ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ, ՄՇԱԿՈՒՅԹԻ ԵՎ ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ /ՀԻՄՆԱԴՐԱՄ/

ՏԵՂԵԿԱՏՎԱԿԱՆ ԵՎ ՀԱՂՈՐԴԱԿՑԱԿԱՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

ԱԼԳՈՐԻԹՄԱԿԱՆ ԼԵԶՈՒՆԵՐԻ ԵՆ ԾՐԱԳՐԱՎՈՐՄԱՆ ԱՄԲԻՈՆ

Grid տեխնոլոգիան ոճերի կասկադավորման աղյուսակներում

Լաբորատոր աշխատանքների կատարման մեթոդական ցուցումներ

ԵՐԵՎԱՆ

ՃԱՐՏԱՐԱԳԵՏ

2019

Կազմող՝ Ա. Լ. Մանուկյան

Grid տեխնոլոգիան ոճերի կասկադավորման աղյուսակներում: Լաբորատոր աշխատանքների կատարման մեթոդական ցուցումներ: Ա. Լ. Մանուկյան: Հայաստանի ազգային Ճարտարագիտական համալսարան. – Եր.Ճարտարագետ, 2019 – 54 էջ:

Սույն մեթոդական ցուցումները հնարավորություն կընձեռեն Grid տեխնոլոգիայի օգնությամբ ուսանողների մոտ զարգացնել ՎԵԲ էջերի ձևավորման հմտությունները:

Նախատեսված է ՀԱՊՀ հիմնադրամի «Տեղեկատվական տեխնոլոգիաներ» (դասիչ` 061105.00.6) մասնագիտությամբ սովորող ուսանողների համար:

Գրախոսներ` Ս. Ս. Ավետիսյան

Ա. Գ. Պողոսյան

Երևան 2019

**Բովանդակություն**

[ՆԵՐԱԾՈՒԹՅՈՒՆ 6](#_Toc25545859)

[Grid-Ի ԳՈՐԾԱՌՈՒՅԹՆԵՐԸ 6](#_Toc25545860)

[ԶՆՆԱՐԿԻՉՆԵՐԻ ԱՋԱԿՑՈՒԹՅՈՒՆ 7](#_Toc25545861)

[ՀԻՄՆԱԿԱՆ ԵԶՐՈՒՅԹՆԵՐ 7](#_Toc25545862)

[Grid ԿՈՆՏԵՅՆԵՐԻ ՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ 10](#_Toc25545863)

[Display 10](#_Toc25545864)

[Grid-template-columns և grid-tempate-rows 12](#_Toc25545865)

[Grid-template-areas 17](#_Toc25545866)

[Grid-template 20](#_Toc25545867)

[Grid-row-gap և grid-column-gap 21](#_Toc25545868)

[Բացահայտ և ոչ բացահայտ ցանց 22](#_Toc25545869)

[Grid-auto-flow 30](#_Toc25545870)

[Minmax () ֆունկցիան 32](#_Toc25545871)

[Ներդրված grid տարրեր 34](#_Toc25545872)

[Տարրերի տեղադրում z-index-ի օգտագործմամբ 36](#_Toc25545873)

[Justify-items 38](#_Toc25545874)

[Align-items 40](#_Toc25545875)

[Justify-content 41](#_Toc25545876)

[Align-content 43](#_Toc25545877)

[Place-items 43](#_Toc25545878)

[Place-content 43](#_Toc25545879)

[Grid ՏԱՐՐԵՐԻ ՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ 45](#_Toc25545880)

[Grid-row-start, grid-row-end, grid-column-start, grid-column-end 45](#_Toc25545881)

[Grid-row և grid-column 47](#_Toc25545882)

[Grid-area 48](#_Toc25545883)

[Justify-self 50](#_Toc25545884)

[Align-self 51](#_Toc25545885)

# ՆԵՐԱԾՈՒԹՅՈՒՆ

Ոչ վաղ անցյալում վեբ էջերի գծանշումը կատարվում էր աղյուսակների, լողացող տարրերի և CSS-ի այլ հատկությունների միջոցով, որոնք այնքան էլ հարմար չէին բարդ վեբ էջերի մոդելավորման համար:

Այնուհետև մշակվեց flexbox-ը՝ գծանշման մոդուլ, որը հատուկ նախագծված էր հուսալի և հարմարվողական էջեր ստեղծելու համար: Flexbox-ը պարզեցրեց տարրերի և դրանց բովանդակության հավասարեցումը, և վեբ էջերի մշակման նախընտրելի համակարգ դարձավ:

Այժմ մշակվել է նոր՝ CSS-grid-Layout (այսուհետև՝ grid) համակարգը, որը վեբ էջերի գծանշման հզոր հնարավորություններ է ընձեռում: Grid-ը հզոր երկչափ համակարգ է, որը ի տարբերություն Flexbox- ի, կարող է միաժամանակ աշխատել սյուների և տողերի հետ:

# Grid-Ի ԳՈՐԾԱՌՈՒՅԹՆԵՐԸ

* Կարելի է ստեղծել ցանց ֆիքսված չափերով շերտերից (օգտագործելով փիքսելներ): Կարելի է նաև ստեղծել ճկուն չափեր ունեցող ցանց (օգտագործելով տոկոսները կամ այս նպատակով նախատեսված նոր fr միավորը):
* Կարելի է տեղադրել տարրերը ցանցի նշված մասում՝ օգտագործելով գծերի համարները կամ անունները: Ցանցը պարունակում է նաև ալգորիթմ՝ ցանցում ակնհայտ դիրքորոշում չունեցող տարրերի տեղադրման համար:
* Ցանցում կարելի է ավելացնել լրացուցիչ տողեր և սյուներ:
* Ցանցը պարունակում է հավասարեցման գործառույթներ, որը թույլ է տալիս վերահսկել տարրերի տեղադրումը ցանցի տարածքում:
* Ցանցի բջիջներում կարող է ավելի քան մեկ տարր տեղադրվել, կամ որոշ տարածքներ կարող են մասնակիորեն համընկնել: Այդ դեպքերը կարելի է վերահսկել z-ինդեքսի միջոցով:

# ԶՆՆԱՐԿԻՉՆԵՐԻ ԱՋԱԿՑՈՒԹՅՈՒՆ

Ստորև բերված է զննարկիչների համար տարբերակների ցանկ, որոնցից սկսվում է Grid Layout-ի ամբողջական աջակցությունը.

1. Սեղանադիր համակարգիչներ.

* Google Chrome - 57-րդ տարբերակից,
* Mozilla Firefox - 52-րդ տարբերակից,
* Opera- 44-րդ տարբերակից,
* Safari - 10.1 տարբերակից,
* Edge - 16-րդ տարբերակից:

1. Բջջային / պլանշետ.

* iOS Safari - 10.3 տարբերակից,
* Opera Mobile - 46-րդ տարբերակից,
* Android - 76-րդ տարբերակից,
* Android Chrome - 78-րդ տարբերակից,
* Android Firefox - 68-րդ տարբերակից:

# ՀԻՄՆԱԿԱՆ ԵԶՐՈՒՅԹՆԵՐ

**Grid կոնտեյներ** (grid container) -տարր, որում կառուցվում է ցանցը: Դա ուղղակի նախնի (ծնող) տարր է բոլոր grid տարրերի համար:

Օրինակ 1.

<div class="grid-container"> <!-- սա grid- կոնտեյներ է-->

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

</div>

**Grid տարրեր** (grid items) - ցանցում տեղադրված տարրեր: Դրանք պետք է լինեն ցանցի կոնտեյների անմիջական ժառանգները (դուստր տարրերը):

Օրինակ 2.

