Repetition

Planering

Vecka 42 - **Repetition**, Animationer, Media, Nyheter i CSS, ev. mer repetition.

Vecka 43 - Repetition + Tentamen

Idag

- Grid exempel
- Flexbox exempel
- CSS Selectors (nth-child och not)
- Transform
- Transitions

Imorgon

- Animationer

Grid

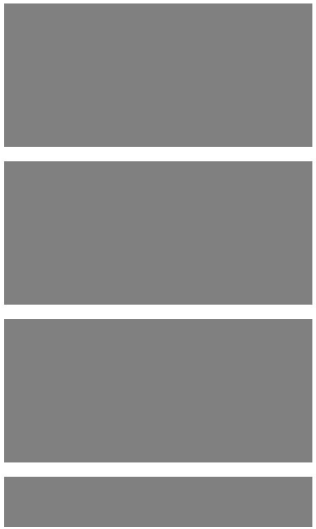
Vi ska nu lösa en av förra veckans grid-övningar:

NORTH LINK

Grid

Vi börjar med att skapa våra element och ge dem en min-height:

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

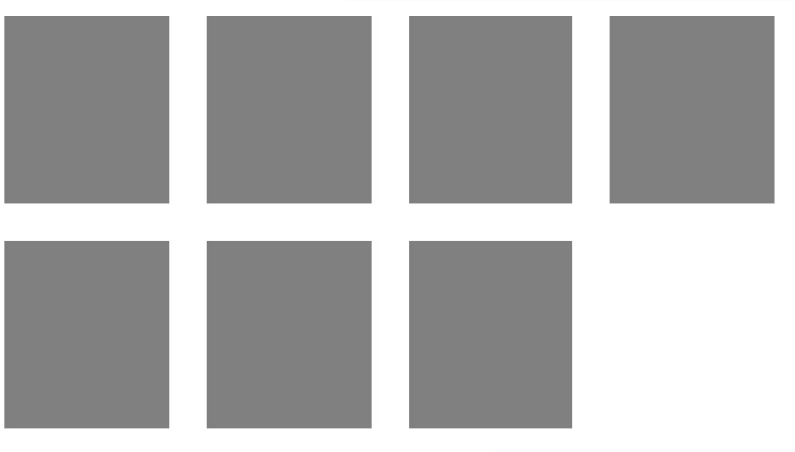


NORTH LINK

Grid

Sedan räknar vi hur många kolumner vi behöver och specificerar det med grid:

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



Resultat

NORTH LINK

Grid

Slutligen säger vi att den första rutan ska sträcka sig över två rader.

```
div > div:nth-child(1) {
  grid-row: 1 / 3;
}
```

Grid

Vi är klara!

