Les 7: Lay-out CSS Grid

HO GENT

Inhoud

- Basisconcepten – CSS Grid
- Inleidend voorbeeld CSS Grid
- CSS Grid
  1. de *grid container (display: grid)*
  2. definiëren van de *grid (grid-template-columns, …)*
  3. positioneren van *grid items* op de grid (*grid-row, grid-column, …)*

  Klik om tekst toe te voegen

  – *Grid-template-areas* property
  – Box alignment in CSS Grid Layout (*justify-self*, *align-self*, …)

Basisconcepten – CSS Grid

HO GENT

CSS Grid Layout

- CSS Grid Layout is een twee dimensionaal lay-out model voor CSS. Het heeft veel mogelijkheden voor de **positionering** van boxes en hun inhoud, alsook voor het controleren van de afmetingen (**sizing**) van de boxes.

HO GENT

Flexible Box Layout vs Grid Layout

- In tegenstelling tot ‘Flexible Box layout’ (kort Flexbox of flex), dat een eendimensioneel lay-outsysteem is, is ‘Grid Layout’ (kort Grid) een tweedimensioneel lay-outsysteem. Zie afbeelding volgende dia.
- Grid en flex kunnen samen gebruikt worden om complexe lay-outs te maken.
- Flexbox wordt besproken in de volgende les.