<div class="grid-container">

<div class="grid-item"></div> <!-- սա grid-տարր է -->

<div class="grid-item"></div> <!-- սա grid-տարր է -->

<div class="grid-item"> <!-- սա grid-տարր է -->

<div class="some-item"></div> <!-- սա grid-տարր **ՉԷ** -->

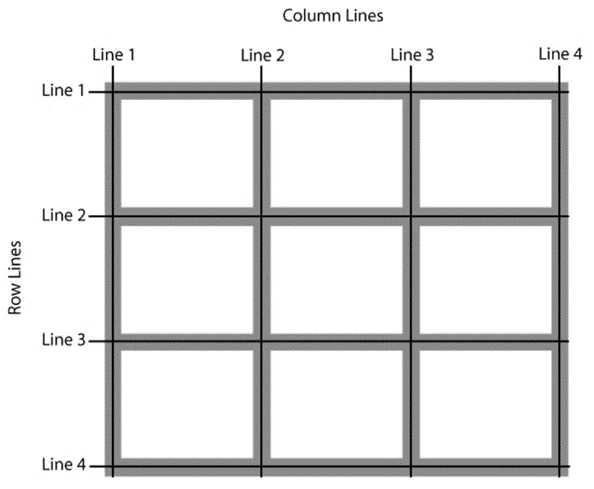
</div>

<div class="grid-item"></div> <!-- սա grid-տարր է -->

</div>

**Grid գծեր** (grid lines) - այս գծերը անտեսանելի ուղղահայաց (columns lines) և հորիզոնական (row lines) գծեր են (նկ. 1), որոնք բաժանում են ցանցի բջիջները և ձևավորում են դրա կառուցվածքը:

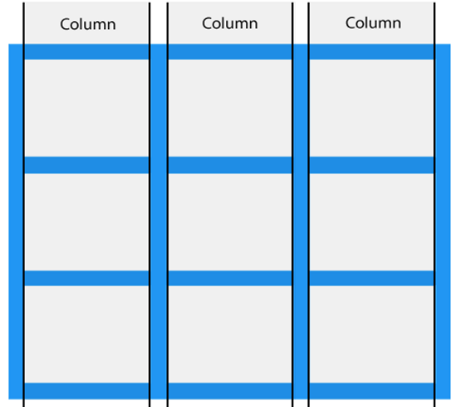
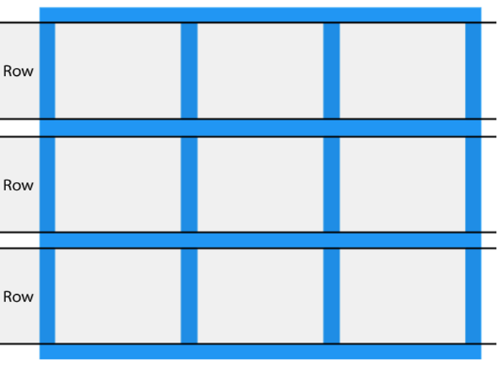
Grid գծերը ինքնաբերաբար համարակալված են: Անհրաժեշտության դեպքում դրանց կարող են տրվել անվանումներ: **Grid** գծերը կարող են կցվել grid տարրերին` թվերով կամ անվանումներով:



*Նկ. 1*

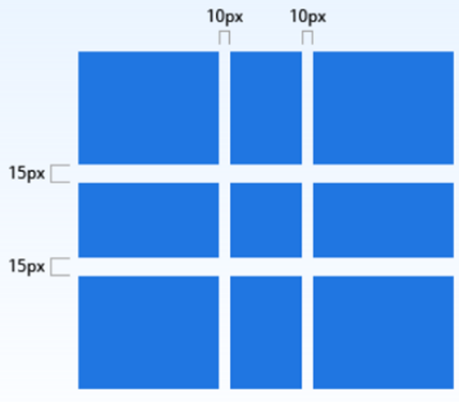
**Grid ուղի** (grid track) **-** gridուղիները (Նկ. 2) զույգ հարևան gridգծերով սահմանափակված տարածքներն են:

Ուղղահայաց gridուղիները ցանցի սյունակներն են (grid column), հորիզոնական ուղիները՝ ցանցի տողերը (grid row):



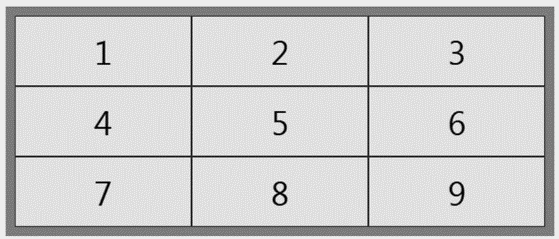
*Նկ. 2*

**Grid բացվածք (**gutters**) -** gridբացվածքները (Նկ. 3), gridուղիների միջև գտնվող դատարկ տարածքներ են:



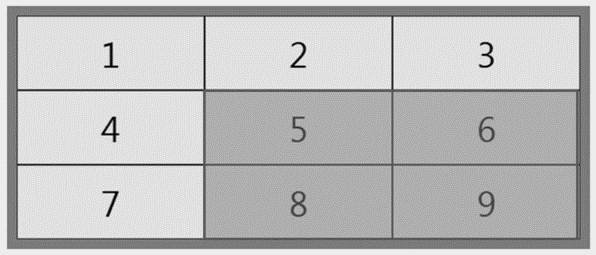
*Նկ. 3*

**Grid բջիջներ** (grid cell) **-** grid բջիջները երկու հարևան տողերի և երկու հարևան սյուների գծերի միջև գտնվող տարածքներն են (նկ. 4): Grid բջիջը ցանցի հիմնական ամենափոքր միավորն է:



*Նկ. 4*

**Grid տարածք** (grid area)- այս տարրը բաղկացած է մեկ կամ մի քանի grid բջիջներից (նկ. 5): Gridտարածքը պետք է լինի ուղղանկյուն, չի կարող T- աձև տարածք լինել:



*Նկ. 5*

# Grid ԿՈՆՏԵՅՆԵՐԻ ՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ

Այս գլխում կդիտարկենք CSS Grid Layout-ի հատկությունները grid կոնտեյների համար.

* display,
* grid-template-columns, grid-template-rows,
* grid-template-areas,
* grid-column-gap, grid-row-gap, grid-gap,
* justify-items, align-items,
* justify-content, align-content,
* grid-auto-columns, grid-auto-rows, grid-auto-flow,
* grid:

## Display

Display հատկության միջոցով ստեղծվում է grid կոնտեյներ, որի բոլոր դուստր տարրերը ինքնաբերաբար դառնում են grid տարրեր:

Ձևաչափը.

**display: grid | inline-grid | subgrid;**

որտեղ.

* **grid** -ը ցանցը ձևավորում է որպես բլոկ,
* **inline-grid** -ը ցանցը ձևավորում է որպես ներտողային բլոկ,
* **subgrid** – արժեքի դեպքում ներդրված կոնտեյներ-ցանցի տողերի/ սյունակների չափերը վերցվում են ծնող տարրից:

Column, float, clear և vertical-align հատկությունները որևէ ազդեցություն չունեն կոնտեյների վրա:

Օրինակ 3.

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

.container{**display:grid;**

border:2px solid #555;

background: #777;

}

.grid-item {

border:1px solid #555;

background: #ccc;

padding:20px;

}

</style>

</head>

<body>

<div class="container">

<div class="grid-item">Մեկ</div>

<div class="grid-item">Երկու</div>

<div class="grid-item">Երեք</div>

<div class="grid-item">Չորս</div>

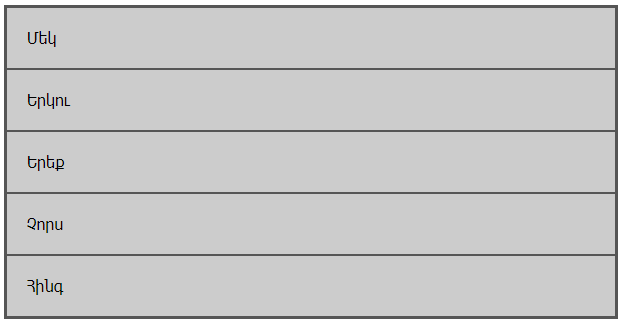
<div class="grid-item">Հինգ</div>

</div>

</body>

</html>

Արդյունքը զննարկչում.



## Grid-template-columns և grid-tempate-rows

Այս հատկությունները որոշում են ցանցի տողերի և սյուների չափերը: Արժեքները միմյանցից բաժանվում են բացատներով, որոնք սահմանում են gridուղիների չափը: Այդտեղ կարելի է նշել նաև gridգծերի անվանումները:

Ձևաչափը.

**grid-template-columns: <track-size> ... | <line-name> <track-size> ...;**

**grid-template-rows: <track-size> ... | <line-name> <track-size> ...;**

որտեղ**.**

* **<track-size> -**ըֆիքսված չափ, տոկոս կամ ցանցի ազատ տարածության մաս է, որը որոշվում է **fr** (fraction) միավորի միջոցով,
* **<line-name> -**ըկամայական անվանում է:

Օրինակ 3-ում նկարագրված ցանցում սահմանենք 200px չափի երեք սյուն:

Օրինակ 4.