Resultat

NORTH LINK

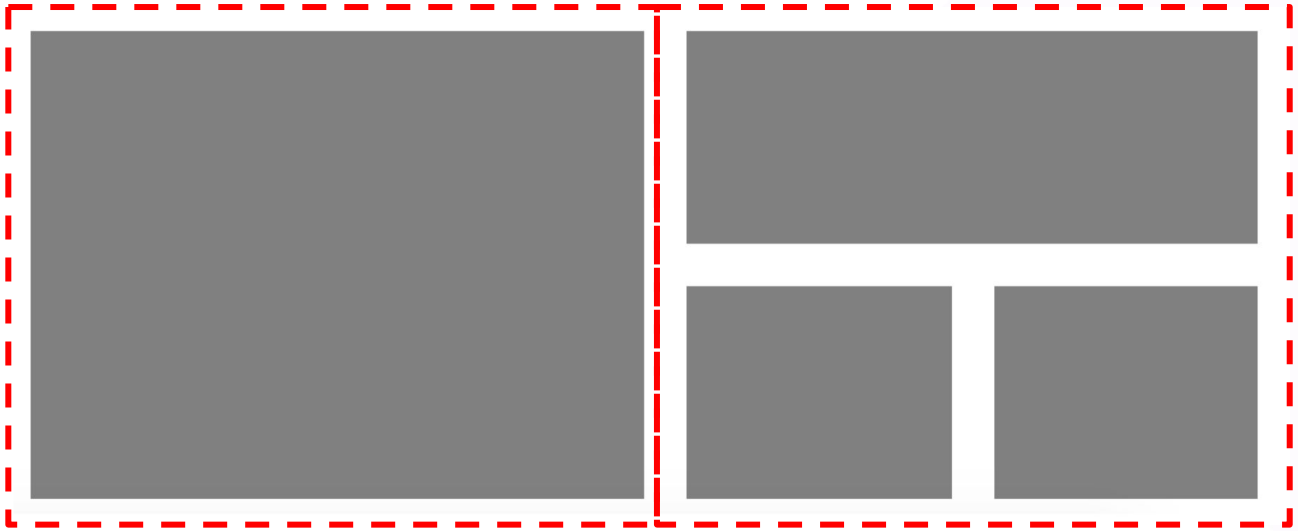
Flexbox

En av förra veckans övningar var att skapa följande struktur mha flexbox. Hur ska vi tänka här?

NORTH LINK

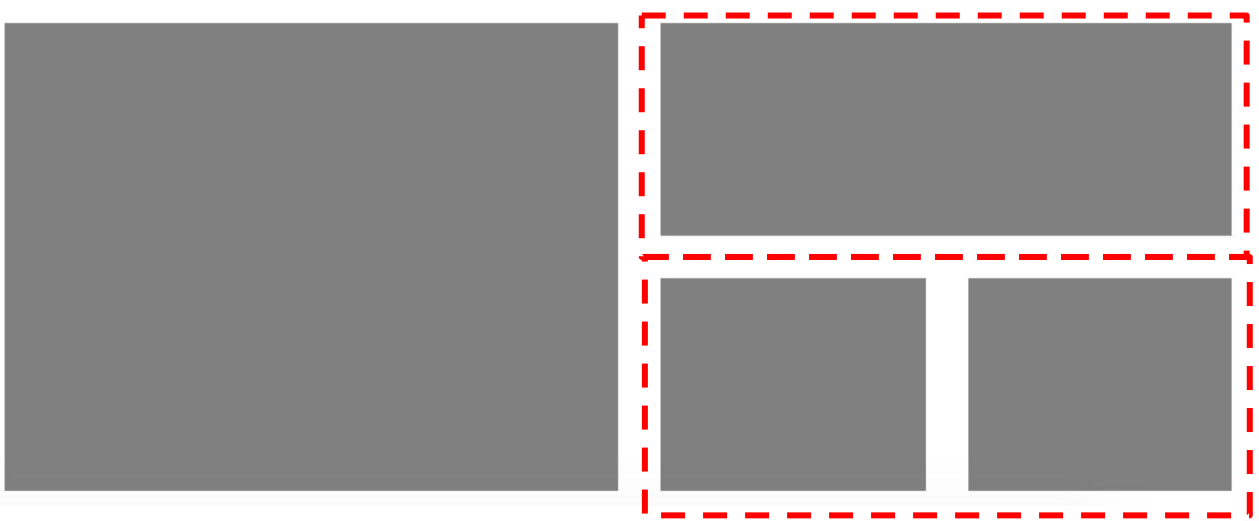
Flexbox

Vi försöker dela in våra block i potentiella flex-items. Vårt första segment består av två block som ska ligga bredvid varandra.