Flexible Box Layout vs Grid Layout

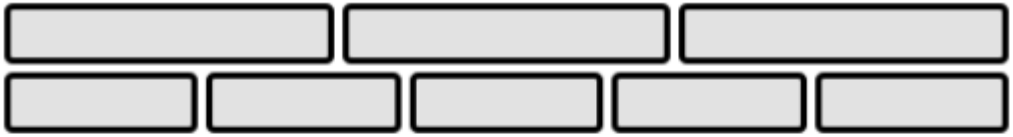


Figure 1 *Representative Flex Layout Example*

Figure 2 *Representative Grid Layout Example*

Bron: https://www.w3.org/TR/css-grid-1/#intro, laatst geraadpleegd op 2021-10-25

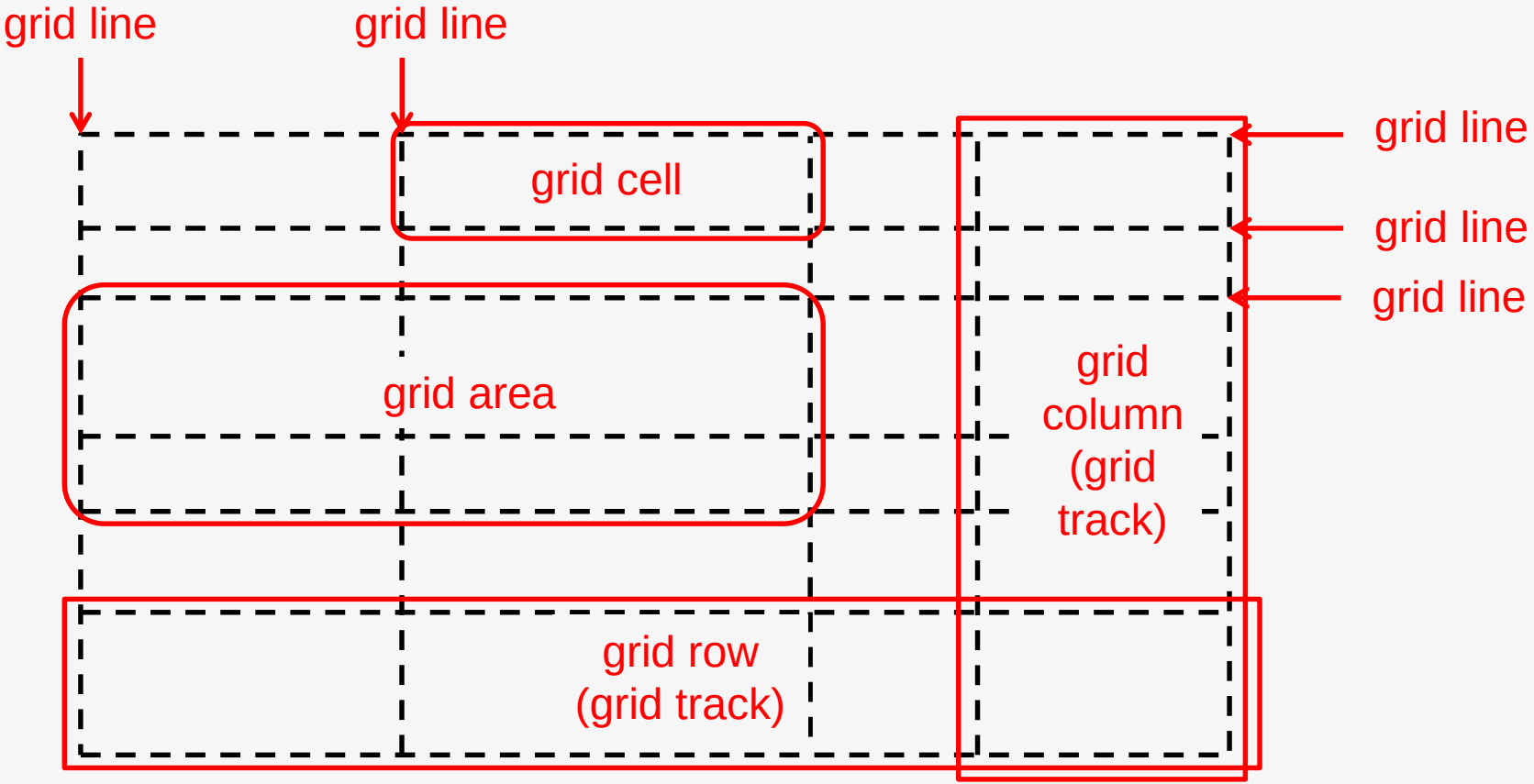
HO GENT

Wat is een grid?

- Een *grid* (of raster) bestaat uit (denkbeeldige) horizontale en verticale lijnen die een *grid container* opdelen in rijen, kolommen en cellen. Te vergelijken met een tabel.
- Op de volgende dia vind je een afbeelding van een grid waarop verschillende grid-termen zijn aangeduid. Meer info vind je op https://www.w3.org/TR/css-grid-1/#grid-concepts

HO GENT

Grid terminology



Inleidend voorbeeld CSS Grid

HO GENT

Theoriebestanden (voorbeelden)

- Open in Visual Studio Code de map **01-css-grid-inleidend-voorbeeld**

HO GENT

Inleidend voorbeeld CSS Grid

- Beschouw een element met daarin een aantal child elements.

Voorbeeld:

```
<div>
  <div>item 1</div>
  <div>item 2</div>
  <div>item 3</div>
  <div>item 4</div>
  <div>item 5</div>
</div>
```

(Voorbeeld)

```
<body>
  <header>Header</header>
  <nav>Navigation</nav>
  <main>Main area</main>
  <footer>Footer</footer>
</body>
```


HO GENT

Inleidend voorbeeld CSS Grid:

We voorzien eerst de ‘boxes’ van enige layout (zie vorige les)

HTML

```html
<div class="grid-container">
  <div>item 1</div>
  <div>item 2</div>
  <div>item 3</div>
  <div>item 4</div>
  <div>item 5</div>
</div>
```

CSS

```css
/* De boxes opmaken */
* {
  box-sizing: border-box;
}

.grid-container {
  background-color: cyan;
  border-radius: 5px;
  padding: 5px;
}

.grid-container div {
  border: 1px solid black;
  background-color: white;
  border-radius: 5px;
  padding: 1em;
}
```


HO GENT

Inleidend voorbeeld CSS Grid:

Hoe nu gebruikmaken van CSS Grid Layout?

1. Creëer een *grid container* met

   ```
   display: grid;
   ```

   Het creëren van een *grid container* heeft tot gevolg dat alle directe kind-elementen van het *grid container*-element, *grid items* worden.

   Met `display: grid` definieer je dus niet alleen de *grid container*, maar ook impliciet de *grid items*.

2. Definieer de *grid (*het aantal rijen en/of kolommen en hun hoogte/breedte), met

   ```
   grid-template-columns
   grid-template-rows
   ```

   (kan ook nog op andere manieren, zie later)