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

.container{**display:grid;**

**grid-template-columns: 200px 200px 200px;**

border:2px solid #555;

background: #777;

}

.grid-item {

border:1px solid #555;

background: #ccc;

padding:20px;

}

</style>

</head>

<body>

<div class="container">

<div class="grid-item">Մեկ</div>

<div class="grid-item">Երկու</div>

<div class="grid-item">Երեք</div>

<div class="grid-item">Չորս</div>

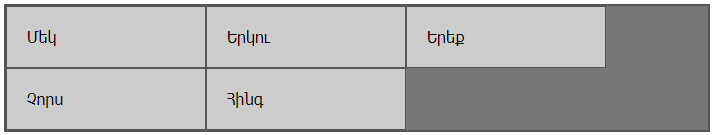
<div class="grid-item">Հինգ</div>

</div>

</body>

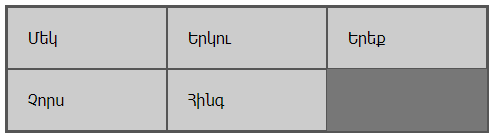
</html>

Արդյունքը զննարկչում.



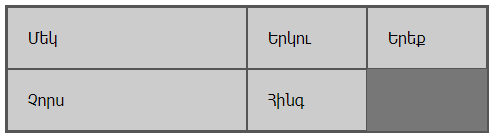
Օրինակ 4-ի wrapper դասում փոխենք grid-template-columns հատկության արժեքները՝

**ա) grid-template-columns: 1fr 1fr 1fr;**



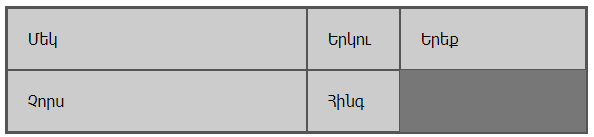
Այս դեպքում մատչելի տարածքը բաժանվում է երեք հավասար սյուներում:

**բ) grid-template-columns: 2fr 1fr 1fr;**



Այստեղմատչելի տարածքի ½ -ը հատկացվել է առաջին սյանը, մյուս երկու սյուները ստացել են ¼ մասեր:

**գ) grid-template-columns: 300px 1fr 2fr;**



Այս դեպքում մատչելի տարածքի 300px –ը հատկացվել է առաջին սյանը, մնացած տարածքի 1/3 -ը հատկացվել է երկրորդ սյանը, իսկ 2/3 –ը երրորդ սյանը:

Տարրի չափի արժեքի հայտարագրերը կրկնելու համար կարելի է օգտագործել repeat() ֆունկցիան:

Օրինակ 5.

.wrapper {

display: grid;

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

}

Կրճատ գրելաձևը.

.wrapper {

display: grid;

**grid-template-columns: repeat(3, 1fr);**

}

Օրինակ 6.

.wrapper {

display: grid;

**grid-template-columns: 20px repeat(6, 1fr) 20px;**

}

Օրինակ 7.

.wrapper {

display: grid;

**grid-template-columns: repeat(5, 1fr 2fr);**

}

Եթե չի նշվում grid գծերի անվանումները grid ուղիների արժեքների միջև, ապա grid գծերին ինքնաբերբար տրվում են դրական և բացասական թվային անուններ (նկ. 6):

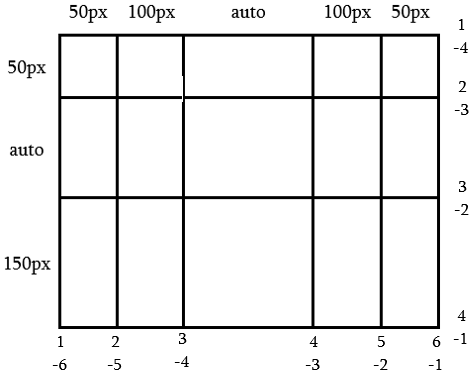
Օրինակ 8.

.container{

grid-template-columns: **50px 100px auto 100px 50px**;

grid-template-rows: **50px auto 150px**;

}



*Նկ. 6*

Կարելի է նաև grid գծերի անվանումները բացահայտ նշել (նկ. 7):

Օրինակ 9.

.container {

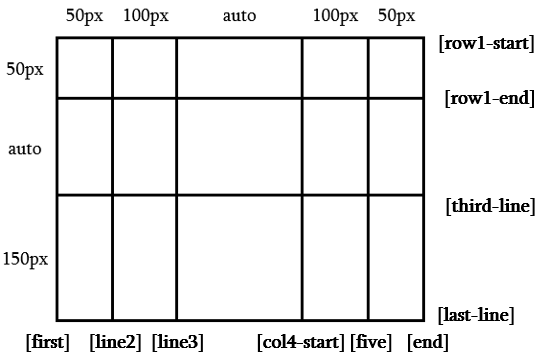
grid-template-columns: **[first] 50px [line2] 100px [line3] auto [col4-start] 100px [five] 50px [end];**

grid-template-rows: **[row1-start] 50px [row1-end] auto [third-line] 150px [last-line];**

}

Տողերը կարող են ունենալ մի քանի անուններ: Օրինակ, այստեղ երկրորդ տողը կունենա երկու անուն ՝ row1-end և row2-start.

grid-template-rows: **[row1-start] 50px [row1-end row2-start] auto [third-line] 150px [last-line];**



*Նկ. 7*

## Grid-template-areas

Grid-template-areas հատկությունը սահմանում է ցանցի ձևանմուշը՝ օգտագործելով grid-տարածքների անունները, որոնք սահմանվում են grid-area հատկությունների միջոցով (այս հատկությունը կիրառվում է grid-տարրերի նկատմամբ):

Grid տարածքի անվան կրկնություններից որոշվում է, թե քանի grid բջիջներ են ձևավորում այս տարածքը: Կետը նշանակում է դատարկ grid բջիջ: Շարահյուսությունն ինքնին արտացոլում է ցանցի կառուցվածքը:

Արժեքները.

* <grid-area-name > - տարածքի անունը տրված է grid-area-ի միջոցով,
* . - կետը նշանակում է դատարկ բջիջ,
* none-տարածքները սահմանված չեն:

Ձևաչափը.

.container {

**grid-template-areas:**

**"<grid-area-name> | . | none | ..."**

**"...";**

}

Օրինակ 10.

.grid-item-header {

grid-area: header;

}

.grid-item-main {

grid-area: main;

}

.grid-item-sidebar {

grid-area: sidebar;

}

.grid-item-footer {

grid-area: footer;

}

.grid-container {

display: grid;

grid-template-columns: repeat(5, 1fr);

grid-template-rows: auto;

grid-template-areas: "header header header header header"

"main main main . sidebar"

"footer footer footer footer footer";

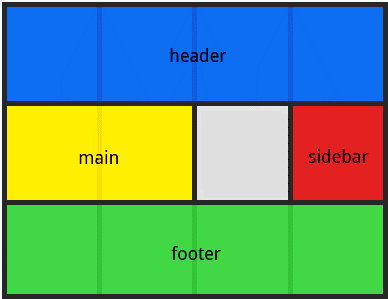
}

Այս կոդը սահմանում է հինգ սյունով և երեք տողով ցանց (նկ. 8): Սյունակների լայնությունը նույնն է՝ յուրաքանչյուրը գրավում է մեկ բաժին (1fr) grid-կոնտեյների ազատ տարածությունից։ Ամբողջ վերին գիծը grid-տարածք է, որը կոչվում է header և պարունակում է հինգ բջիջ:

Միջին գիծը ներառում է երեք բջջից բաղկացած main grid-տարածքը, մեկ դատարկ բջիջ և մեկ բջջից բաղկացած sidebar grid- տարածքը:

Վերջին գիծը ամբողջովին զբաղեցնում է footer grid-տարածքը հինգ բջջով:

Այս ցանցի վերջնական սխեման հետեւյալն է.



