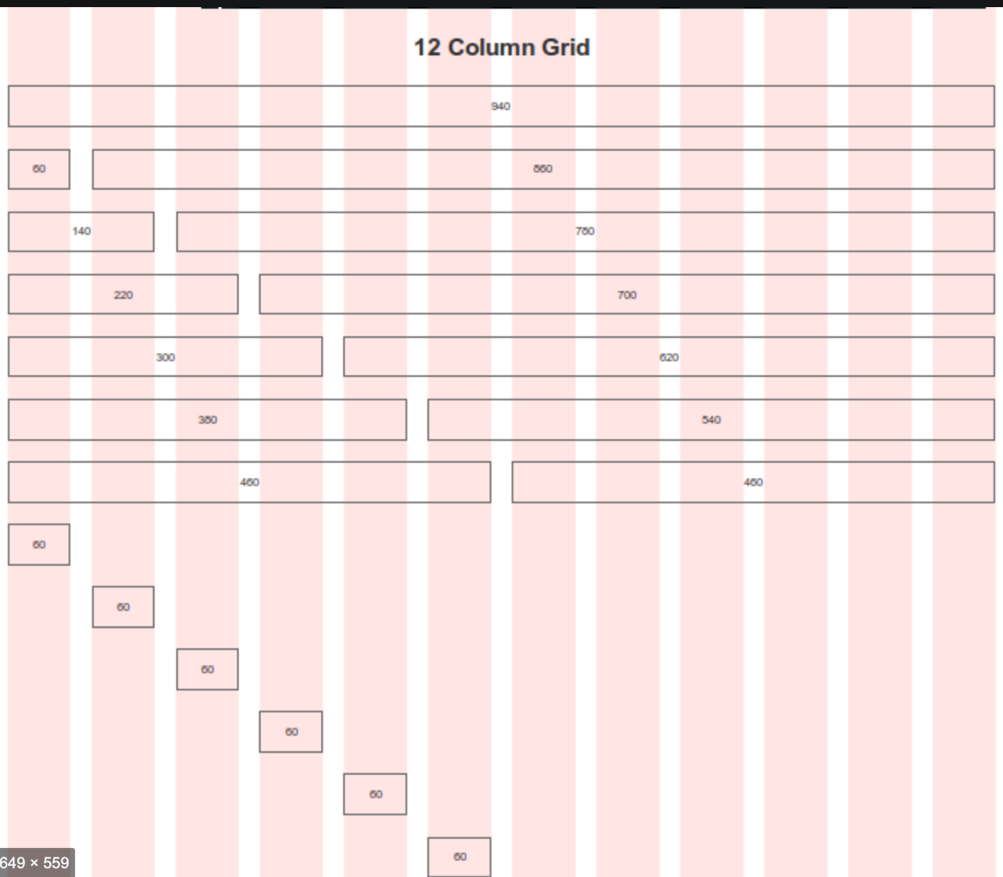
Css grid ray out

오늘은 그리드 레이아웃, 플렉스박스 레이아웃에 대하여 알아보겠습니다.

Grid란 어떤 공간에 임의의 수의 가상의 선을 그어 선으로 나눠진 영역들을 기준으로 아이템들을 배치시키는 레이아웃 기술을 말합니다. 그리드 레이아웃은 float레이아웃 보다는 쉬우며, 낮은 버전의 브라우저에서도 많이들 지원하기에 float레이아웃보다는 많이 사용됩니다.

다만 네이버 앱브라우저에서는 지원하지 않습니다.



보통 그리드 레이아웃을 사용 할 때 가상의 임의의 선의 개수를 12개로 선정합니다 . 선이 많으면 많을 수록 아이템을 배치할 때 세심하게 조정 가능 하지만, 선이 많아지면 낮은 해상도 모니터에서는 선과 선 사이의 픽셀이 고 해상도 모니터 보다 훨씬 작아져 세심하게 하려는 의도가 사라지게 됩니다. 그렇기 때문에 너무 작게 쪼개는 것 보다 적당한 수의 그리드를 사용 하는 것이 좋습니다.

그리드 레이아웃은 선과 선 사이의 공간을 이용하기 때문에 아이템의 배치가 쉬운 것이 장점입니다. 사진의 그리드 속 아이템들을 보면 다양한 크기로 아이템들의 크기가 정해지고 아이템들의 위치를 정할 수 있습니다.

천천히 예제를 통하여 grid레이아웃을 사용할 때 어떤 프로퍼티가 있는지, 어떻게 사용하는지 한번 알아봅시다. Html을 만들고 body태그안에 다음과 같은 형태의 div를 만들어 주세요 .

<div class="wrapper">

    <div>1</div>

    <div>2</div>

    <div>3</div>

    <div>4</div>

    <div>5</div>

    <div>6</div>

  </div>

우리는 저 wrapper라는 div속에 가상의 선을 그어보도록 하겠습니다 .