HO GENT

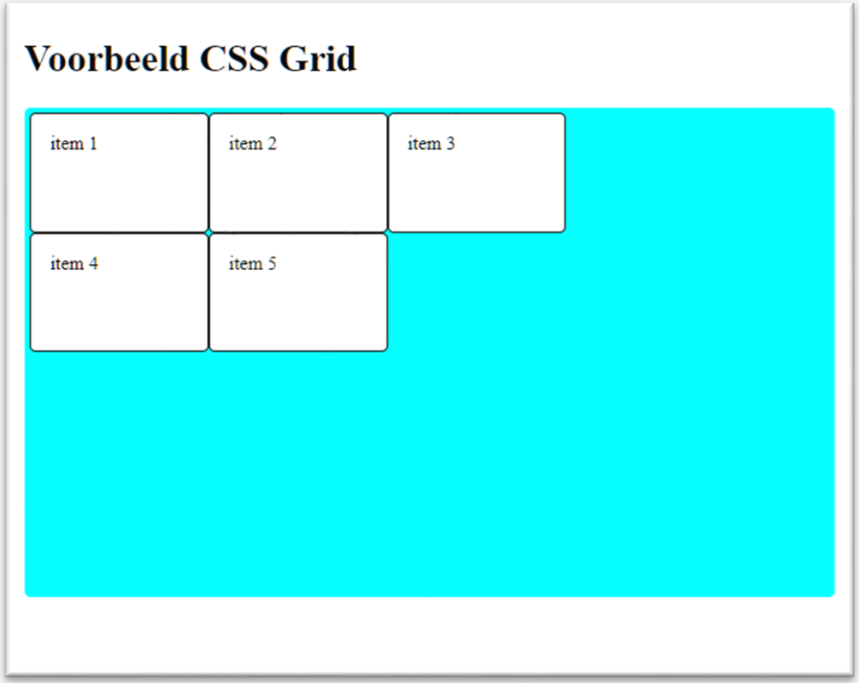
Inleidend voorbeeld CSS Grid:

Definiëren van de grid-container, de grid-items en de grid

CSS

```css
/* CSS Grid Layout */
.grid-container {
  display: grid;
  grid-template-columns:
    150px 150px 150px;
  grid-template-rows:
    100px 100px 100px 100px;
}
```

Merk op dat de grid items automatisch rij per rij in de grid geplaatst worden.



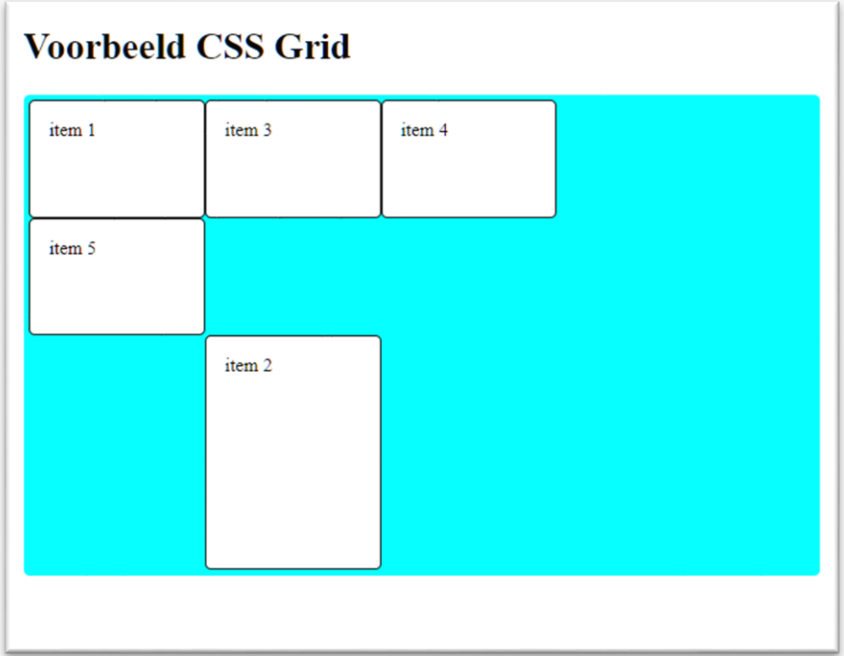
HO GENT

Inleidend voorbeeld CSS Grid:

Grid items positioneren op de grid met Line Numbers

We kunnen grid-items ook expliciet positioneren op de grid. In het voorbeeld wordt ‘item 2’ gepositioneerd op de grid.

```
/* CSS Grid Layout */
.grid-container {
  zie vorig dia
}
div:nth-of-type(2) {
  grid-row: 3/5;
  grid-column: 2;
}
```