*Նկ. 8*

Հայտարարության մեջ յուրաքանչյուր տող պետք է ունենա նույն թվով բջիջներ:

Մեկ դատարկ grid բջջի ստեղծման համար կարելի է օգտագործել ցանկացած թվով կետեր: Այլ կերպ ասած, քանի դեռ այդ կետերի միջև բացատներ չկան, դրանք մեկ դատարկ grid բջիջ են։

Պետք է նկատի ունենալ, որ այստեղ grid-գծերին անուններ չի տրվում, միայն grid-տարածքներն են անուններ ստանում: Ընդ որում, grid-տարածքների երկու ծայրերում grid-գծերը կստանան անուններ ինքնաբերաբար:

Օրինակ, եթե grid-տարածքը կոչվում է header, ապա սկզբնական հորիզոնական և սկզբնական ուղղահայաց grid-գծերը կստանան header-start անունը, իսկ վերջնական հորիզոնական և վերջնական ուղղահայաց grid-գծերը կստանան header-end անունը: Այսպիսով, որոշ grid գծեր կարող են ունենալ մի քանի անուններ:

Վերը նշված օրինակում սկզբնական ձախ grid-գիծն ունի երեք անուն՝ header-start, main-start և footer-start:

## Grid-template

**Grid-template** հատկությունը կրճատ գրառում է, որը թույլ է տալիս միաժամանակ սահմանել **grid-template-rows, grid-template-columns և grid-template-areas** հատկությունների արժեքները:

Ձևաչափը.

.container {

**grid-template: none | <grid-template-rows> / <grid-template-columns>;**

}

Արժեքները.

* **none** - սահմանում է բոլոր երեք հատկությունների ելակետային արժեքները,
* **< grid-template-rows> / <grid-template-column> -** սահմանում է grid-template-column և grid-template-rows հատկություններին համապատասխան նշված արժեքները, իսկ grid-template-areas հատկությանը վերագրում է none արժեքը:

Ամբողջական և կրճատ գրելաձևի օրինակ.

.grid-container {

**grid-template-rows**: [row1-start] 100px [row1-end row2-start]

100px [row2-end];

**grid-template-columns**: auto 50px auto;

**grid-template-areas**: "header header header"

"footer footer footer";

}

Կրճատ գրելաձև.

.grid-container {

**grid-template**:[row1-start] "header header header" 100px [row1-end]

[row2-start] "footer footer footer" 100px [row2-end]

/ auto 50px auto;

}

## Grid-row-gap և grid-column-gap

Բջիջների միջև grid բացվածք (Նկ․ 9) կարելի է ստեղծել՝ grid-column-gap և grid-row-gap հատկությունների, կամ grid-gap կրճատ հատկության միջոցով․

.container {

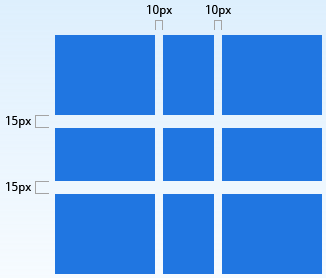
grid-template-columns: 100px 50px 100px;

grid-template-rows: 80px auto 80px;

**grid-column-gap: 10px;**

**grid-row-gap: 15px;**

}



*Նկ*. *9*

Grid բացվածքները ստեղծվում են միայն սյուների և շարքերի միջև, բայց ոչ ցանցի արտաքին եզրերի համար:

Grid-gap հատկությունը grid-row-gap և grid-column-gap հատկությունների կրճատ ձևն է: Այս դեպքում արժեքի մեջ նշվում է բացատով բաժանված երկու պարամետր, առաջինը սահմանում է տողերի՝ երկրորդը սյուների միջև բացվածքրի չափը․

.grid-container {

**grid-gap: 30px 15px;**

}

Եթե տողերի և սյուների միջև նախատեսվում է սահմանել նույն բացվածքը, ապա grid-gap հատկության համար կարելի է նշել մեկ պարամետր․

.grid-container {

**grid-gap: 20px;**

}

## Բացահայտ և ոչ **բացահայտ** ցանց

CSS Grid- ը օգտագործում է բացահայտ և ոչ բացահայտ ցանցի հայեցակարգը:

Բացահայտ ցանցը սահմանվում է grid-template-rows, grid-template-column և grid-template-area հատկությունների միջոցով։

Երբ grid տարրերը գտնվում են բացահայտ ցանցից դուրս grid կոնտեյները ինքնաբերաբար գեներացնում է ոչ բացահայտ ցանց։

Օրինակ 11.

<!DOCTYPE HTML>

<html>

<head>

<style type="text/css">

.wrapper {display:grid;

**grid-template-rows: 50px 50px;**

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

grid-gap: 10px;

border:2px solid #555;

background: #777;

}

.grid-item {

border:1px solid #555;

background: #ccc;

text-align: center;

}

</style>

</head>

<body>

<div class="wrapper">

<div class="grid-item">Մեկ</div>

<div class="grid-item">Երկու</div>

<div class="grid-item">Երեք</div>

<div class="grid-item">Չորս</div>

<div class="grid-item">Հինգ</div>

<div class="grid-item">Վեց</div>

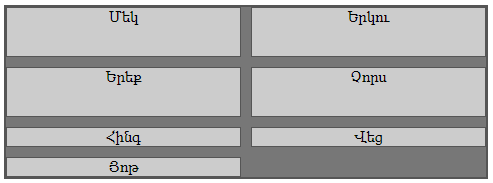
<div class="grid-item">Յոթ</div>

</div>

</body>

</html>

Զննարկիչում կստանանք հետևյալ արդյունքը՝



Այս օրինակում, հստակ սահմանվել է երկու տող և երկու սյուն, որտեղ տեղավորվել է չորս grid տարր։ Սակայն կա հինգ grid տարր։ Այսպիսով ստեղծվել է ոչ բացահայտ ցանց, որպեսզի տեղավորվի հինգերրդ լրացուցիչ տարրը։

Երևում է, որ լրացուցիչ շարքը նույնքան բարձր չէ, որքան նախորդ երկուսը: Դրա պատճառն այն է, որ տողի բարձրությունը սահմանվել է grid-template-rows հատկության միջոցով, որն օգտագործվում է միայն բացահայտ ցանցերի համար:

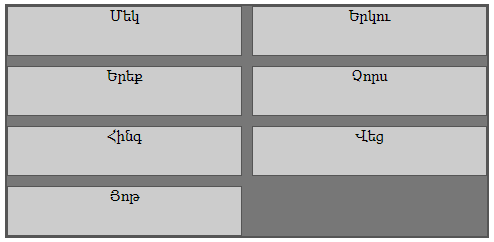
Ոչ բացահայտ ցանցի տողի բարձրությունը պետք է սահմանվի grid-auto-rows հատկության միջոցով: Բայց քանի դա արված չէ, կիրառվել է բովանդակության չափը:

Եթե օրինակ 11-ում ավելացնենք հետևյալ նկարագրությունը՝

.wrapper {

grid-auto-rows: 50px; }

կստանանք նույն չափի (50px) տողեր․



Դիտարկենք ևս մեկ օրինակ.

Օրինակ 12.

<!DOCTYPE html>

<html>

<head>

<title>grid-auto-rows</title>

<style>

.grid-container,

.grid-container2 {

display: grid;

margin-top: 5px;

padding: 10px;

background: #ccc;

}

.grid-container > div,

.grid-container2 > div {

background: #666;

border: 1px solid #000;

text-align: center;

}

.grid-container {

grid-template-columns: 50% 50%;

grid-template-rows: 20px 20px;

grid-auto-rows: 40px;

}

.grid-container2 {

grid-template-columns: 50% 50%;

grid-template-rows: 20px 20px;

grid-auto-rows: 40px 65px;

}

.item-a {

grid-row: 3 / 4;

background: #888 !important;

}

.item-b {

grid-row: 4 / 5;

background: #999 !important;

}

</style>

</head>

<body>

<div class = "grid-container">

<div class = "item-a">A</div>

<div class = "item-b">B</div>

<div>C</div>

<div>D</div>

<div>E</div>

<div>F</div>

</div>

<div class = "grid-container2">

<div class = "item-a">A</div>

<div class = "item-b">B</div>

<div>C</div>

<div>D</div>

<div>E</div>

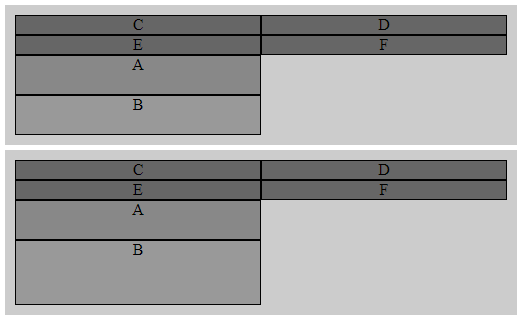
<div>F</div>

</div>

</body>

</html>

Արդյունքը զննարկիչում.



