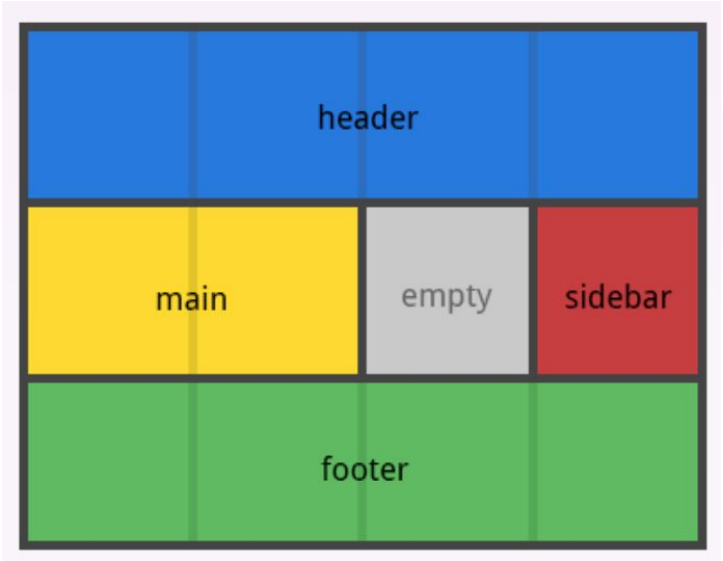
CSS Grid Layout

NORTH LINK

Vad är Grid?

CSS-verktyg för att bygga rutnätsbaserade layouter i två dimensioner.



NORTH LINK

Grid

- Grid kan användas istället för flexbox eller bootstrap för att bygga responsiva layouter.
- Grid är dock relativt nytt och har inte lika bra browserstöd som flexbox.
- Grid kan hjälpa oss att skapa komplexa layouter, som ibland kan vara svårare att uppnå med flexbox.

Vår första Grid-layout

Vi ska nu skapa vår första Grid-layout. Till att börja med skapar vi några rutor:

```html
<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 class="item"></div>
</div>
```

```css
.item {
  background: #ccc;
  min-height: 20px;
  margin: 10px;
}
```

Exempel

Display: Grid

För att göra vår container till en `grid` använder vi `display: grid.`

```
.container {
  display: grid;
}
```

Vi kan dock inte se någon större skillnad ännu. Rutorna ligger kvar på samma sätt. För att skapa oss ett rutnät behöver vi definera hur många kolumner vi vill ha. Detta gör vi med hjälp av `grid-template-columns`

grid-template-columns

Vi kan använda en ny enhet som grid introducerar som heter `fr` (fraction). Med den här enheten kan vi specificera hur många kolumner vi vill ha och hur stora de ska vara.

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Exempel

NORTH LINK

grid-template-columns

Vad händer om vi ändrar värdet på 1fr till 2fr på den första kolumnen? Då kommer den kolumnen att bli dubbelt så stor som de andra.

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```

Exempel

NORTH LINK

grid-template-columns

På det här sättet kan vi ganska enkelt skapa en struktur av kolumner:

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 4fr 1fr 2fr;
}
```

Exempel

grid-template-rows

På samma sätt som vi kan styra hur raderna ska bete sig med hjälp av `grid-template-rows`

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 4fr 1fr 2fr;
}
```

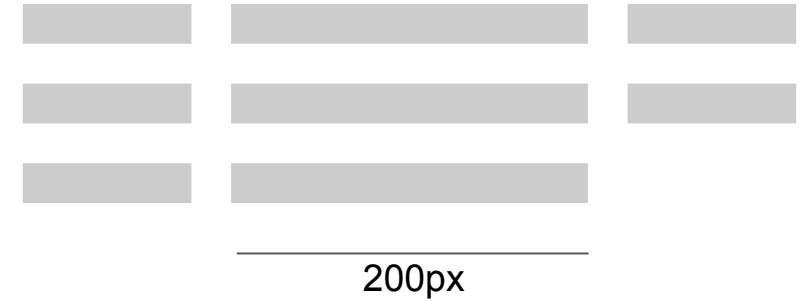

Exempel

NORTH LINK

Absoluta värden

Vi kan också använda oss av absoluta värden istället för fractions.

