https://flexboxfroggy.com/

Դաս 9 - Grid

CSS Grid Layout-ը դասավորության համակարգ է, որը հնարավորություն է տալիս երկու ուղղություններով (տողեր և սյուներ) ձևավորել ցանց: Ի տարբերություն Flexbox-ի, որը միակողմանի համակարգ է, Grid-ը երկկողմանի է և առավել ճկուն է մեծ նախագծերի համար:

```
.container {
  display: grid;
  grid-template-columns: 200px 1fr 200px;
  grid-template-rows: 100px auto 100px;
}
```

Օրինակ հին ժամանակներում կայքերը պատրաստելիս օգտագործում էին աղյուսակները՝ head, body, sidebar, footer -ը տեղաբաշխելու համար, հետո օգտագործում էին float: left, right, հետո **flex-ները**, մասնավորապես bootstrap4 -ը դա է օգտագործում, իսկ արդեն flex-ից հետո ստեղծվել է grid -ը որը բուն արդեն ցանցի համար է ստեղծված: Սկսած 2019թ -ից գրեթե բոլոր բրաուզերները արդեն ունակ են աշխատել grid -ի հետ: Սակայն դեռ մինչ այսօր կան որոշ բրաուզերներ, որոնք չեն կարողանում օգտագործել **Grid**: Բրաուզերների ցուցակը տեսնելու համար կարող ենք օգտվել https://caniuse.com/ կայքից, որոնման դաշտում գրելով grid:
Քանի որ հեռախոսի էկրանները փոքր են և մեծամասամբ հեռախոսի բրաուզերներում օբյեկտները դասավորվում են մեկը մյուսից հետո, այսինքն բլոկային տարբերակով, օգտոգործվում է բլոկային տեգեր:
Տերմինաբանություն՝
Կոնտեյներ (container) - դա որոշ պարունակություն է, որը ունի display: grid հատկություն: Կոնտեյները ունի որոշ քանակությամբ տողեր, սյուներ և թույլ է տալիս կառավարել իր պարունակության վարքագիծը: Ի տարբերություն **Flex**-ի եթե մենք ցանկանանք տարբեր ոճավորումներ տանք տողերին, անհրաժեշտ չէ ստեղխել առանձին կոնտեյներներ տողերի համար:
Պարունակություն (Item) - Կոնտեյների դուստր էլեմենտներն են: Item-ները կարող են նույնպես ունենալ grid որոշ հատկություններ, որոնք կարող են ազդել իրենց դիրքավորման վրա:

```
    <style>
        div>div {
            width: 150px;
            height: 150px;
            border: 1px solid black;
            border-radius: 15px;
            background-color: gray;
        }
        .container {
            display: grid;
            grid-template-columns: repeat(4, 150px);
            grid-gap: 10px;
        }
```

```
    </style>
<body>
    <div class="container">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
        <div class="item"></div>
    </div>
</body>
```

Grid-line - հիմնական կոնցեպտն է, որի շնորհիվ կառուցվելու է մեր հիմնական Grid-հատկությունները: Արտաքնապես կարծես մեր Grid-ը աղյուսակ է, սակայն իրչականում դա այդպես չէ, այն հատուկ ցանց է: Grid-line -ը պետք է կարողանանք պատկերացնել: Օրինակ վերևի օրինակում մենք ունենք 4 ուղղահայաց grid-line և 3 հորիզոնական:
Grid-Cell - **Grid-line** -ի արանքը ընգած տարածքն է: Այն պարտադիր չէ, որ պետք է լինի **Grid-item**, քանի որ կարող է լինել դեպքեր, որ այն մնա դատարկ, կամ էլ մեկ **Grid-cell** -ում տեղակայված լինի մեկից ավելի **Grid-item**-ներ:
Grid-Track - Տիրույթ է որոշ երկու զուգահեռ grid-line -րի արանքում, պարզ լեզվով ասած այն տող է կամ սյուն:

```
.yellow {
            background-color: yellow;
        }
    </style>
</head>

<body>
    <div class="container">
        <div class="item"></div>
        <div class="item"></div>
        <div class="item "></div>
        <div class="item yellow"></div>
        <div class="item yellow"></div>
        <div class="item yellow"></div>
    </div>
</body>
```