Այս օրինակում յուրաքանչյուր կոնտեյներում գեներացվել է երկու ոչ բացահայտ տող։ Առաջին կոնտեյներում յուրաքանչյուր ոչ բացահայտ տողի չափը 40 պիքսել է, իսկ երկրորդ կոնտեյներում առաջին ոչ բացահայտ տողին համապատասխանում է 40 պիքսել, իսկ երկրորդին՝ 65 պիքսել:

Այժմ դիտարկենք grid-auto-columns հատկությունը։

Օրինակ 13․

<!DOCTYPE html>

<html>

<head>

<title>grid-auto-columns</title>

<style>

.grid-container,

.grid-container2 {

display: grid;

margin-top: 5px;

padding: 10px;

background: #ccc;

}

.grid-container > div,

.grid-container2 > div {

background: #666;

border: 1px solid #000;

text-align: center;

padding: 20px 0;

}

.grid-container {

grid-template-columns: 25px 100px;

grid-auto-columns: 100px;

}

.grid-container2 {

grid-template-columns: 1fr 1fr;

grid-auto-columns: 2fr;

}

.item-a {

grid-column: 3 / 4;

grid-row: 2 / 3;

background: #888 !important;

}

.item-b {

grid-column: 4 / 5;

grid-row: 2 / 3;

background: #999 !important;

}

</style>

</head>

<body>

<div class = "grid-container">

<div class = "item-a">A</div>

<div class = "item-b">B</div>

<div>C</div>

<div>D</div>

<div>E</div>

<div>F</div>

</div>

<div class = "grid-container2">

<div class = "item-a">A</div>

<div class = "item-b">B</div>

<div>C</div>

<div>D</div>

<div>E</div>

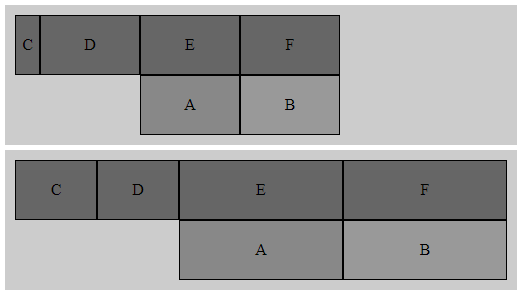
<div>F</div>

</div>

</body>

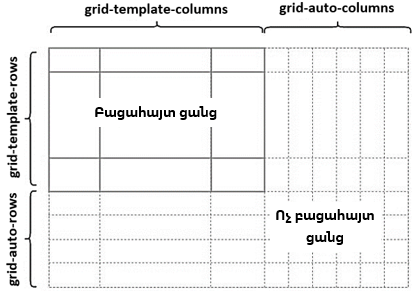
</html>

Այս օրինակում երկու կոնտեյներից յուրաքանչյուրը պարունակում է չորս սյուն, որոնցից երկուսը ստեղծվել են ոչ բացահայտ, և երկու տող: Առաջին կոնտեյներում ոչ բացահայտ սյուների չափը 100 պիքսել է, իսկ երկրորդ կոնտեյներում՝ 2fr ։



Ընդհանուր առմամբ բացահայտ ցանցն օգտագործում է grid-template-rows և grid-template-columns հատկությունները, իսկ ոչ բացահայտ ցանցը՝ grid-auto-rows և grid-auto-columns հատկությունները։

Նկ. 10-ում բերված սխեմայում ցուցադրվում է, թե ինչպես են կոնտեյների հատկությունները սահմանում համապատասխան gridուղիների չափերը։



*Նկ*. *10*

## Grid-auto-flow

Եթե որոշ grid տարրեր բացահայտ չեն դիրքավորվում ցանցում, ապա ավտոմատ տեղադրման ալգորիթմը գործարկվում է դրանք ինքնաբերաբար տեղադրելու համար:

Grid-auto-flow  հատկությունը սահմանում է, թե ինչպես պետք է աշխատի ավտոմատ տեղադրման ալգորիթմը:

Թույլատրելի արժեքներն են.

* **row**  (լռելյայն արժեք) - ցանցի տողերը լրացվում են հերթով, անհրաժեշտության դեպքում ավելացվում են նոր տողեր,
* **column** - ցանցի սյուները լրացվում են հերթով, անհրաժեշտու-թյան դեպքում ավելացվում են նոր սյուներ,
* **dense**  - լրացնել ցանցի նախկինում ձևավորված բաց տարածքները, եթե հայտնվեն համապատասխան չափի grid տարրեր: Սա կարող է հանգեցնել տարրերի արտածման կարգի փոփոխության:

Ձևաչափը․

**grid-auto-flow: row | column | row dense | column dense**

Օրինակ 14.

<!doctype html>

<html>

<head>

<title>Example</title>

<style type="text/css">

.grid-container {

display: grid;

grid-template-rows: 50px 50px;

grid-template-columns: 100px 100px 100px 100px 100px;

grid-auto-flow: column;

background:#fff;

}

div {

background:#999;

border:1px solid black;

font-size:25px;

}

.item-a {

grid-column: 1;

grid-row: 1 / 3;

background:#ccc;

}

.item-e {

grid-column: 5;

grid-row: 1 / 3;

background:#ccc;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="item-a">item-a</div>

<div class="item-b">item-b</div>

<div class="item-c">item-c</div>

<div class="item-d">item-d</div>

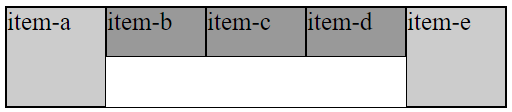
<div class="item-e">item-e</div>

</div>

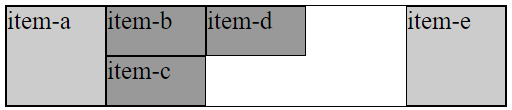
</body>

</html>

Արդյունքը զննարկիչում.



Այժմ grid-auto-flow հատկության row արժեքը փոխենք column-ով, կստանանք հետևյալ արդյունքը.



## Minmax () ֆունկցիան

Բացահայտ ցանցի չափերը նշելիս կամ ինքնաբերաբար ստեղծված սյուների կամ տողերի չափերը որոշելիս հաճախ անհրաժեշտություն է առաջանում grid ուղիների համար սահմանել նվազագույն չափ, բայց վստահ լինել, որ անհրաժեշտության դեպքում grid ուղիները կձգվեն դրանցում ավելացված ամբողջ բովանդակությունը տեղավորելու համար:

Այսպիսի դեպքերում հարմար է օգտագործել minmax() ֆունկցիան, որը սահմանում է grid ուղու նվազագույն և առավելագույն չափերը։

Ձևաչափը՝ **minmax(min, max)**

Օրինակ 14.

<!doctype html>

<html>

<head>

<title>Example</title>

<style type="text/css">

.container {

display: grid;

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

grid-template-rows: repeat(3, minmax(55px, auto));

grid-gap: 5px;

}

.item {

border:1px solid;

background: #bbb;

padding:5px;

}

</style>

</head>

<body>

<div class="container">

<div class="item item1">1</div>

<div class="item item2">2</div>

<div class="item item3">3</div>

<div class="item item4">4</div>

<div class="item item5">

<p>Minmax () ֆունկցիան սահմանում է grid ուղու

նվազագույն և առավելագույն չափերը։</p>

</div>

<div class="item item6">6</div>

<div class="item item7">7</div>

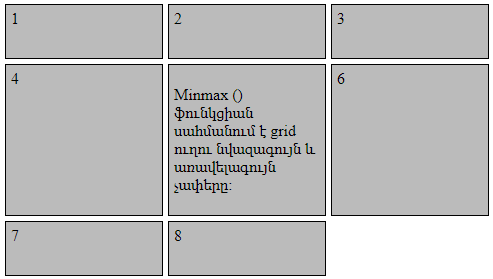
<div class="item item8">8</div>

</div>

</body>

</html>

Արդյունքը զննարկիչում.