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



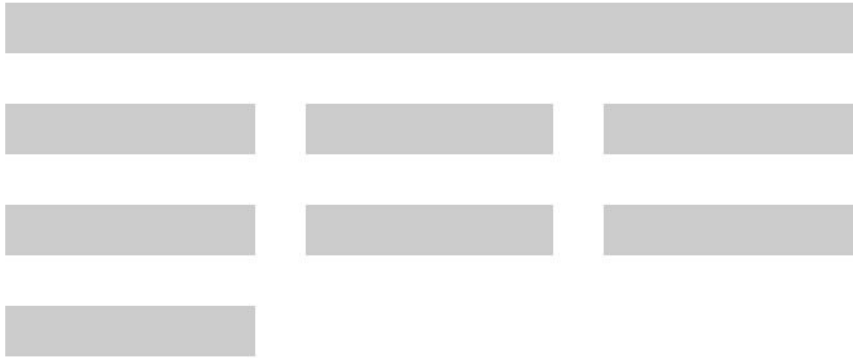
Exempel

Manipulera enskilda grid-rutor

Vi kan använda oss av `grid-column-start` och `grid-column-end` för att ändra storleken på en ruta:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}

.item:nth-child(1) {
    grid-column-start: 1;
    grid-column-end: 4;
}
```

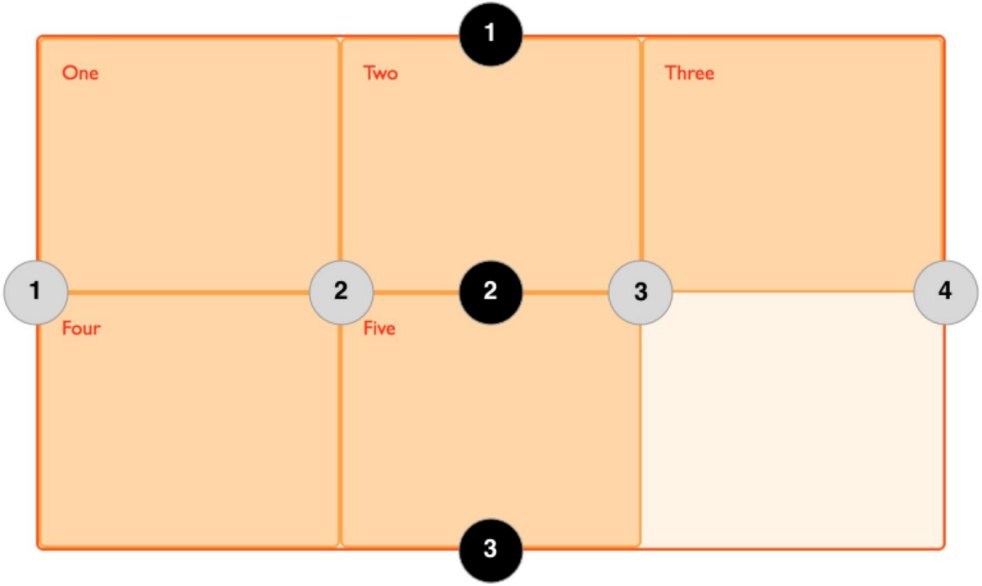


Exempel

NORTH LINK

Grid Lines

För att det ska bli rätt ska `grid-column-start` och `grid-column-end` ska motsvara imaginära linjer i rutnätet.




```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}

.item:nth-child(1) {
    grid-column-start: 1;
    grid-column-end: 4;
}
```

NORTH LINK

Grid-column

Vi kan också skriva så här istället:

```
.item:nth-child(1) {
    grid-column: 1 / 4;
}
```

Exempel

NORTH LINK

Grid-row

På samma sätt kan vi ändra på radernas storlek:

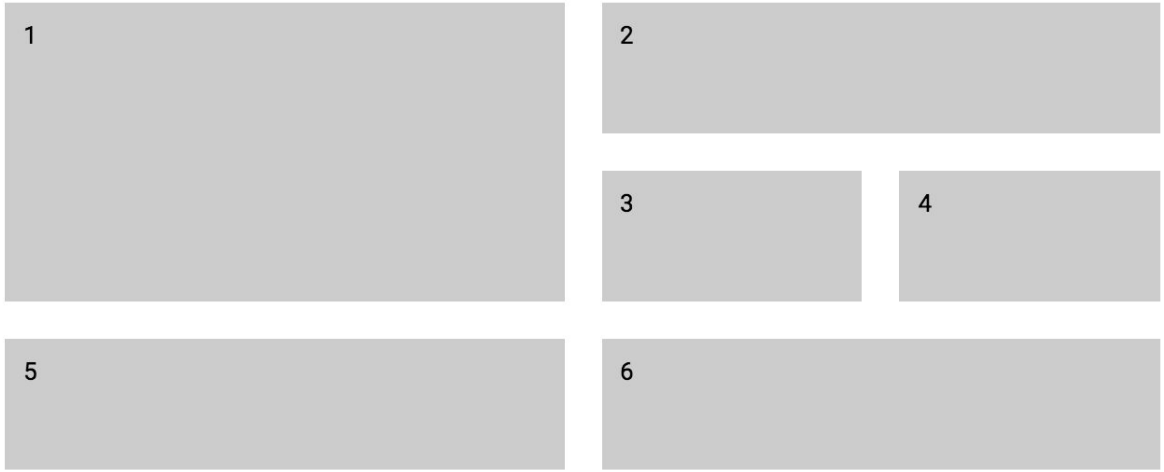
```
.item:nth-child(1) {
    grid-row: 1 / 3;
}
```

Exempel

NORTH LINK

Grid Layout - exempel

Med hjälp av de här verktygen kan vi redan bygga ganska komplexa strukturer:



Exempel

NORTH LINK

Grid column gap

Vi kan även specificera marginalerna mellan rutorna med hjälp av grid-column-gap och grid-row-gap:

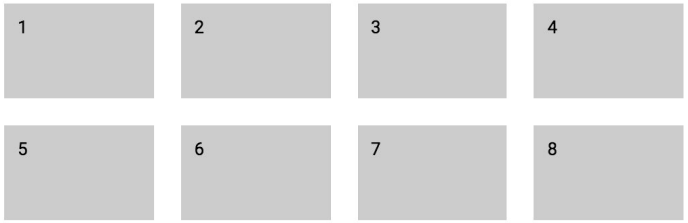
```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
  grid-column-gap: 10px;
  grid-row-gap: 10px;
}
```

Exempel

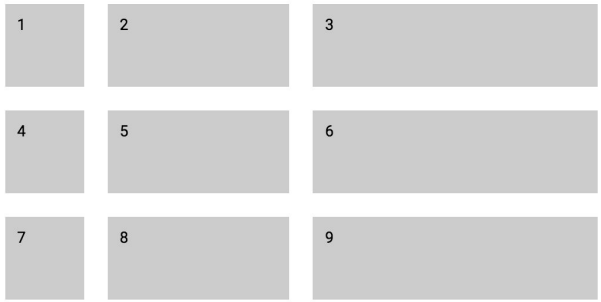
Övning

Prova att skapa följande layouter med grid:

a)



b)



c)

Object-fit: cover

I några av våra exempel kommer vi använda oss av en relativt ny CSS property som heter `object-fit`.

Genom att sätta `object-fit: cover` på en bild kan vi tvinga propertionerna på en bild att bli rätt i utbyte mot att den beskärs.

Dock stöds det inte av Internet Explorer

Exempel - Object-fit: cover

Exempel

Object-fit + CSS Grid

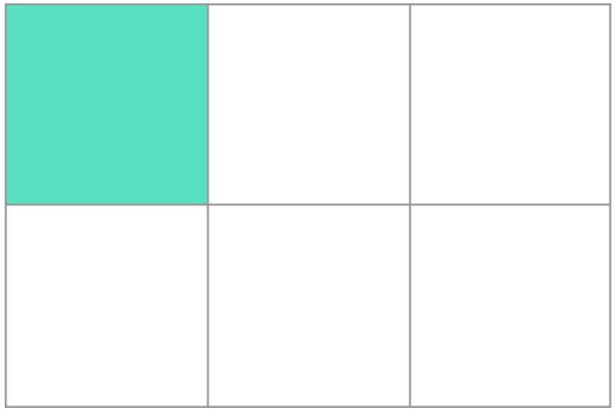
Grid och object-fit fungerar väldigt bra ihop om man vill göra t.ex. ett galleri i olika storlekar.

Då kan vi specificera upp vårt rutmönster och lägga bilder inuti våra grid-celler, som vi sedan kan visa delar av med object-fit.

NORTH LINK

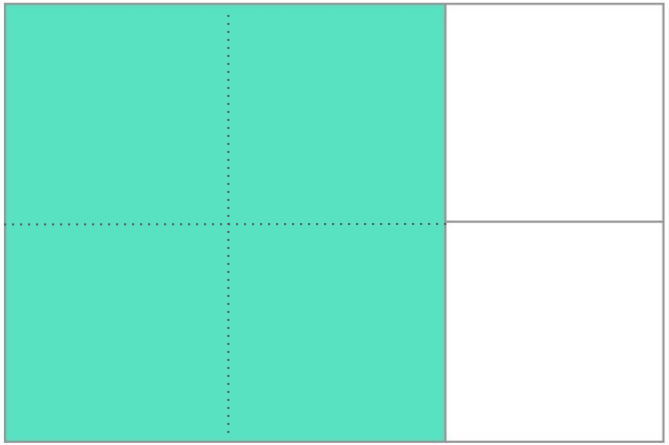
Grid Cell

En ruta i gridnätet



Grid Area

I grid kallar vi en Rektangulär yta med en eller flera rader/kolumner för en **Grid Area.**



NORTH LINK

grid-template-areas

`grid-template-areas` ger oss möjlighet att definera olika areor i vår layout. För detta behöver vi använda nya CSS-regler:

- grid-area: <string> - Används för att ge vårt element ett namn
- grid-template-areas - Används för att rita upp vår layout med hjälp av namnen.

