



IT-TALENTS ASSOCIATION



CASCADING STYLE SHEET

CSS



CSS (Cascading Style Sheet)-ը օգտագործվում է մեր պատրաստած HTML ֆայլի էլեմենտներին style տալու համար:

Style-ի նկարագրությունը կազմված է selector-ներից (selector-ների միջոցով մենք ընտրում ենք այն էլեմենտը, որին ցանկանում ենք style տալ) և style-ի նկարագրության բաժնից, որը գրվում է {} փակագծերի մեջ, այն ունի հետևյալ տեսքը՝

```
p{  
    font-family:Arial,Verdana,sans-serif;  
    color:blue;  
}
```

Ամեն մի style-ի հատկանիշից հետո դնում ենք կետ, ստորակետ (:)

CSS-ը մեր HTML ֆայլում հասանելի դարձնելու համար կա երեք եղանակ՝

- inline
- internal
- external

Inline սա այն եղանակն է, երբ մենք մեր css կոդը գրում ենք անմիջապես թեգի ներսում, որը վերաբերում է կոնկրետ այն ատրիբուտին, որում գրված է style-ը:



Inline style գրելու համար ցանկացած թեգի մեջ բացում ենք `style=""`; ատրիբուտը և գրում նրա հատկությունները, օրինակ՝

```
<h1 style="color:blue;">Վերևագիր h1 չափ</h1>
```

Internal տարբերակով style գրելու համար մեր HTML ֆայլի ցանկացած տողում բացում ենք `<style></style>-` թեգը և անմիջապես այս թեգի մեջ գրում մեր css կոդը, օրինակ՝

```
<style>
  h1{
    color:blue;
  }
</style>
```

Այսպիսով մենք մեր HTML ֆայլում գտնվող բոլոր `h1` թեգերին տալիս ենք կապույտ գույն:

External եղանակը, որը ավելի հարմար է օգտագործման մեջ, այս եղանակի ժամանակ մենք կարող ենք ունենալ օրինակ առանձին style.css ֆայլ, այն միացնում ենք մեր HTML ֆայլին և առանձին style.css ֆայլի մեջ գրում ենք CSS կոդը, ինչպես ներկայացված է internal տարբերակում: Մեր HTML ֆայլին մենք կարող ենք անսահմանափակ քանակությամբ style միացնել: Եթե ուզում ենք մեր երկրորդ CSS ֆայլը միացնել մեր CSS ֆայլին, ապա օգտվում ենք @import url (mobile.css) տարբերակից url-ում նշելով այն CSS ֆայլի ճանապարհը, որը ցանկանում ենք միացնել: Նշում ենք հարաբերական ճանապարհ՝ հաշված տվյալ CSS ֆայլից, որի մեջ ուզում ենք միացնել ուրիշ CSS ֆայլ: @import -ի օգնությամբ կարող ենք միացնել նաև ինչ-որ կայքերից font-ի տարբեր տեսակներ հետևյալ եղանակով՝

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,cyrillic);
```

Էլեմենտը(թեգը) ընտրելու համար օգտվելու ենք selector-ներից:

Առաջին selector-ը էլեմենտի (թեգի) selector-ն է, գրում ենք պարզապես այն թեգի անունը, որին ուզում ենք style տալ: Բացում ենք {} փակագծերը և գրում նրա նկարագրությունը: Վերցնենք բոլոր ք թեգերը և նրանց տանք կարմիր գույն:

```
p{  
color:red;  
}
```

Եթե ցանկանում ենք վերցնել բոլոր թեգերը, ապա կարող ենք գրել հետևյալ ձևով՝