## Ներդրված grid տարրեր

Grid տարրերը կարող են իրենք դառնալ grid կոնտեյներ։ Այսինքն՝ կարելի է ներդնել grid-տարրը մեկ այլ grid-տարրի մեջ՝ դրանով իսկ ստեղծելով ներդրված grid տարր։

Նման ներդրված ցանց ստեղծելու համար միայն պետք է կիրառել display:grid  (կամ display:inline-grid) հատկությունը համապատասխան grid տարրի վրա: Կարեի է նաև օգտագործել display:subgrid  հատկությունը:

Օրինակ 13.

<!doctype html>

<html>

<head>

<title>Example</title>

<style type="text/css">

#outer-grid {

display: grid;

grid-template-rows: 1fr 1fr;

grid-template-columns: 1fr 1fr;

grid-gap: 8px;

}

#outer-grid > div {

background-color: limegreen;

color: white;

font-size: 4vw;

padding: 8px;

}

#inner-grid {

display: grid;

grid-template-columns: 1fr 1fr;

grid-gap: 5px;

}

#inner-grid > div {

background: lightgreen;

padding: 8px;

}

</style>

</head>

<body>

<div id="outer-grid">

<div>1</div>

<div>2</div>

<div>3</div>

<div id="inner-grid">

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

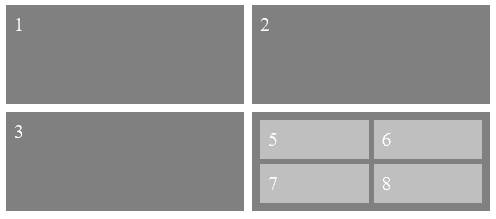
</div>

</div>

</body>

</html>

Արդյունքը զննարկիչում.



## Տարրերի տեղադրում z-index-ի օգտագործմամբ

Մի քանի grid տարր կարող է զբաղեցնել նույն բջիջը: Դիտարկենք հետևյալ օրինակը.

Օրինակ 14.

<!doctype html>

<html>

<head>

<title>Example</title>

<style>

.wrapper {

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-auto-rows: 100px;

}

.box1 {

grid-column-start: 1;

grid-column-end: 4;

grid-row-start: 1;

grid-row-end: 3;

}

.box2 {

grid-column-start: 1;

grid-row-start: 2;

grid-row-end: 4;

}

.box {

border:2px solid #333;

font-size:2em;

}

.box1 {background:#555;}

.box2 {background:#ccc;}

.box3 {background:#999;}

.box4 {background:#ddd;}

.box5 {background:#777;}

</style>

</head>

<body>

<div class="wrapper">

<div class="box box1">Մեկ</div>

<div class="box box2">Երկու</div>

<div class="box box3">Երեք</div>

<div class="box box4">Չորս</div>

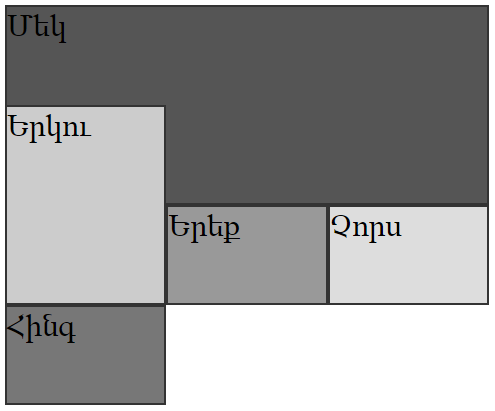
<div class="box box5">Հինգ</div>

</div>

</body>

</html>

Արդյունքը զննարկիչում.



*Նկ*. *11*

Box2 տարրը այժմ ծածկում է box1-ին և ցուցադրվում է վերևում, քանի որ կոդում box1-ից հետո է սահմանված:

Այն կարգը, որում տարրերը տեղադրվում են միմյանց վրա, կարելի է կառավարել z-index հատկության միջոցով՝ ճիշտ այնպես, ինչպես դիրքավորված տարրերի դեպքում։

Եթե box1-ի Z-index հատկությանը տանք ավելի մեծ արժեք, քան box2-ին, ապա box1-ը կծածկի box2-ին և կցուցադրվի վերևում (նկ․ 12):

Օրինակ 14-ում ավելացնենք հետևյալ նկարագրությունը․

.box1 {

z-index: 2;

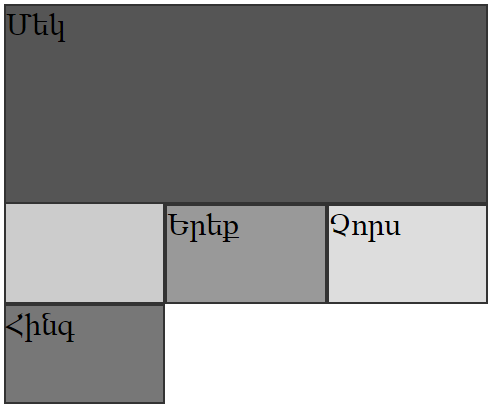
}

.box2 {

z-index: 1;

}

կստանանք այս պատկերը.



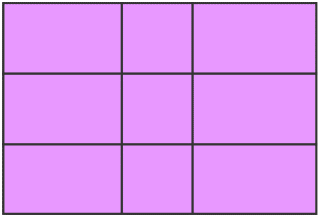
*Նկ*. *12*

## Justify-items

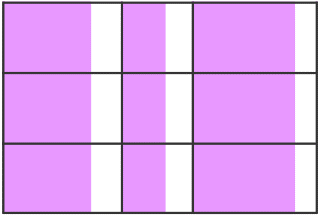
Justify-items հատկությունը հավասարեցնում է grid-տարրերի բովանդակությունը տողի առանցքի երկայնքով: Արժեքնրն ազդում են կոնտեյների բոլոր grid-տարրերի վրա:

Այս հատկությունն ընդունում է հետևյալ արժեքները.

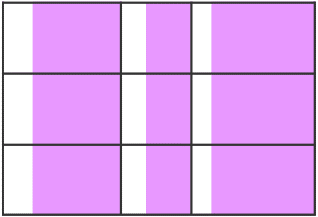
* **stretch**  (լռելյայն արժեք) - բովանդակությունը լրացնում է grid տարածքի ամբողջ լայնությունը.



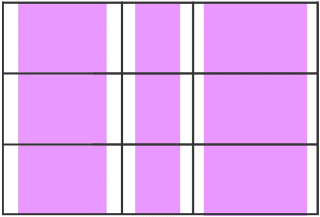
* **start** - բովանդակությունը սեղմվում է grid տարածքի ձախ եզրին.



* **end** - բովանդակությունը սեղմվում է grid տարածքի աջ եզրին.



* **center** – տեղի է ունենում հավասարեցում ըստ grid տարածքի կենտրոնի .

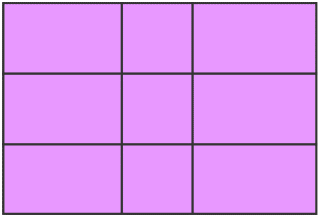


## Align-items

Align-items հատկությունը հավասարեցնում է grid-տարրերի բովանդակությունը սյան առանցքի երկայնքով: Արժեքնրն ազդում են կոնտեյների բոլոր grid-տարրերի վրա:

Այս հատկությունն ընդունում է հետևյալ արժեքները.

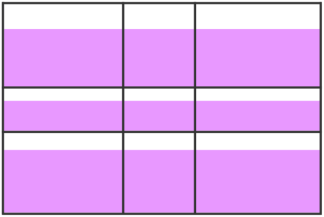
* **stretch**  (լռելյայն արժեք) - բովանդակությունը լրացնում է grid տարածքի ամբողջ լայնությունը.



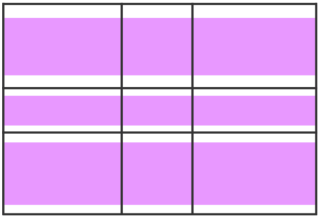
* **start**  - բովանդակությունը սեղմվում է grid տարածքի վերին մասում.



* **end** - բովանդակությունը սեղմվում է ցանցի տարածքի ստորին եզրին.



* **center** – տեղի է ունենում հավասարեցում ըստ grid տարածքի կենտրոնի.

