozokat. Azt is meg kell mondanunk a böngészőnek, hogy hol és hogyan helyezze el a dolgokat.

|  |  |
| --- | --- |
| Grid | Inline-grid |
|  |  |