Flexbox

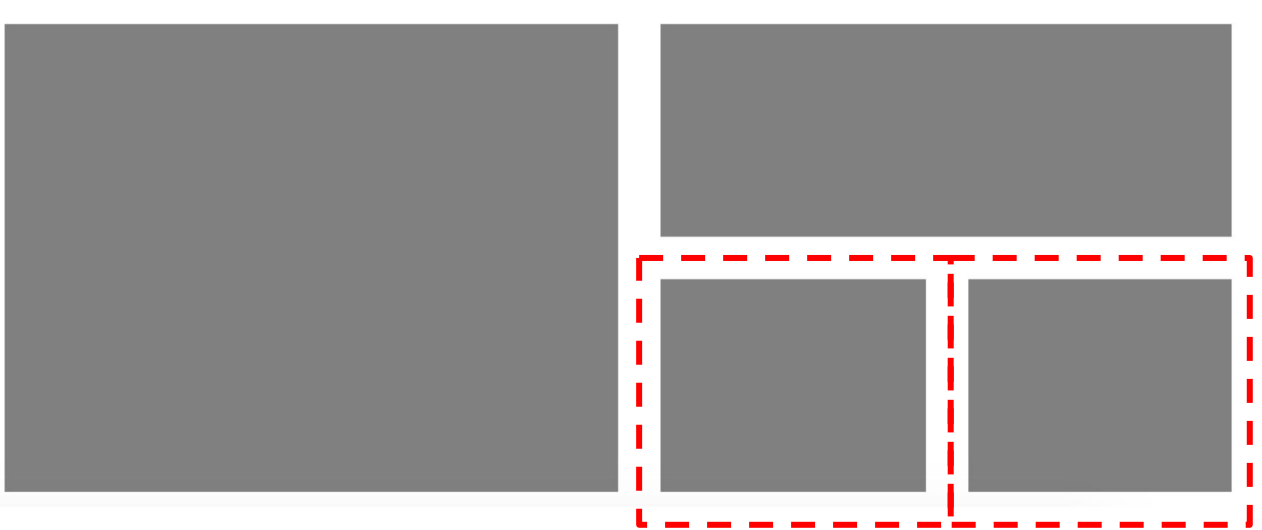
I vårt högra block hittar vi sedan ytterligare två block:



NORTH LINK

Flexbox

Och i det understa blocket hittar vi två potentiella flex-items till:



Flexbox

Vi behöver därför skapa en nästlad struktur. Vi börjar med att skapa två lika stora block:

```css
.item {
  min-height: 100px;
  margin: 10px;
  background-color: grey;
  flex: 1;
}

.sec1 {
  display: flex;
}
```

```html
<div class="sec1">
  <div class="item"></div>
  <div class="item"></div>
</div>
```

[Resultat]

Flexbox

Vi delar sedan upp den högra blocket i två lika stora block och lägger dem på rad:

```
<div class="sec1">
  <div class="item"></div>
  <div class="sec2">
    <div class="item"></div>
    <div class="item"></div>
  </div>
</div>
```

```
.sec1 {
  display: flex;
}

.sec1 > .item,
.sec2 {
  flex: 1;
}

.sec2 {
  display: flex;
  flex-direction: column;
}
```

Flexbox

Då får vi följande resultat:

[Resultat]

Flexbox

Slutligen vill vi dela upp det sista blocket i två delar. Vi gör på samma sätt:

```
<div class="sec1">
  <div class="item"></div>
  <div class="sec2">
    <div class="item"></div>
    <div class="sec3">
      <div class="item"></div>
      <div class="item"></div>
    </div>
  </div>
</div>
```

```
.sec3 {
  display: flex;
}

.sec3 > .item {
  flex: 1;
}
```

NORTH LINK

Flexbox

Slutligt resultat

Resultat

NORTH LINK

nth-child

- CSS Selector för att välja ett element beroende på dess position.
- Används på “syskon”-element.

Exempel - nth-child

Vi kan t.ex. skriva :nth-child(4) om vi vill välja det fjärde elementet i en lista.

```html
<ul>
  <li>Element 1</li>
  <li>Element 2</li>
  <li>Element 3</li>
  <li>Element 4</li>
  <li>Element 5</li>
</ul>
```

```css
ul li:nth-child(4) {
  font-size: 20px;
}
```

Resultat

NORTH LINK

Exempel - nth-child

Notera att det endast funkar på element som ligger bredvid varandra. I exemplet nedan kommer inte **C** att få en blå färg eftersom det inte är syskon till A och B.

```html
<div>
  <h1>A</h1>
  <h1>B</h1>
  <div>
    <h1>C</h1>
  </div>
</div>
```

```css
h1:nth-child(3) {
  color: blue;
}
```