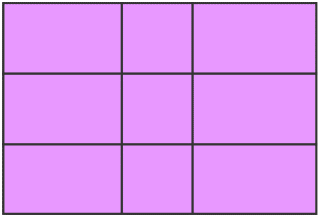


## **Justify**-content

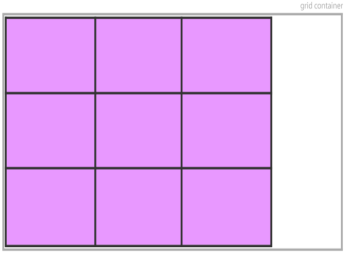
Justify-contentհատկությունըհավասարեցնում է ցանցը տողի առանցքի երկայնքով:

Թույլատրելի արժեքներն են.

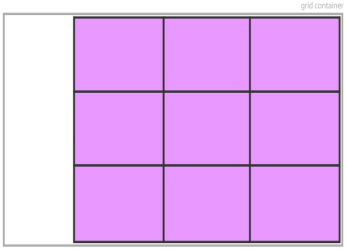
* **stretch** - փոփոխում է ցանցի տարրերի չափերը այնպես, որ ցանցը կարողանա լրացնել grid կոնտեյներիամբողջ լայնությունը.



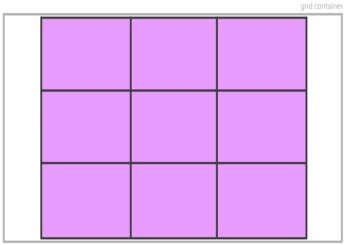
* **start** - ցանցը հավասարեցնում է ըստ grid կոնտեյներիձախ եզրի.



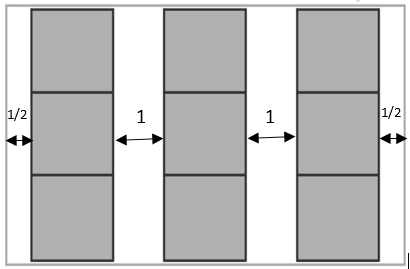
* **end**- ցանցը հավասարեցնում է ըստ grid կոնտեյներիաջ եզրի.



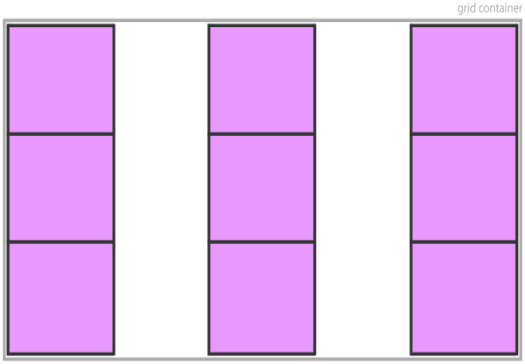
* **center** - ցանցը հավասարեցնում է ըստ grid կոնտեյներիկենտրոնի.



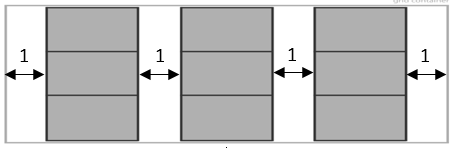
* **space-around -** grid տարրերի միջև ավելացվում է հավասար չափի տարածք և այս տարածքի կեսի չափով տարածք ավելացվում է ցանցի եզրերին.



* **space-between -** grid տարրերի միջև ավելացվում է հավասար չափի տարածք, իսկ ցանցի ծայրերում ոչին չի ավելացվում.



* **space-evenly** - grid տարրերի միջև ավելացվում է հավասար չափի տարածք՝ ներառյալ ցանցի ծայրերը։



## Align-content

Align-content հատկությունը հավասարեցնում է ցանցը սյան առանցքի երկայնքով և ընդունում է նույն արժեքները, ինչ որ justify-content -ը։

Նշենք միայն start  և end արժեքների տարբերությունը։ Եթե justify-content-ի դեպքում ցանցը հավասարեցված է ձախ (start) կամ աջ (end) եզրին, ապա այդ արժեքները align-content  հատկությամբ օգտագործելու դեպքում ցանցը հավասարեցվում է կամ grid կոնտեյներիվերին մասում (start), կամ ներքևում (end):

## Place-items

Place-items հատկությունը grid կոնտեյների բոլոր տարրերը հավասարեցնում է սյան և տողի առանցքների երկայնքով (սա align-Items և [justify-items](https://basicweb.ru/css/css_pr_justify-items.php) հատկությունների կրճատ գրելաձևն է):

Ձևաչափը՝

**place-items**: **"align-items"**;

**place-items**: **"align-items justify-items"**;

Եթե երկրորդ արժեքը բացակայում է, ապա առաջին արժեքն օգտագործվում է երկուսի համար։

**align-items**: **"stretch | center | start | end | baseline | initial | inherit"**;

**justify-items**: **"stretch | center | start | end | baseline | initial | inherit"**;

## Place-content

Place-content  հատկությունը որոշում է, թե ինչպես պետք է զննարկիչը ցանցը հավասարեցնի grid կոնտեյների սյան և տողի առանցքների երկայնքով (սա align-content և justify-content հատկությունների կրճատ գրելաձևն է):

Ձևաչափը՝

**place-content**: **"align-content"**;

**place-content**: **"align-content justify-content"**;

Եթե երկրորդ արժեքը բացակայում է, ապա առաջին արժեքն օգտագործվում է երկուսի համար։

**align-content**: **"stretch | flex-start\* | flex-end\* | center | space-between | space-around | space-evenly | initial | inherit"**;

**justify-content**: **"flex-start\* | flex-end\* | center | space-between | space-around | space-evenly | initial | inherit"**;

# Grid ՏԱՐՐԵՐԻ ՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ

Այս գլխում կդիտարկենք grid տարրերի համար սահմանված CSS Grid Layout-ի հատկությունները.

* grid-column-start, grid-column-end,
* grid-row-start, grid-row-end,
* grid-column, grid-row, grid-area,
* justify-self, align-self:

## Grid-row-start, grid-row-end, grid-column-start, grid-column-end

Օգտագործելով այս հատկությունները՝ կարելի է նշանակել ցանցում grid տարրի դիրքը՝ հղում կատարելով grid գծերին:

Grid-row-start / grid-column-start  հատկությունները որոշում են այն grid գիծը, որից սկսվում է grid տարրը, իսկ grid-row-end / grid-column-end հատկությունները որոշում են այն grid գիծը, որում ավարտվում է grid տարրը:

Թույլատրելի արժեքներն են.

* **<line>** - թիվ (grid գծի համարը) կամ grid գծի անվանումը,
* **span <number>** - grid տարրը կզբաղեցնի նշված քանակի grid ուղիներ,
* **span <name>** - grid տարրը կզբաղեցնի մինչև այն grid գիծը եղած միջակայքը, որի անունը նշված է արժեքի մեջ,
* **auto**  - ավտոմատ տեղադրում:

Կիրառման սխեման․

**.grid-item {**

**grid-row-start: <number>** կամ

**<name>** կամ

**span <number>** կամ

**span <name>** կամ

**auto**

**}**

Սխեման կիրառելի է նաև մյուս երեք հատկության համար։

Օրինակ 15․

.grid-item {

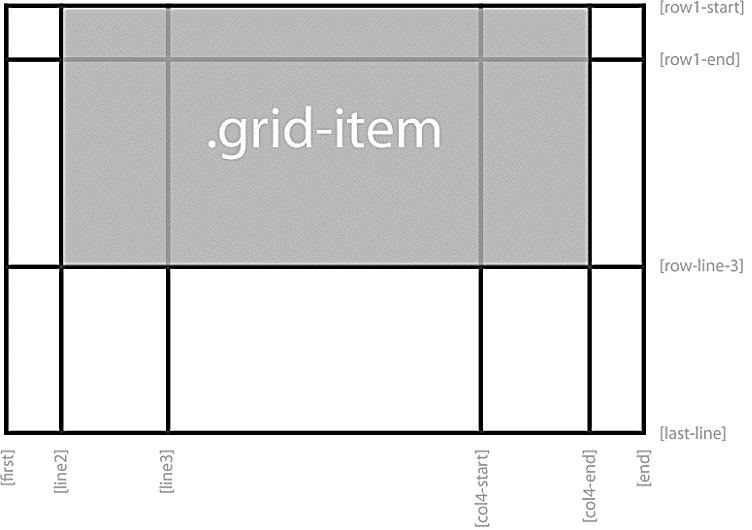
grid-row-start: row1-start;

grid-row-end: 3;

grid-column-start: 2;

grid-column-end: col4-end;

}



Օրինակ 16.