```
*{  
color:red;  
}
```

Այսպիսով բոլոր Էլեմենտներին տալիս ենք կարմիր գույն:

Հաջորդը class-ով selector-ն է: Այս դեպքում մենք էլեմենտին կարող ենք տալ class="" ատրիբուտը: Եվ CSS-ով նրան տալ style: HTML ֆայլում տեսքը կլինի այսպես՝ `<p class="first">text...</p>`

CSS-ում՝ `.first{
color : red;
}`

Class-ով selector-ը ունի հետևյալ տեսքը՝ դրվում է կետ (.) և անմիջապես նրա կողքին գրվում է այն class-ի անունը, որին ուզում ենք style տալ: Class ատրիբուտը կարող ենք տալ մեր HTML-ում անվերջ քանակությամբ թեգերին, և, որ թեգը ունեցավ class ատրիբուտով էլեմենտը՝ կընդունի այդ class-ով selector-ում գրված style-ը:

Հաջորդը id-ով selector-ն է: Ինչպես հայտնի է HTML-ի դասից, id-ն ունիկալ է և չի կարող կրկնվել, ուստի id-ով selector-ը հարմար չէ օգտագործել CSS-ում:
Տեսքը HTML-ում՝

```
<p id="first">text...</p>
```

Տեսքը CSS-ում՝

```
#first{  
color:red;  
}
```

Հաջորդ selector-ը ներդրված էլեմենտի sector-ն է, որը ունի հետևյալ տեսքը՝

```
ul li{  
color:red;  
}
```

Սա նշանակում է, որ մեր գրած style-ը վերաբերում է HTML ֆայլում գտնվող li թեգերին, որոնք գտնվում են ul թեգերի մեջ՝ պարզապես առանձնացնելով selector-ում գրված թեգերը բացատանիշներով:

Օրինակ՝

```
p.first span{  
color:red;  
}
```

Սա նշանակում է, որ մենք ցանականում ենք ընտրել բոլոր span թեգերը, որոնց ծնողը(սա նշանակում է, որ span թեգը գտնվում է p թեգի մեջ, որը ունի նշված class-ը) պ թեգը first class ատրիբուտով է:

Հաջորդ selector-ը անմիջապես իր մեջ գտնվող էլեմենտի selector-ն է:

Եթե ունենք հետևյալ կոդը HTML-ում

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

Ապա CSS-ում գրելով հետևյալ selector-ը

```
div > img{  
border:1 px solid red;  
}
```

Մենք տալիս ենք եզրագիծ 1 px հաստությամբ, կարմիր գույնի միայն այն img թեգին, որի անմիջապես ծնողն է հանդիսանում div թեգը, իսկ 2-րդ img թեգի ծնողն է հանդիսանում ա թեգը, այդ պատճառով մեր գրած style-ը չի վերաբերում 2-րդ img թեգին:

Հաջորդ selector-ը անմիջապես հաջորդող էլեմենտի selector-ն է:

```
div+p{  
color:red;  
}
```

Սա նշանակում է, որ մենք ընտրում ենք HTML-ում բոլոր div թեգերին անմիջապես հաջորդող p թեգը ուղղակի դնելով + նշանը:

Ատրիբուտով selector՝

```
img[title]{  
Ըստում ենք այն img թեգը, որը ունի title ատրիբուտ:  
}  
img[title="image"]{  
Ըստում ենք այն img թեգը, որը ունի title ատրիբուտ և նրա արժեքը հավասար է "image"  
}  
img[title^="image"]{  
Ըստում ենք այն img թեգը, որը սկզբում է image բառով հավասարից առաջ դնելով ^ նշանը  
}  
img[title$="image"]{  
Ըստում ենք այն img թեգը, որը վերջանում է image բառով դնելով հավասարից առաջ $ նշանը:  
}  
img[title*= "image"]{  
Ըստում ենք այն img թեգը, որը պարունակում է image բառը:
```

Ցանկացած էլեմենտի selector-ի վերջում գրելով :hover- սա նշանակում է, որ մենք ցանկանում ենք նշել տվյալ էլեմենտի style-ը այն ժամանակ, երբ մկնիկի սլաքը պահված կլինի նրա վրա:

:focus - կնշանակի, որ մենք ուզում ենք տալ style այն ժամանակ , երբ browser-ի focus-ը կլինի այդ էլեմենտի վրա:

:checked - դնելով կնշանակի, որ style ենք տալիս այն էլեմենտին, որն ունի checked ատրիբուտ:

Եթե էջում ունենք օրինակ՝

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

Ապա գրելով՝