Grid-range - Գրիդ տիրույթ: Տիրույթները կարելի է անվարկել, ու հետագայում իրմել տիրույթներին իրենց անուններով և այլն: Դրա մասին իհարկե կծանոթանանք:

```
<body>
    <div class="container">
        <div class="item yellow"></div>
        <div class="item yellow"></div>
        <div class="item "></div>
        <div class="item yellow"></div>
```

```
        <div class="item yellow"></div>
        <div class="item "></div>
    </div>
</body>
```

Ամփոփենք՝ ընդհանուր GRID տերմինները որ մենք թվարկեցինք 6 էին՝ **Կոնտեյներ (container) , Պարունակություն (Item), Grid-line, Grid-Cell, Grid-Track և Grid-range:**

grid- ի հասարակ օրինակ օգտագործելով միայն մեկ կլասս

```
        .container {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
            gap: 1rem;
        }
```

Fr - fraction, доля, բաժին

```
        .container > div {
            min-width: 150px;
            min-height: 150px;
            border: 1px solid black;
            background-color: gray;
        }
```

```
        .container {
            display: grid;
            gap: 1rem;
        }
```

Այժմ փորձենք փոխազդել grid-ի հետ item-ների միջոցով՝

```
.container > div {
            min-width: 150px;
            min-height: 150px;
            border: 1px solid black;
            background-color: gray;
        }

        .container {
            display: grid;
            gap: 1rem;
        }

        .item1 {
            grid-column: 1 / 2;
        }

        .item2 {
            grid-column: 2 / 3;
        }
```

Մեր փոփոխությունները ավտոմատ փոխազդել են նաև այն item-ների վրա, որոնք մենք չենք նշել: Դիմելով grid-column 1/2, 2/3 մենք կարծես թե նշել ենք որ մեր ցանցը պետք է ունենա ուղղահայաց 3 grid-line: Կարող ենք շարունակել, տալով item3 -ին **grid-column: 3 / 4**;
Մենք դեռ տալիս ենք հատկություններ միայն grid-item -ներին: Եկեք ստեղծենք ևս մեկ grid-container:

```
<div class="container">
        <div class="item5"></div>
        <div class="item6"></div>
        <div class="item7"></div>
    </div>
```

Այժմ մենք աշխատում ենք ինչպես աղյուսակների հետ: Աղյուսակները ինչպես հիշում ենք կարող ենք այնպես անել, որ երկու կամ ավելի բջիջ միավորվեն ինչպես մեկը: Նմանատիպ ֆունկցիոնալ կա նաև grid -ում:

```
.item5 {
            grid-column: 1 / 2;
            grid-row: 1 / 3;
        }
        .item6 {
            grid-column: 2/3;
            grid-row: 1/2;
        }
```

Ինչպես տեսնում ենք մենք գրել ենք ընդամենը 2 style բայց արդեն մեր աղյուսակը ստացել է այլ տեսք: Մեր item5 -ը փաստորեն զբաղեցնում է 1-ից 2-րդ սյուները և 1-ից 3-րդ տողերը: Նման հայտարարման մեթոդով հեշտ է նաև փոփոխելու item-ների հերթականությունը, ենթադրենք եթե մենք ցանկանանք, որ item5 -ը նախորդի item5-ին:
Մենք օգտագործում ենք կոմբինացված տարբերակը, սակայն կարող էինք օգտագործել՝ **grid-row-start: 1**; **grid-row-end: 3**; սակայն առաջին տարբերակով հայտարարելը ավելիօ հարմար և կարճ է:

Կարող ենք նաև հստակ չսահմանել ավարտը, օրինակ՝

```
.item:first-child {
            grid-column: 2 / span 2;
        }
```

Սկսվում է 2-րդ ից և զբաղեցնում 2 սյուն

Ենթադրենք ցանկանում ենք պրիմիտիվ կայքի կարկաս հավաքենք՝

```
<body>
    <div class="container">
        <header>
            <h1>Header</h1>
        </header>

        <article>
            <h2>Title</h2>
            <p>Lorem500</p>
```

```
        </article>
        <aside>
            <h3>Mi ban</h3>
            <blockquote> <!-- цитата-->
                Inch vor meki gracy
            </blockquote>
        </aside>
    </div>
</body>
```

```
<style>
        header {
            background-color: gold;
        }
        article {
            background-color: aqua;
        }
        aside {
            background-color: bisque;
        }
    </style>
```

Օրինակ հեռախոսային տարբերակների համար գրեթե արդեն պատրաստ է, քանի որ փոքր էկրանների դեպքում հիշում ենք որ անհրաժեշտ է կիրառել բլոկային տարբերակով: Ուղղակի կարող ենք ավելացնել՝

```
.container {
            display: grid;
            gap: 10px;
        }
```

```
header {
            background-color: gold;
            grid-column: 1/3;
            grid-row: 1/2;
        }
```

Կարող ենք նաև տեղափոխել

```
        article {
            background-color: aqua;
            grid-column: 2/3;
            grid-row: 2/3;
        }
```

```
div.item$*5
grid-template-columns:
grid-template-rows:
```

Սյուների համար շաբլոն սարքելիս՝

```
.item {
            min-width: 100px;
            min-height: 100px;
            background-color: blue;
        }
        .container {
            display: grid;
            gap: 10px;
            grid-template-columns: 200px 500px 150px;
        }
```

Տողերի համար սարքելիս՝ եթե տողերի քանակը շատ է մեր թվարկածից, ապա այն կընդունի լռելյայն արժեքը

```
.container {
            display: grid;
            gap: 10px;
            grid-template-columns: 200px 500px;
            grid-template-rows: 200px 500px;
        }
```

Եթե նույն չափը պետք է կրկնել մի քանի անգամ կարող ենք օգտագործել repeat՝

```
grid-template-columns: repeat(3, 150px);
```

Եթե ցանկանում եմ հայտարարել միայն առաջին և վերջին item-ի չափը՝

```
grid-template-columns: 150px 1fr 150px;
```

Ֆրակցիան զբաղեցնում է ազատ տիրույթը: Այն համագործաքցում է այլ ֆրակցիաների հետ, օրինակ՝

```
grid-template-columns: 2fr 1fr 150px;
```

Ստանում ենք հարաբերակցություն, իր ֆունկցիոնալով նման է %-ային արժեքներին:

Սովորաբար ֆրակցիաները տողերի համար չեն տալիս այլ օգտագործում են **auto**:

Նաև gap -ին կարող ենք տալ երկու արժեք, օրինակ ինչպես margin, padding -ին՝

```
gap: 10px 25px;
```

Flex-ում գիտենք որ կա **order**, նույն հատկությունը կա նաև **gap** -ում:
Դատարկ տիրույթներ

```
<div class="container">
        <header>header</header>
        <article>
            <h2>Vernagir</h2>
            <p>lorem 150</p>
        </article>
        <aside>
            mi ban
        </aside>
        <footer>
```

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

```
<style>
        header {
            background-color: gray;
        }
        article {
            background-color: pink;
        }
        aside {
            background-color: red;
        }
        footer {
            background-color: aquamarine;
        }
    </style>
```

Տանք Grid հատկություն՝

```
.container {
            display: grid;
            gap: 10px;
        }
```