.grid-item {

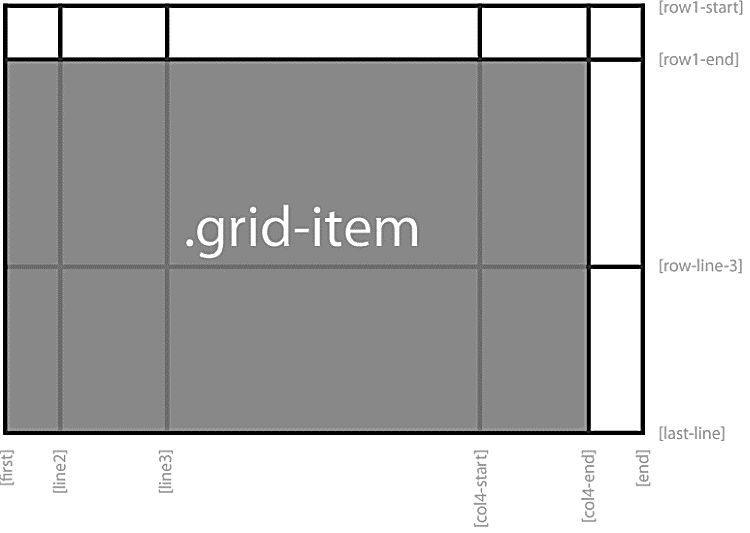
grid-row-start: 2;

grid-row-end: span 2;

grid-column-start: 1;

grid-column-end: span col4-end;

}



Եթե grid-row-end / grid-column-end  հատկությունները հայտարար-ված չեն, ապա լռելյայն grid տարրը կզբաղեցնի մեկ grid ուղի:

## Grid-row և grid-column

Grid-row և grid-column հատկությունները grid-row-start + grid-row-end և grid-column-start + grid-column-end հատկությունների կարճ գրելաձևերն են։

Արժեքները նույն արտահայտություններն են, ինչ որ կանոնավոր տարբերակների հատկություններինը:

Կիրառման սխեման.

**.grid-item {**

**grid-row: <start-line> / <end-line> կամ**

**<start-line> / span <value>;**

**grid-column: <start-line> / <end-line> կամ**

**<start-line> / span <value>;**

**}**

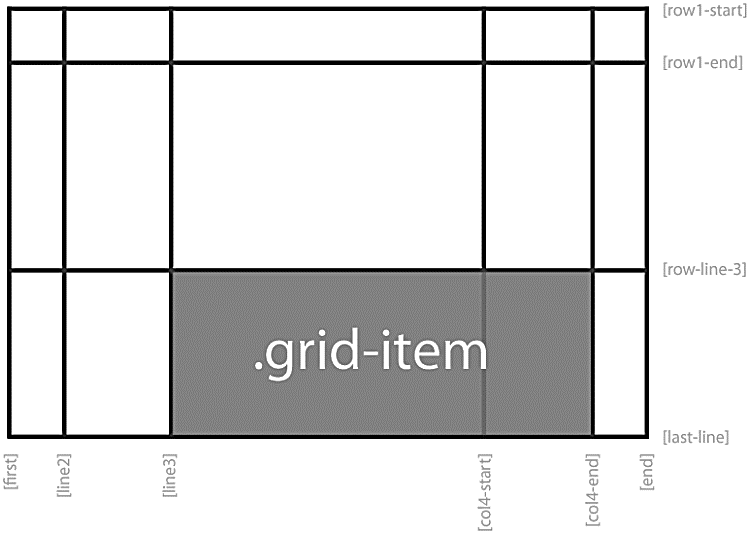
Օրինակ 17.

.grid-item {

grid-row: row-line-3 / 4;

grid-column: 3 / span 2;

}



Եթե grid տարրի վերջնակետը հայտարարված չէ, ապա լռելյայն grid տարրը կզբաղեցնի մեկ grid ուղի:

## Grid-area

Grid-area  հատկությունը օգտագործելով՝ կարելի է անուններ տալ grid տարրերին, որպեսզի դրանցից հետո հղում իրականացվի դրանց վրա grid-template-areas հատկության միջոցով ստեղծված ձևանմուշից:

Այս հատկությունը կարող է օգտագործվել նաև որպես grid-row-start + grid-column-start + grid-row-end + grid-column-end հատկություն-ների պարամետրերի ավելի կարճ գրառման միջոց:

Արժեքները.

* **<name>** — անուն,
* **<row-start> / <column-start> / <row-end> / <column-end>** - grid գծերի համարներ կամ անուններ:

Կիրառման սխեման.

.grid-item {

**grid-area: <name>** կամ

**<row-start> / <column-start> / <row-end> / <column-end>;**

}

Օրինակ 18.

Grid տարրին անվանում վերագրել․

.grid-item {

grid-area: header;

}

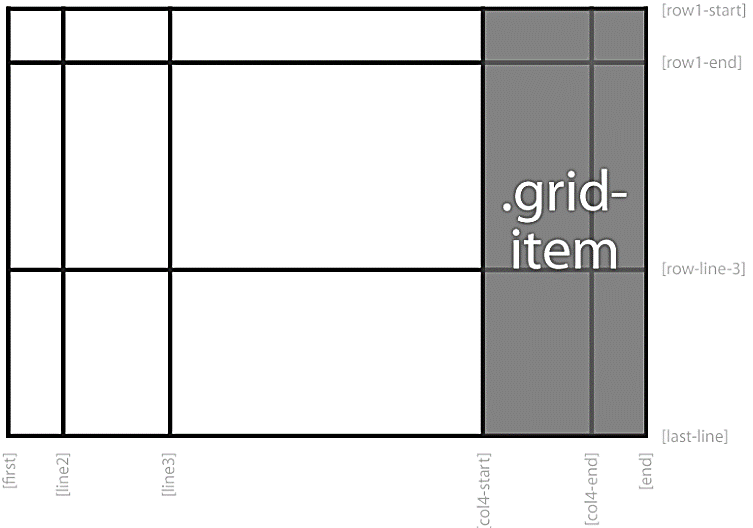
Օրինակ 19.

Օգտագործել հատկությունը որպես grid-row-start, grid-column-start, grid-row-end և grid-column-end հատկությունների արժեքների կարճ գրառում։

.grid-item {

grid-area: 1 / col4-start / last-line / 6;

}



## Justify-self

Justify-self հատկությունը հավասարեցնում է grid տարրի բովանդակությունը տողի առանցքի երկայնքով: Այս հատկությունը տարածվում է մեկ grid տարրի բովանդակության վրա:

Արժեքները.

* **stretch**  (լռելյայն արժեք) - բովանդակությունը լրացնում է grid տարրի ամբողջ լայնությունը,
* **start**  - բովանդակությունը սեղմվում է grid տարրի ձախ եզրին,
* **end**  - բովանդակությունը սեղմվում է grid տարրի աջ եզրին,
* **center**  - բովանդակությունը հավասարեցվում է ըստ grid տարրի կենտրոնի:

Օրինակ 19.

|  |  |
| --- | --- |
| .grid-item {  justify-self: stretch;  } |  |
| .grid-item {  justify-self: start;  } |  |
| .grid-item {  justify-self: end;  } |  |
| .grid-item {  justify-self: center;  } |  |

## Align-self

Align-self հատկությունը հավասարեցնում է grid տարրի բովանդակու-թյունը սյան առանցքի երկայնքով: Այս հատկությունը տարածվում է մեկ grid տարրի բովանդակության վրա:

Արժեքները.

* **stretch**  (լռելյայն արժեք) - բովանդակությունը լրացնում է grid տարրի ամբողջ լայնությունը,
* **start**  - բովանդակությունը սեղմվում է grid տարրի վերին եզրին,
* **end**  - բովանդակությունը սեղմվում է grid տարրի ստորին եզրին,
* **center**  - բովանդակությունը հավասարեցվում է ըստ grid տարրի կենտրոնի:

Օրինակ 20.

|  |  |
| --- | --- |
| .grid-item {  align-self: stretch;  } |  |
| .grid-item {  align-self: start;  } |  |
| .grid-item {  align-self: end;  } |  |
| .grid-item {  align-self: center;  } |  |