```
ul li:nth-child(odd){
ընտրում ենք ս 1 թեղի մեջ գտնվող բոլոր կենա 1 i թեգերը
}
```

իսկ՝

```
ul li:nth-child(even){
ընտրում ենք ս 1 թեղի մեջ գտնվող բոլոր զույգ 1 i թեգերը
}
```

- :nth-child(n)--n-ի փոխարեն գրելով ինչ-որ մի թիվ գրում ենք style ո-րդ էլեմենտի համար:
- :first-child գրելով նշում ենք մեր list-ի առաջին էլեմենտը:
- :last-child գրելով նշում ենք մեր list-ի վերջին էլեմենտը:
- :not() գրելով նշում ենք բացառությունները, օրինակ՝ ul li:not(.first) սա նշանակառ է, որ ընտրում ենք սլ-ի մեջ գտնվող բոլոր li թեգերը բացի այն li-ից, որը ունի first-class: Էլեմենտի համար ինչ-որ նոր էլեմենտ ավելացնելու համար գրում ենք՝

```
::before psevodelement@  
li::before{  
    content="";  
}
```

Որի content բաժնում նշում ենք թե ինչ symbol ենք ուզում ավելացնել լի թեգի սկզբի մասից, իսկ եթե ինչ-որ տեքստ կամ symbol ավելացնել, ապա անպայման պետք է գրենք content="" դատարկ չափերտներով:

Նույն կերպով էլեմենտի վերջից ավելացնելու համար կօգտվենք ::after psevdoelement-ից:

Selector-ները խմբավորելու համար, եթե նրանք պետք է ունենան նույն style-ը, ապա գրում ենք selector-ները անջատելով դրանք ստորակետով(,):

```
h1,p.first,ul li:not(.first){  
color:red;  
}
```

Selector-ների ազդեցիկության աղյուսակ

```
id=100 միավոր;  
class=10 միավոր;  
element(p) = 1միավոր;  
style-ը գրելով թեզի աևմիջապես ներսում=1000միավոր;
```

Քանի որ մեկ էլեմենտին կարող ենք տարբեր ձևերով style տալ, ապա այդ էլեմենտի վրա ազդելու է այն style-ը, որի ազդեցիկությունը ամենամեծն է: Օրինակ՝

```
1) ul li.first{  
color:blue(կապույտ գույն);  
}  
2) li#second{  
color:red(կարմիր գույն);  
}
```

1-ին դեպքում selector-ի միավորը կազմում է `ul(1) li(1) .first(10)` == $1 + 1 + 1 = 12$ միավոր, իսկ 2-րդում `li(1) #second(100)` == $100 + 1 = 101$ միավոր: Քանի որ 2-րդ դեպքում ստացված միավորը ավելի մեծ է, ուրեմն կաշխատի 2-րդ դեպքում գրված selector-ը:

Էլեմենտները լինում են երկու տեսակ՝ inline և block:

block-ային էլեմենտները զբաղեցնում են ամբողջ տողը, իսկ inline էլեմենտները այն տարածքը, որըան անհրաժեշտ է իրենց: Էլեմենտի տիպը փոխելու համար selector-ում գրում ենք՝ **display:(inline,block);** այս երկու տարբերակներից մեկը: Էլեմենտին ներսից տարածություն տալու համար գրում ենք՝

```
padding  
padding-top:10px; վերևի տարածություն  
padding-bottom:5px; ներքևի տարածություն  
padding-left:3px; ձախ կողմից տարածություն  
padding-right:20px; աջ կողմից տարածություն
```

Կա ավելի կարճ գրելաձն.

```
padding:10px 20px 5px 3px;
```

Հերթականությունը նշելով ժամսլաքի ուղղությամբ՝ սկսած վերևից:

Իսկ էլեմենտին դրսի կողմից տարածություն տալու համար գրում ենք՝

```
margin-top:10px;  
margin-bottom:5px;  
margin-left:3px;  
margin-right:20px;
```

Նույն ձևով նշվում է նաև կարճ տարբերակը.