```
grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
```

Template areas

Med hjälp av **`grid-template-areas`** kan vi definiera upp exakt hur vi vill att vårt grid-mönster ska bete sig:

```
.container {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-areas:
        "header header header header"
        "area1 area2 area2 area2"
        "footer footer footer footer";
}
```

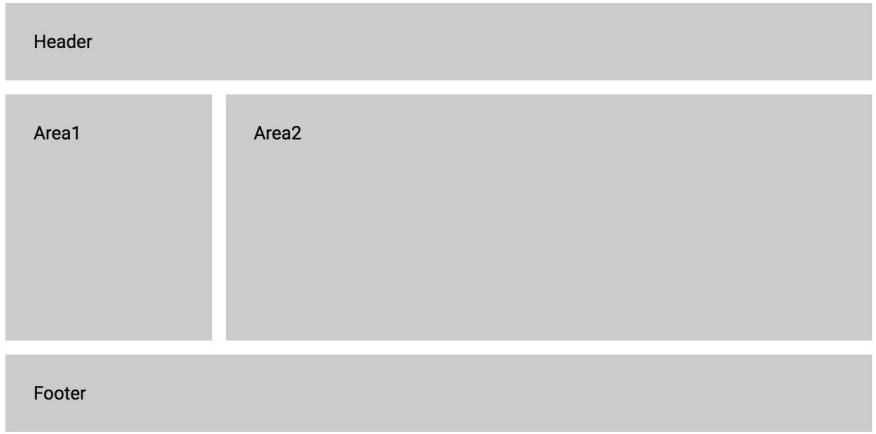
```
.header {
    grid-area: header;
}
.area1 {
    grid-area: area1;
}
.area2 {
    grid-area: area2;
}
.footer {
    grid-area: footer;
}
```

JSFiddle

Template areas

Vi kan även använda oss av rader när vi skriver våra template-areas:

```
.container {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-rows: 1fr 2fr 1fr 1fr;
    grid-template-areas:
        "header header header header"
        "area1 area2 area2 area2"
        "area1 area2 area2 area2"
        "footer footer footer footer";
}
```



[JSFiddle]

NORTH LINK

Exempel

Vi ska nu prova att implementera ett galleri med hjälp av CSS Grid.

Vårt mål är att få till samma struktur som på bilden till höger.

NORTH LINK

Hur börjar vi?

Vi definierar upp hur många bilder vi vill ha och lägger dem i en container.

```
<div class="container">
  <div class="a"></div>
  <div class="b"></div>
  <div class="c"></div>
  <div class="d"></div>
  <div class="e"></div>
</div>
```

NORTH LINK

Antalet kolumner

Först definerar vi hur många kolumner vi behöver.

På den översta raden ser vi att bilden till höger verkar vara 3 gånger så stor som bilden till vänster. Därför kommer vi att behöva 4 rader.