.wrapper{

      display: grid;

    }

Display : grid 속성을 부여함으로써 이제 부터 wrapper는 그리드를 담고 있는 그리드 컨테이너가 됩니다.

이렇게 하고 브라우저에서 확인하니 별 차이가 없습니다.

조금 보기 좋게 하기 위해 inline style로 div마다 색상과 폰트 컬러를 지정 해주도록 하겠습니다.

 <div style="background-color: aqua; color : white">1</div>

    <div style="background-color: blue; color : white">2</div>

    <div style="background-color: black; color : white">3</div>

    <div style="background-color: red; color : white">4</div>

    <div style="background-color: green; color : white">5</div>

    <div style="background-color: turquoise; color : white">6</div>

그리고는 이 div를 3 \* 2 의 테이블 형태로 만들기 위해서 wrapper에 추가합니다.

  .wrapper{

      display: grid;

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

      grid-template-rows: 50px 50px;

    }

이렇게 하니 깔끔하게 정리된 모습의 div가 나타났습니다.

Grid-template-columns 속성과 grid-template-rows라는 속성들이 추가가 되었는데, 먼저 grid-template-columns는 명시적인 행의 컬럼의 넓이를 지정해줍니다. 띄어쓰기로 배치되어있는 크기들은 첫번째 컬럼부터 마지막 컬럼까지 매핑되며, 만약 100px 100px이라고 적어준다면 한 row에는 2개의 컬럼밖에 들어가지 않습니다. 만약 다음과 같이 작성한다면

.wrapper{

      display: grid;

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

      grid-template-rows: 50px ;

    }

Div가 나란히 6개가 100px씩 영역을 차지 하는 모습을 볼 수 있습니다.

Grid-template-rows또한 몇 번 사용하면서 감이 오셨을 것 같은데 rows는 명시적인 로우의 높이입니다. 그래서 저곳에 2번 크기를 지정 해준다면 2개의 row가 두개가 되서 각 row는 명시되어 있는 크기 마다 높이가 결정되게 됩니다.

굳이 픽셀 단위 말고 %단위도 사용가능 하니 다들 써보세요.

이번에는 3 \*3 테이블을 만들어주세요 .

그리고는 첫 번째 div에 class = “item1”을 추가해주시고 다음과 같이 작성해주세요

.item1 {

    grid-column-start: 1;

    grid-column-end: 4;

   }

작성하고 확인해보니 첫번째 div가 3번째 컬럼의 끝까지 커진 모습을 볼 수 있습니다.

왜 column : 3이 아니라 4일까요 ? 그 이유는 다음과 같습니다 컬럼이 3개로 나뉘어 졌을 때 눈에 보이지 않는 그리드는 사실 3개가 아니라 4개입니다.



그리드가 이렇게 그려졌기 때문에 마지막 그리드는 4가 되게 됩니다.

Column도 있었으니 Grid-rows도 있겠죠 ? 그리드 rows와 column을 이용해서 다음과 같이 적으면

.item1 {

      grid-column-start: 1;

      grid-column-end: 3;

    }

    .item3 {

      grid-row-start: 2;

      grid-row-end: 4;

    }

   .item4 {

      grid-column-start: 2;

      grid-column-end: 4;

    }

이렇게 만들 수도 있습니다 .

그리고 명시적으로 그리드 별 영역을 지정해준다음 영역에 들어갈 요소들을 지정해주면 해당 요소가 알아서 그리드에 맞게 배치 시킬 수 있습니다.

그리드를 3\*4 로 만들어보도록 하겠습니다.

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

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

그리고 grid-template-areas : 라는 속성을 이용할 건데 이 속성은 큰 따옴표로 감싸진 스트링 구조 안에 컬럼과 row에 맞게 영역을 지정해주면 나중에 태그들을 영역별로 지정할 때 알아서 배치되고 크기 또한 컬럼에 맞게 조절 됩니다.

.wrapper{

      display: grid;

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

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

      grid-template-areas:

      "header header header"

      "navi main main"

      "navi main main"

      "footer footer footer";

    }

다음으로 div 들을 4개로 줄인 다음 class를 지정 해줍니다.

<div class ="item1 header" style="background-color: aqua; color : white">1</div>

    <div class ="navi" style="background-color: blue;color : white">2</div>

    <div  class ="main" style="background-color: black; color : white">3</div>

    <div class ="footer"style="background-color: red; color : white">4</div>

그리고 각 클래스별로 grid-area속성을 이용해서 영역별로 배치시켜 줍니다.

    .header{

      grid-area: header;

    }

    .navi{

      grid-area: navi;

    }

    .main{

      grid-area: main;

    }

    .footer{

      grid-area: footer;

    }

이렇게 작성 하면 기본 블로그 레이아웃이 완성 되게 됩니다.

마지막으로 column간 공백을 지정 할때는 그리드 컨테이너에서 column-gap : 크기 를 사용하고, 행간의 간격을 결정할 때는 row-gap을 사용합니다.