```
margin:10px 20px 5px 3px;
```

Իսկ էլեմենտին եզրագիծ տալու համար գրում ենք:

Վերևի եզրագծի համար.

```
border-top-width:2px;  
border-top-style:dotted;  
border-top-color:red;
```

Կարճ ձևը.

```
border-top:2px dotted red;
```

Աջ եզրագծի համար.

```
border-right-width:2px;  
border-right-style:dotted;  
border-right-color:red;
```

Կարճ ձևը.

```
border-right:2px dotted red;
```

Ներքսի եզրագծի համար.

```
border-bottom-width:2px;  
border-bottom-style:dotted;  
border-bottom-color:red;
```

Կարճ ձևը.

```
border-bottom:2px dotted red;
```

Զախ եզրագծի համար.

```
border-left-width:2px;  
border-left-style:dotted;  
border-left-color:red;
```

Կարճ ձևը.

```
border-left:2px dotted red;
```

Իսկ այս բոլորի կարճ տարբերակը.

```
border:1px solid red;
```

Որտեղ առաջինը (border width) հաստությունն է, 2-րդը՝ տիպը
(dotted,dashed,solid), 3-րդը՝ գույնը, որ վերաբերելու են 4-րդ կողմի եզրագծերին:

Position

Էլեմենտին երբ մենք տալիս ենք position, կարողանում ենք կառավարել նրա դիրքը ըստ էկրանի

position:absalute-ի դեպքում էլեմենտը դառնում է անկախ բոլոր էլեմենտներից, մի մակարդակ բարձրանում է վերև և մյուս էլեմենտներին զիջում է իր տեղը:

position:relative-ի դեպքում էլեմենտը դառնում է անկախ, բայց չի զիջում իր տեղը մյուս էլեմենտներին:

position:fixed-ի դեպքում էլեմենտը դառնում է անկախ, զիջում է իր տեղը և մնում ֆիքսված անգամ scroll անելու դեպքում:

Եթե էլեմենտի ծնող էլեմենտին նույնական տանք ինչ-որ position, ապա նրա մեջ գտնվող էլեմենտը կտեղաշարժվի արդեն հաշված և ծնող էլեմենտի սկզբնակետից: Այսինքն կառավարում ենք կոորդինատները հաշված ծնողից:

Float պարամետրը տալու դեպքում նշելով right, left էլեմենտին դարձնում ենք անկախ և նրան տանում էկրանի աջ կամ ձախ կողմ: Օրինակ՝

```
img{  
float:right;  
}
```

Եթե ունենանք երկու նկար, մեկին տանք float left, մյուսին՝ float right, ապա նրանից հետո գտնվող տեքստը կգա և կզբաղեցնի այդ երկու նկարների միջև տարածությունը:

Այդ դեպքում տեքստին տալով `clear:both;` արժեքը տեքստը կազատի այդ ազատ տարածությունը: Տեքստի դիրքը կառավարելու համար կօգտագործենք տեքստի `align` պարամետրը:

```
p{  
text-align:(left,right,center)  
}
```

Դպրոցից հայտնի է մի մատ խորքից արտահայտությունը. CSS-ում մի մատ խորքից գրելու համար.