```
display: grid;
grid-gap: 10px;
grid-template-columns: 1fr 1fr 1fr 1fr;
```

JSFiddle

grid-area

Eftersom vi vill lösa uppgiften med en grid-template-area så måste vi ge våra grid-items namn som vi kan använda i våra areor.

```
.a {
  grid-area: a;
}

.b {
  grid-area: b;
}

.c {
  grid-area: c;
}

.d {
  grid-area: d;
}

.e {
  grid-area: e;
}
```

Vår grid-template-area

Nu vill vi definera våran area. Om vi utgår från bilden så kommer vi fram till följande area:

```
grid-template-areas:
  "a b b b"
  "c c d d"
  "e e e e";
```

JSFiddle

Bilder

Nu vill vi lägga in våra bilder och se hur de passar i vår layout.

```
<div class="container">
  <div class="a">
    <img src="https://i.imgur.com/xbyoAPv.jpg" />
  </div>
  <div class="b">
    <img src="https://i.imgur.com/OOBwAXCr.jpg" />
  </div>
  <div class="c">
    <img src="https://i.imgur.com/MNXC2pV.jpg" />
  </div>
  <div class="d">
    <img src="https://i.imgur.com/KKuQH.jpg" />
  </div>
  <div class="e">
   <img src="https://i.imgur.com/Afhv1so.jpg" />
  </div>
</div>
```

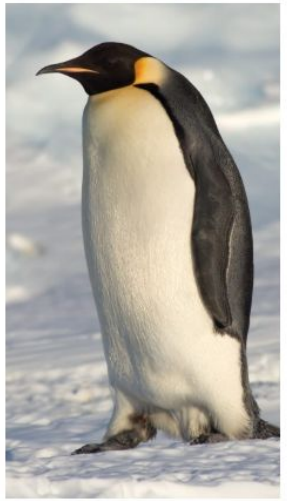
```
img {
  width: 100%;
}
```

JSFiddle

Bilder

Vi vill dock att bilderna ska fylla ut hela rutan, så vi lägger även till height: 100%, vilket gör att bilderna blir utdragna.

```
img {
  width: 100%;
  height: 100%;
}
```



JSFiddle

NORTH LINK

Object-fit: cover

För att få rätt proportioner på våra bilder använder vi nu object-fit: cover på bilden.