Resultat

Exempel - nth-child

Nth-child bryr sig inte om vilken typ elementen har när den beräknar position. Vill vi t.ex. välja ut div-taggen med nth-child. Detta kommer inte fungera t.ex.

```
<div class="container">
  <h1>A</h1>
  <h1>B</h1>
  <div>
    <h1>C</h1>
  </div>
</div>
```

```
.container > div:nth-child(1) {
  color: blue;
}
```

NORTH LINK

Exempel - nth-child

Däremot kan vi använda position 3.

```html
<div class="container">
  <h1>A</h1>
  <h1>B</h1>
  <div>
    <h1>C</h1>
  </div>
</div>
```

```css
.container > div:nth-child(3) {
  color: blue;
}
```

Resultat

NORTH LINK

Exempel - nth-child

Om vi vill välja ut ett element oberoende av typ kan vi använda *

```
div > *:nth-child(3) {
  color: blue;
}
```

Resultat

nth-child - Even & Odd

När vi behöver välja vart annat element kan vi använda, **even** och **odd.**

- even - väljer alla element som kommer på jämna positioner (2, 4, 6 etc)
- odd - väljer alla element som kommer på udda positioner (1, 3, 5 etc)

```
#list1 > div:nth-child(even) {
  color: blue;
}
```

Resultat

nth-child - Anpassade funktioner

Vi kan även skriva våra egna definitioner för att välja ut element på formen **An+B**.
Följande selector väljer ut vart tredje element:

```
#list1 > div:nth-child(3n + 1) {
  color: blue;
}
```

Resultat

NORTH LINK

:not()

Vi kan använda **:not** för att exkludera element från våra selectors. Inuti :not()-funktionen kan vi skriva selectors som vi inte vill ska omfattas av vår CSS-kod.

:not - Exempel

Här vill applicera styling på alla .item-element **utom** den som har klassen **.active.**

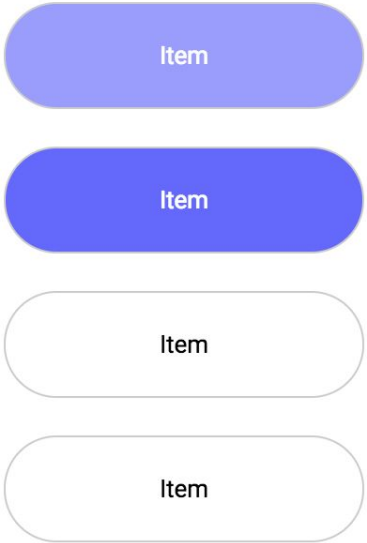
```
<div class="container">
  <div class="item">Item</div>
  <div class="item active">Item</div>
  <div class="item">Item</div>
  <div class="item">Item</div>
  <div class="item">Item</div>
</div>
```

```
.item:not(.active):hover {
  color: white;
  background: #9a9aff;
}
```

NORTH LINK

:not - Exempel

När vi tittar på vårt exempel ser vi att .active-elementet inte får stylingen, vilket var vad ville uppnå.



Resultat

NORTH LINK

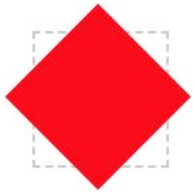
Transform

`transform: <property>`

Används för att manipulera ett objekts storlek, form, position i rummet etc.

Transform

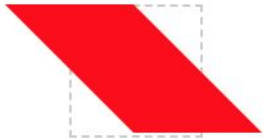
Rotate



Translate

Scale

Skew



Exempel: Transform

JSFiddle

Rotate

Vi kan rotera ett element i tre dimensioner. X, Y och Z-axeln.

- rotateX
- rotateY
- rotateZ

Exempel

Rotate - Exempel

Vänd pilen så att den pekar nedåt istället.