```
p{  
text-indent:20px(նշելով չափը px-ներով);  
}
```

`line-height:20px;` - տալով նշում ենք տեքստի տողերի միջև եղած հեռավորությունը:
`vertical-align:center(top,bottom);` - նշում ենք մեր էլեմենտի հավասարեցումը ուղղահայաց ուղղությամբ:



Word-spacing:10px; - պարամետրով նշում ենք տեքստի բառերի միջև հեռավորությունը, իսկ letter-spacing:10px; - ով տառերի միջև հեռավորությունը: Տեքստի ներքեւից, վերևից կամ տեքստի վրա գիծ դնելու համար օգտագործում ենք.
text-decoration:

underline (ներքեւի մասում),
overline (վերևի մասում),
line-through (տեքստի վրա);

Տեքստի ուղղությունը նշելու համար՝ direction:rtl(right to left);(ltr)(left to right); Տեքստը մեծատառերով գրելու համար՝ text-transform: (uppercase (բոլոր տառերը
մեծատառ), lowercase (բոլոր տառերը փոքրատառ), միայն առաջին տառերը մեծատառով գրելու համար (capitalize)):

font-ը նշելու համար՝
font-family: և գորում ենք font-ի անունը, կարող ենք գտնել մի քանի անուն մեկը չգտնելու դեպքում, մյուսն օգտագործելու համար:

font-style:italic - տեքստը շեղ գրելու համար:

font-variant:small-caps; - տեքստը մեծատառերով, բայց փոքր գրելու համար:

font-weight:bold: - տեքստը հաստ տառերով գրելու համար:

font-size:10px; - տառաչափը նշելու համար:

color:red(blue,#000000,rgb(255,255,255),rgba(0,0,0,0.5)) - գույշը տարբեր եղանակներով նշելու համար:

Աթեգի pseudo-class-ներ են կոչվում աթեգի տարբեր ժամանակներում, տարբեր պահելաձևերը:

Եթե ունենք աթեգ՝

```
a:hover{  
որոշում է թեգի տեսքը, այն ժամանակ երբ մկնիկի կուրսորը պահված է այդ թեգի վրա  
}  
a:active{  
որոշում է աթեգի տեսքը այն ժամանակ, երբ click-ը սեղմված է աթեգի վրա  
}  
a:visited{  
եթե մենք թեգի վրա մի անգամ սեղմել ենք թեգի տեսքը կլինի այս selector-ում գրված տեսքով  
}  
a:link{  
սա աթեգի սկզբնական տեսքի համար է նախատեսված  
}
```

Մկնիկի կուրսորը փոխելու համար՝ cursor: pointer, help, resize.

Եթե ունենք table թեգ և նրա մեջ գտնվող բոլոր էլեմենտների համար ունենք style, օրինակ՝ **border:1px solid red;**

ապա այդ եզրագծերը միացնելու համար օգտագործում ենք՝ **border-collapse: collapse;**

պարամետրը, իսկ եզրագծերի միջև տարածության համար՝ **border-spacing: 20px;**

Նշելով չափը ul-ի li թեգերի դիմաց դրված նշումները փոխելու

համար **list-style-type:square(քառակուսի), circle(շրջան), Armenian;**

Իսկ այդ նշումների փոխարեն նկար տեղադրելու համար՝

```
list-style-image:url(նշելով նկարի հասցեն);
```

Այդ մարկերների դիրքը փոխելու համար՝

```
list-style-position:(inside,outside); Inside - ներսի կողմից, outside - դրսի կողմից
```

Հետևի ֆոնին գույն տալու համար՝

```
background-color:red;
```

Ֆոնի նկար տեղադրելու համար՝

```
background-image:url(նկարի հասցեն);
```

Նկարի կրկնվելու համար՝

```
background-repeat:repeat(կրկնվում է ամբողջ մակերեսով),  
no-repeat(չի կրկնվում),repeat-x(կրկնվում է x-երի առանցքով),  
repeat-y(կրկնվում է y-ների առանցքով)
```

Background-position:(x,y) -x-ի և y-ի փոխարեն նշելով թվերը պիքսելներով, նշում ենք նկարի դիրքը ըստ տրված բլոկի, որի մեջ դրված է նկարը. x-ի, y-ի փոխարեն կարող ենք նշել նաև 'center, (left,right,bottom,top):'

Նկարը ֆիքսված պահելու համար՝