```
.container > div {
  background-color: #ccc;
  height: 100%;
}

img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}
```

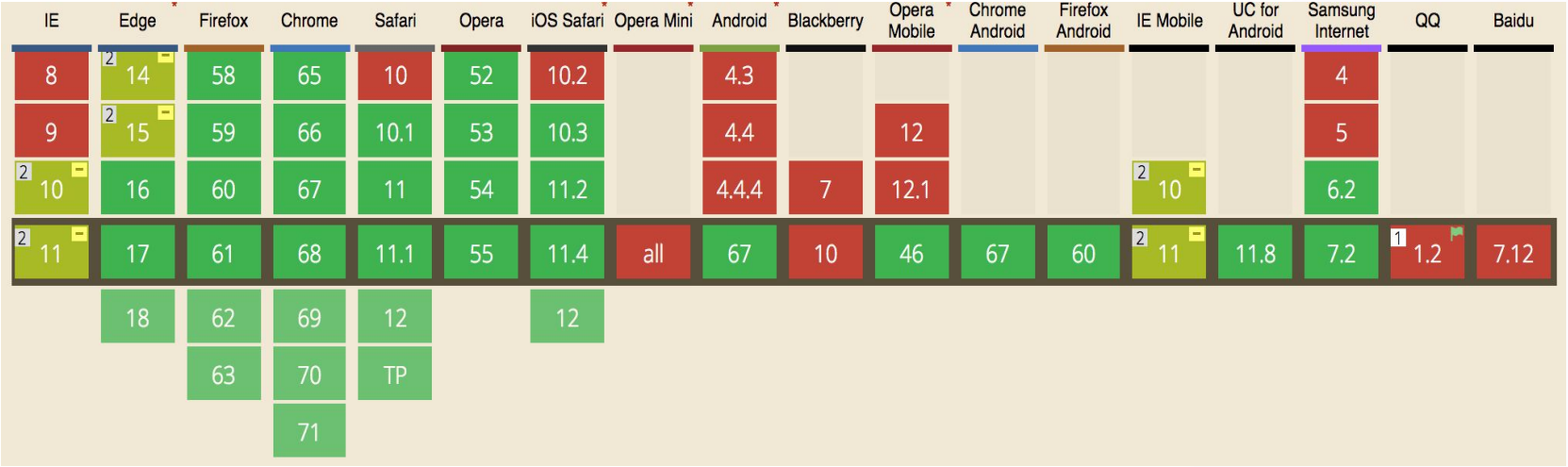
Vårt slutliga resultat

Nu är vårt rutmönster klart!

Resultat

NORTH LINK

Browser-stöd



IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android	Blackberry	Opera Mobile	Chrome Android	Firefox Android	IE Mobile	UC for Android	Samsung Internet	QQ	Baidu
8	2 14	58	65	10	52	10.2		4.3							4		
9	2 15	59	66	10.1	53	10.3		4.4		12					5		
2 10	16	60	67	11	54	11.2		4.4.4	7	12.1			2 10		6.2		
2 11	17	61	68	11.1	55	11.4	all	67	10	46	67	60	2 11	11.8	7.2	1 1.2	7.12
	18	62	69	12		12											
		63	70	TP													
			71														

https://caniuse.com/#feat=css-grid

NORTH LINK

Slutligen

Nya koncept som introduceras i grid är bland annat:

- repeat
- minmax
- auto-fit

Repeat

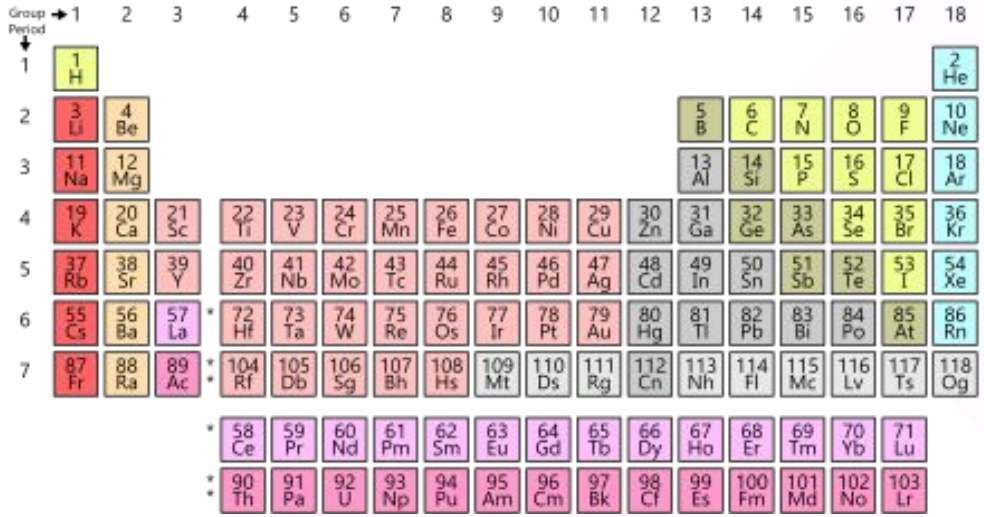
Repeat kan användas för att slippa skriva samma sak flera gånger.

repeat(4, 1fr) = 1fr 1fr 1fr 1fr.

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

Repeat

repeat (18, 1fr)....



NORTH LINK

minmax

minmax() är en funktion som gör att vi kan definiera ett intervall. T.ex. kan vi säga att en kolumn ska vara mellan 100px och 200px med minmax(100px, 200px)

```
grid-template-columns: 1fr minmax(100px, 200px) 20%;
```

auto-fit

Auto-fit kan användas tillsammans med repeat för att få automatisk radbrytning.

T.ex.

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

I det här fallet kommer alla element vi lägger i vår grid att få 150px i bredd och när de inte får plats på en rad kommer de visas på nästa.

JSFiddle

auto-fit

Ett vanligt sätt att använda använda auto-fit på är att kombinera det med minmax.

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

JSFiddle

Övningar

https://github.com/LinkNorth/HTMLCSS/tree/master/exercises/week6/grid

http://cssgridgarden.com/

Laboration 2

Glöm inte att lämna in Laboration 2 senast Fredag kl **08.59!**

NORTH LINK