NORTH LINK

Rotate - Exempel

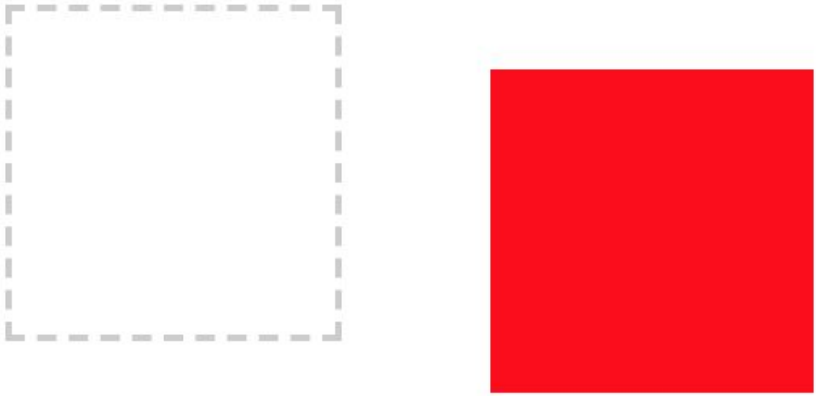
Vi vill att pilen ska vända sig 90 grader i z-planet, så vi använder rotateZ:

```
transform: rotateZ(90deg);
```

Exempel

Translate

Om vi vill flytta på ett element kan vi använda translate(X, Y)

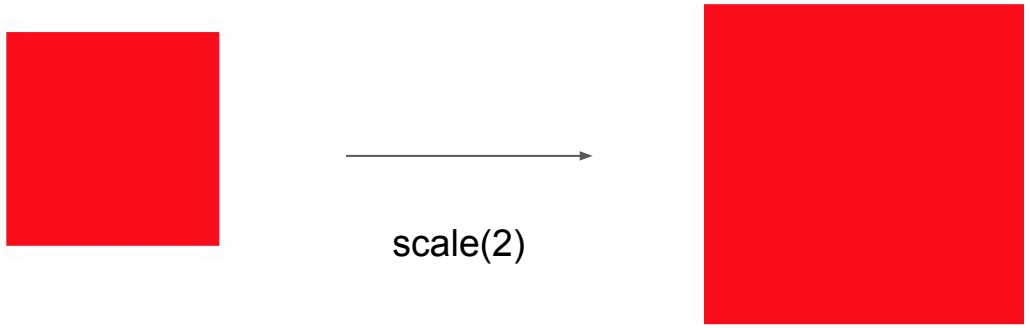


Exempel

NORTH LINK

Scale

Om vi vill ändra storlek på ett element kan vi skala det med **scale**



Exempel

Transitions

Vi kan använda transitions för att skapa animationer.

```
transition: <property> <duration> <timing-function>
```

- property - Vad vi vill animera
- duration - Hur lång animationen ska vara
- timing-function - Funktion som beskriver hastigheten över tid för animationen

Transitions

- När aktiveras transitions?
  - Byter mellan två tillstånd
  - T.ex. :hover eller med javascript
- Animatable CSS Properties
- `transition: all`
- Går även att använda:
  - transition-delay
  - transition-duration
  - transition-property
  - transition-timing-function

Exempel: Transition

```
transition: background-color 2s linear;
```

JSFiddle

NORTH LINK

Exempel: Transition

```
transition: all 0.5s linear;
```

JSFiddle

NORTH LINK

Transitions + Transform

JSFiddle

NORTH LINK

Exempel - Transition + Transform

Vi ska nu prova att skapa en transition där vi spegelvänder en bild och gör den lite mindre när vi för muspekaren över bilden.

NORTH LINK

Exempel - Transition + Transform

För att uppnå detta skriver vi följande transform på vår bild, där vi roterar längst Y-axeln 180 grader.

```
.img-container:hover {
  transform: rotateY(180deg) scale(0.9);
}
```

Och skriver vår transition:

```
transition: transform 0.5s ease 0.1s;
```

Exempel - Transition + Transform

Resultat

Övningar

https://github.com/LinkNorth/HTMLCSS/tree/master/exercises/week7/css