```
background-attachment:fixed; և նկարի scroll-ի ժամանակ ֆիքսված է մնում:
```

Նկարի դիրքը փոխելու համար՝ ըստ տրված բլոկի

`background-clip:border-box,content-box,padding-box;`

1-ին դեպքում կմնա նկարի այն հատվածը, որը գտնվում է նկարի border-ից ներս,
2-րդ դեպքում նկարի այն հատվածը, որը գտնվում է padding-ից ներս, իսկ 3-րդ
դեպքում այն հատվածը, որը content-blok-ի մեջ է:

`background-size:contain(cover)(50px)`

Նշում ենք նկարի չափը cover-նկարը ձգելով ամբողջ մակերեսով:

contain-եթե նկարի ինչ-որ մի կողմը հասնում է border-ին, ապա նկարի չափը մնում է
այդ չափով, եթե ունենք տեքստ և այդ տեքստի բլոկի լայնությունը փոքր է, և
տեքստը չի տեղավորվում, ապա այդ տեքստին կարող ենք

տալ `text-overflow:ellipsis;` և այն հատվածի փոխարեն, որը չի
տեղավորվել ավտոմատ կտեղադրվեն 3 կետեր (...)

Այժմ կծանոթանանք gradient-ների հետ: Gradient-ը ինչ-որ տարբեր գույներից ստացված պատկերն է, այս լինում է CSS-ում երկու տեսակ՝ linear(գծային) և radial(շրջանի տեսքով):

```
div{  
background:linear-gradient(red,blue);  
}
```

Նրա պարամետրերի մեջ կարող ենք նշել նաև ուղղությունը.

```
:linear-gradient(to left red,blue);
```

Կարող ենք նշել նաև անվերջ քանակությամբ գույներ.

```
linear-gradient(to right top red,blue,green);
```

Ամեն գույնից հետո %-%ներով կարող ենք նշել թե քանի տոկոս տարածք է զբաղեցնելու այդ գույնը՝ մի գույնից մյուսի անցումը ավելի տեսանելի դարձնելու համար:

Radial-gradient()-ը ունի նույն հատկությունները ինչ-որ նախորդը:

Կրկնվող gradient պատրաստելու համար gradient-ի սկզբից ավելացնում ենք repeating բառը:

```
Repeating-linear-gradient();  
Repeating-radial-gradient();
```

Block-ի եզրերը կլորացնելու համար կօգտագործենք՝

border-radius:15px(20%);

Նշելով կլորացման չափսը px-ներով կամ %-ներով:

Տեքստին ստվեր տալու համար օգտագործում ենք՝

text-shadow:(x,y,z,k);

- Y - Ստվերի դիրքը y-ների առանցքով
- X - Ստվերի դիրքը x-երի առանցքով
- Z - Ստվերի ցրվածություն
- K - Ստվերի չափի մեծացում

Անհմացիաները գեղեցկացնելու համար կարող ենք այն էլեմենտներին, որոնք անհմացիայի են ենթարկվում տալ transition:

Գրելածնի կարճ տարբերակն է, օրինակ՝

```
a{  
    color:red;  
    transition:0.5s all;  
}  
a:hover{  
    color:blue;  
}
```

- Transition-property - նշում ենք այն css պարամետրի անունը, որը գեղեցիկ անհմացիա ենք ցանկանում սարքել: Օրինակ width կամ all արժեքը նշելու դեպքում անհմացիայի կենթարկվեն այն պարամետրերը, որոնց արժեքները փոփոխվել են:
- Transition-duration - նշում ենք անհմացիայի ժամանակը:
- Transition-timing-function - նշում ենք անհմացիայի ձևը:
- Transition-delay - նշում ենք անհմացիայի հետաձգման ժամանակը:

Եթե ուզում ենք գրել կարճ ձևով, ապա պետք է պահպանենք այս հերթականությունը՝