Այժմ ուսումնասիրենք

```
            grid-template-areas: none;
```

Եթե այն չենք օգտագործում, ապա իր արժեքը հավասար է none

```
header {
            background-color: gray;
            grid-area: header;
        }
        article {
            background-color: pink;
            grid-area: article;
        }
        aside {
            background-color: red;
            grid-area: aside;
        }
        footer {
            background-color: aquamarine;
            grid-area: footer;
        }
```

Կարծես թե ամեն ինչ փչացավ, բայց դա նրանից է, որ՝

```
.container {
            display: grid;
            gap: 10px;
            grid-template-columns: repeat(3, 1fr);
            grid-template-areas:
```

```
        "header"
        "article"
        "aside"
        "footer";
        }
```

կարել ի է սահմանել նաև այսպես՝

```
grid-template-areas:
                "header header header"
                "article article aside"
                "footer footer footer";
```

Այժմ սարքենք դատարկ տիրույթ՝ օգտագործելով կետը

```
.container {
            display: grid;
            gap: 10px;
            grid-template-columns: repeat(4, 1fr);
            grid-template-areas:
                "header header header header"
                "article article . aside"
                "footer footer footer footer";
        }
```

Կա նաև ավելի հեշտացված գրելու մեթոդ՝

```
grid-template: none |  grid-template-rows | grid-template-columns
```

```
grid-template: "header header header header" 50px
                "article article . aside"
                "footer footer footer footer" 50px / 1fr 1fr 50px 1fr;
```

Սակայն այն օգտագործվում է եթե չունենք area -ներ

Այժմ ուսումնասիրենք հավասարեցումները՝

```
header {
            background-color: gray;
        }

        article {
            background-color: pink;
        }

        aside {
            background-color: red;
        }

        footer {
```

```
            background-color: aquamarine;
        }

        .container {
            display: grid;
            gap: 10px;
            grid-template: auto / 1fr 1fr 1fr; (auto կամ ասեմ 500px 300px)
        }
```

Ու կարող ենք օգտագործել justify-items ինչպես դա արել ենք flex-ում:

```
justify-items: start;
justify-items: center;
justify-items: end;
justify-items: stretch; // լռելյայն տարբերակ
```

Նույնը կա նաև align-items -ի համար:

Կա նաև կոմբինացված տարբերակը

```
place-items: center center;
```

Այժմ մենք չունենք սահմանաձակում ներ կոնտեյների լայնության կամ երկարության համար, ենթադրենք մեզ դա անհրաժեշտ է՝

```
.container {
            display: grid;
            width: 700px;
            gap: 10px;
            grid-template: auto / 100px 250px 120px;
            border: 1px solid black;
            place-items: center center;
        }
```

Այժմ ունենք հավելյալ տարածք որը չի օգտագործվում:

```
.container {
            display: grid;
            width: 700px;
            gap: 10px;
            grid-template: auto / 100px 250px 120px;
            border: 1px solid black;
            place-items: center center;
        }
```

Ու ցանկանում ենք դիրքավորել մեր ցանցը օգտագործելով ազատ տիրույթը, նման պարագայում մեզ օգնության է գալիս արդեն justify-content

```
justify-content: space-between;
```

Նույն եղանակով նաև ուղղահայաց դիրքավորման համար՝

```
align-content: start;
```

Կա նաև կոմբինացված տարբերակը՝ սկզբից align-content, justify-content

```
place-content: center space-evenly;
```

Ֆլեքսից հիշում ենք նաև որ կարող ենք հայտարարել անհատական հատկություններ կոնկրետ item-ներին: Այստեղ նույնպես կարող ենք տալ justify-self, align-self

```
aside {
            background-color: red;
            align-self: start;
        }
```

Սակայն Grid-ով աշխատելիս խորհուրդ չի տրվում օգտագործել մնան եղանակով ֆիքսված չափսեր, քանի որ բուն Grid-ի իմաստը կորում է:

Ադապտիվ Grid
Ունենք՝

```
<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">5</div>
        <div class="item item6">6</div>
        <div class="item item7">7</div>
    </div>
```

```
<style>
        html, body {
            margin: 0;
        }
        .container {
            background-color: gainsboro;
            padding: 1.5rem;
            display: grid;
            gap: 10px 15px;
        }

        .item {
            background-color: orange;
            border-radius: 15px;
            padding: 1rem;
        }
    </style>
```

Սահմանում ենք մինիմալ և մաքսիմալ չափերը, կարող ենք սահմանել պիքսելներով, ֆրակցիաներով

```
.container {
            background-color: gainsboro;
            padding: 1.5rem;
            display: grid;
```

```
            gap: 10px 15px;

            grid-template-columns: 200px 1fr minmax(100px, 200px);
        }
```

Օրինակ եթե շատ մեծ տողեր ունենանք, ապա այն կարող է սահմանից դուրս գալ, դա բացառելու համար օգտագործում ենք՝

```
grid-auto-rows: minmax(100px, auto);
```

Այն սահմանում է բոլոր նոր տողերի համար մինիմալ 100 պիկսել

Օրինակ ենթադրենք մեզ պետք է որպեսզի մեր ֆուտերը եթե կոնտենտը քիչ է կպած լինի ներքևի հատվածին՝

```
        <div class="item item1">1</div>
        <div class="item item2">2</div>
        <div class="item item3">3</div>
```

```
.container {
            background-color: gainsboro;
            padding: 1.5rem;
            display: grid;
            gap: 10px 15px;
            grid-template-rows: 100px minmax(100px, 1fr) 50px;
```

Սա հիմա չի աշխատում քանի որ մենք դիվ ենք հայտարարել ու նրա բարձրությունը գիտենք որ հավասար է իր պարունակության բարձրությանը եթե սահմանված չի, այդ իսկ դեպքում պետք է տանք նաև height: 100vh; որպեսզի բարձրություն սահմանենք: Այսինքն եթե grid container -ը չունի սահմանված բարձրություն այն եղանակը աշխատելու է որպես auto:

```
            height: 100vh;
            /* padding: 1.5rem; */
```

Հիշում ենք նաև որ flex-ում մենք կարող էինք փոփոխել ուղղությունը, այսինքն սահմանել row, column: Մնամ ֆունկցիոնալ կա նաև այստեղ՝

```
display: grid;
            gap: 10px 15px;
            grid-template-columns: 200px 1fr minmax(100px, 200px);
            grid-template-rows: minmax(100px, auto) 100px 200px;
            grid-auto-flow: row;
```

Վերադառնանք մեր սյունակների ադապտիվ ցուցադրման օրինակին՝

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

Ինքը կրկնում եր մեր փոխանցված մեծքությունը նշված քանակի անգամ: Բացի փոխանցվող կրկնությունների քանակը մենք կարող ենք փոխանցել նաև մեթոդը:

```
grid-template-columns: repeat(auto-fit, minmax(200px, 250px))
```

```
grid-template-columns: repeat(auto-fill, minmax(200px, 250px))
```

Auto-fit -ի տրամանաբությունը՝

```
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr))
```

Ինքը փորձում է զբաղեցնել ողջ ազատ տիրույթը՝ մեծացնելով մեր item-ների չափերը:

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))
```

Auto-fill -ի տրամանաբությունը՝
Ինքը տեսնում է թե քանի հատ կարելի է լրացնել մաքսիմալ տվյալ տողում՝

```
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr))
```

Մենք չենք ուսումնասիրենք grid հատկությունը, այն կատարում է նույն գործողությունները ինչ grid-template:
Մենք կարող ենք տալ անհատական վարքագիծ ցանկացած grid-item ներին: Տալ վարքագիծ ընդհանուր grid-container -ին:
[Responsive Periodic Table with CSS Grids](#)
[index.html - nodebox - CodeSandbox](#)