```
{ transition: transition-property transition-duration transition-timing-function transition-delay; }
```

Այս պարամետրերից պարտադիր է նշել գոնե transition-property-ն և transition-duration-ը:

Transform - պարամետրը հնարավորություն է տալիս փոփոխել նշված էլեմենտը՝
Transform: translate(), translateX(), translateY(), rotate(), scale(), scaleX(), scaleY(), skew(),
skewX(), skewY()

Transform-origin - հնարավորություն է տալիս փոխել ձևափոխության առանցքը:

Նշում է transform, ապա նշում են այն պարամետրերը, որոնք ուզում ենք փոփոխել:

- translate(x,y) - տեղափոխում է էլեմենտը նշված x և y կոորդինատներով:
- translateX(10px) - տեղափոխում է էլեմենտը x-երի առանցքով 10px-ով:
- translateY(5px) - տեղափոխում է էլեմենտը y-ների առանցքով 5px-ով:
- rotate(20deg)- պտտում է էլեմենտը նշված անկյունում deg (աստիճան):
- scale(x,y), scaleX(10px), scale(5px) - մեծանում է էլեմենտի չափերը նշված առանցքներով չափման միավոր %-ներով կամ թվերով:
scale(0.5),scale(50%)
- scale(2) - Սա նշանակում է, որ մենք ուզում ենք մեծացնել երկու անգամ:
- skew(), skewX(), skewY() - էլեմենտը ձգում է նշված առանցքներով:

Animation

@keyframes

- animation-name: myrot; - անիմացիայի անուն
- animation-duration: 2s; - ժամանակ
- animation-timing-function: linear; անիմացիայի տիա
(ease,ease-in,ease-out,ease-in-out)
- animation-delay: 1s; անիմացիայի ուշացումը
- animation-iteration-count: infinite; - քանի անգամ է կրկնվելու անիմացիան (infinite - անվերջ)
- animation-direction: alternate; - ուղղությունը նորմալ, revers (alternate-անիմացիան 0-ից գնում է 100% հետո հետ է գալիս 0%):

Անհմացիա ստեղծելու համար նախ պետք է սկզբից սահմանել այն և հետո տալ որևէ էլեմենտի:

Անհմացիան գրում են հետևյալ կերպով՝

@keyframes myrot (սա անհմացիայի անունն է)

```
{  
  from{  
    color:red;  
  }  
  to{  
    color:blue;  
  }  
}
```

Կամ ավելի հարմար է % - ներով:

@keyframes myrot (սա անհմացիայի անունն է)

```
{  
  0%{  
    color:red;  
  }  
  25%{  
    color:blue;  
  }  
  50%{  
    color:coral;  
  }  
  100%{  
    color:yellow;  
  }  
}
```

2-րդ եղանակով գրելիս ավելի մանրամասն կարող ենք գրել անհմացիան, իսկ առաջին եղանակով գրելիս կարող ենք գրել միայն էլեմենտի սկզբնական և վերջնական դիրքերի համար, օրինակ գույնի անհմացիան:

Անհմացիան գրելուց հետո նրան տանք ինչ-որ աթեգ:

Սա կարճ գրելածն է, իսկ երկար գրելածնը վերոնշյալ ձևով է գրվում:

```
a{  
    animation:myrot 2s linear 1s infinite alternate;  
}
```

- Column-count: նշում ենք մեր սյուների քանակը, իսկ ազատ տարածությունը բաշխվում է սյուների մեջ հավասար:
- Column-width: նշում ենք մեր սյուների լայնությունը, բայց column-count - ը արդեն չի գործում:
- Column-gap: նշում ենք տարածությունը սյուների միջև:
- Column-rule-style: նշում ենք բաժանարար գծի ոճը:
- Column-rule-width: նշում ենք բաժանարար գծի հաստությունը:
- Column-rule-color: նշում ենք բաժանարար գծի գույնը:
- Column-rule: column-rule-width_ column-rule-style_ column-rule-color

- Perspective: Այս պարամետրով մենք նշում ենք, որ աշխատում ենք 3d տարածությունում:
- Perspective-origin: Նշում ենք կոորդինատների սկզբնակետը perspective - ի համար:
- Backface-visibility: Նշում ենք երևալու է արդյոք էլեմենտների հետևի մասերը (visible,hidden)
- Translate3d(X, Y, Z)-x,y,z - ի փոխարեն նշում ենք, որ առանցքով և ինչքան px-ով ենք ուղղում տեղափոխել մեր էլեմենտը:
- TranslateZ() - այս ձևով նշում ենք միայն z-երի առանցքի համար:

Էլեմենտին filter տալու համար օգտվում ենք filter պարամետրից:

```
a{
  filter:blur(5px)
}
```

- Blur() - խավարեցնում է էլեմենտը:
- Brightness() - լուսավորությունը:
- Contrast() - ապահովում է contrast-ը
- Drop-shadow()- աշխատում է box shadow-ի նման
- Grayscale() - սև և սպիտակ նկար
- Opacity() - թափանցիկությունը
- Saturate() - նշում ենք գույների առատությունը
- Sepia() - retro

Media Query-ները օգտագործվում են մեր կայքը տարբեր Էկրաններում ճիշտ ձևով բացելու համար: Media Query-ները գրվում ն հետևյալ ձևով՝

```
@media only screen and (max-width:1200px){  
    a{  
        color:red;  
    }  
}
```

Սա նշանակում է, որ մինչև 1200px լայնքով Էկրանում բոլոր ա թեգերը կլինեն կարմիր գույնի:

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

Իսկ 2-րդ տարբերակում 1200-ից մեծ էկրանների վրա մեր ա թեգերը կլինեն կարմիր գույնի: Գոյություն ունեն հիմնական չափեր, որոնց համար գրվում են media query-ները, բայց դա չի նշանակում, որ մենք չենք կարող մեր ուզած չափի համար գրել media query:

Media query-ի մարմնում գրում ենք այն ամենը ինչը պետք է փոխվի մինչև այդ չափի էկրանների համար, եթե նշված է max-width կամ նշում ենք այն ամենը, ինչը պետք է փոխվի նշված էկրանից մեծ լինելու դեպքում (min-width):

Media Query-ները գրվում են հիմնականում՝

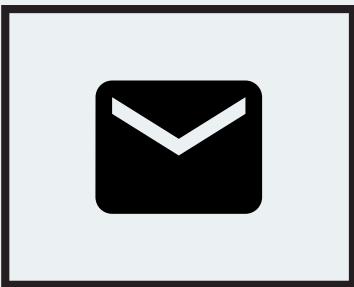
- **min-width:1200px** - մոնիթորների
- **max-width:1024px**- notebook-ների
- **max-width:768px** - պլանշետների
- **max-width:425px**- հեռախոսների
- **max-width:320px**-փոքր հեռախոսների համար:



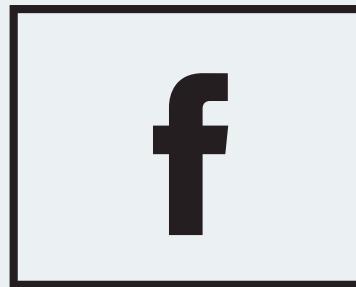
Նշենք media query-ի գոման
օրինակ.

```
@media only screen and (min-width: 1201px){  
    ul li{  
        width: 200px;  
    }  
}  
  
@media only screen and (max-width: 1200px){  
    ul li{  
        background-color: green!important;  
    }  
}  
  
@media only screen and (max-width: 991px){  
    ul li{  
        display: block!important;  
        width: 200px;  
    }  
}  
  
@media only screen and (max-width: 768px){  
    ul li{  
        background-color: red!important;  
    }  
}  
  
@media only screen and (max-width: 560px){  
    ul li{  
        display: block!important;  
        background-color: #0B1B48!important;  
        width: 120px;  
    }  
}  
  
@media only screen and (max-width: 480px){  
    p{  
        display: block!important;  
    }  
}
```

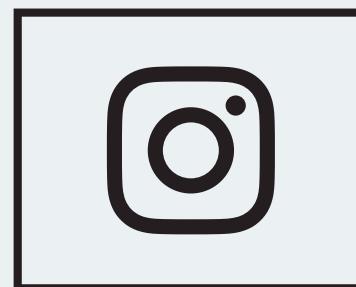
Follow Us



EMAIL



FACEBOOK



INSTAGRAM