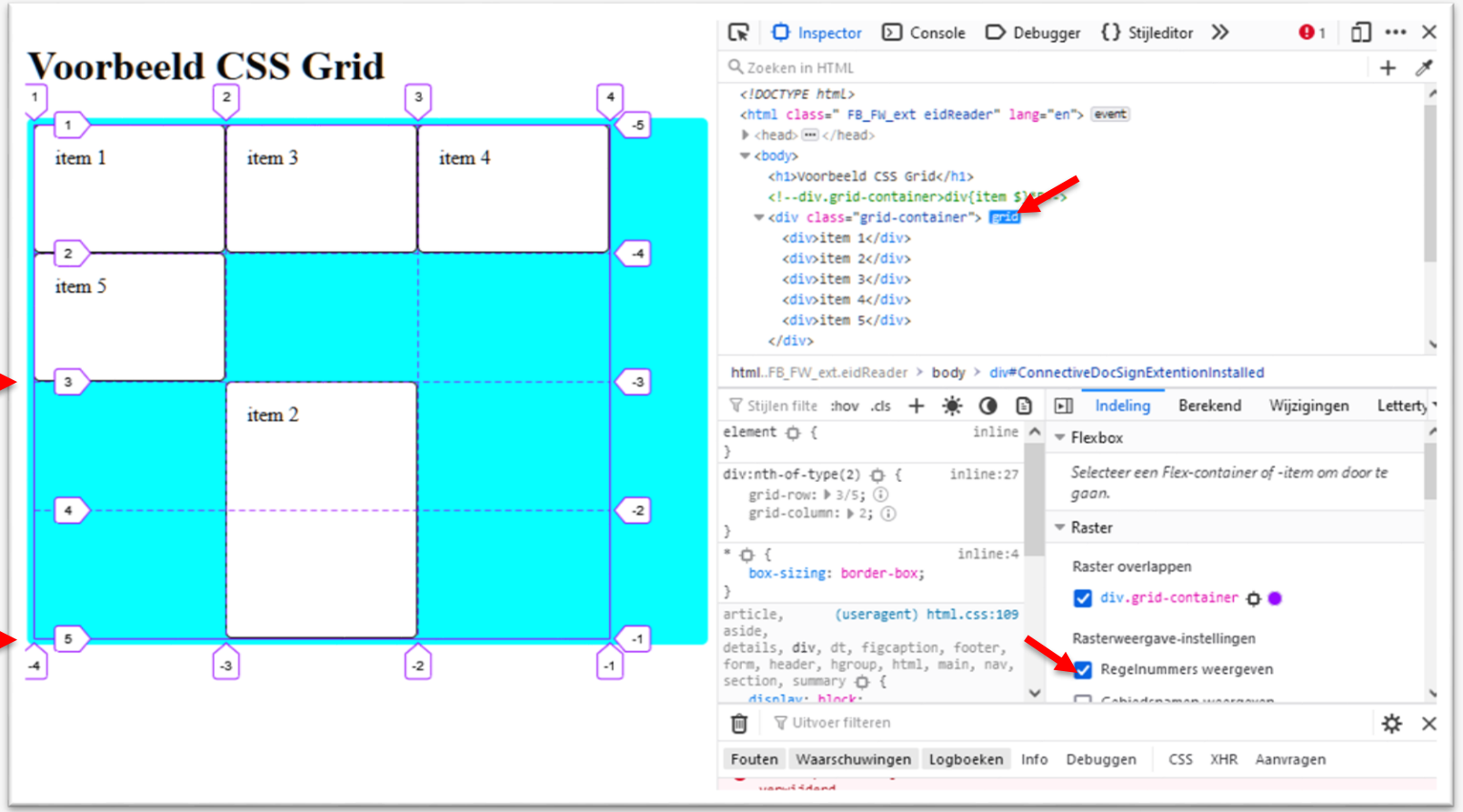


Merk op dat de getallen bij grid-row en grid-column lijnnummers zijn (zie volgende dia). 'grid-row: 3/5’ (betekent dus dat de div rij 3 en rij 4 overspant

HO GENT

Inleidend voorbeeld CSS Grid:

Afbeelding met Grid Lines en Line Numbers zichtbaar via de Firefox Grid Inspector



CSS Grid Layout

HO
GENT

Waarom CSS grid layout?

- Zorgt voor een goede scheiding tussen inhoud (html) en opmaak (css). We kunnen een html-element een andere positie op de grid geven zonder dat we daarvoor de markup (html) moeten wijzigen.

HO GENT

Algemene procedure gebruik Grid Layout

1. Creëer een *grid container*. Hiermee definieer je impliciet ook de *grid items*.
2. Definieer de *grid*
3. Plaats de *grid items* op de grid.

HO GENT

1. Creëer de *grid container*

Je creëert een *grid container*-element met `display: grid` of `display: inline-grid`.

- `display: grid` creëert een block-level-element
- `display: inline-grid` creëert een inline-level-element.

Het creëren van een *grid container* heeft tevens tot gevolg dat alle directe kind-elementen van het *grid container*-element, *grid items* worden.

HO GENT

2. Definieer de *grid*

De drie properties `grid-template-rows, grid-template-columns` en `grid-template-areas` (zie later) definiëren de ‘expliciet grid ‘ van een grid-container. De uiteindelijke grid kan echter groter zijn vanwege grid-items die buiten de ‘expliciet grid’ geplaatst worden. In dit geval zullen er impliciet tracks gecreëerd worden. De afmetingen van deze tracks worden bepaald door `grid-auto-rows` en `grid-auto-columns`.

HO GENT

Voorbeeld 01:

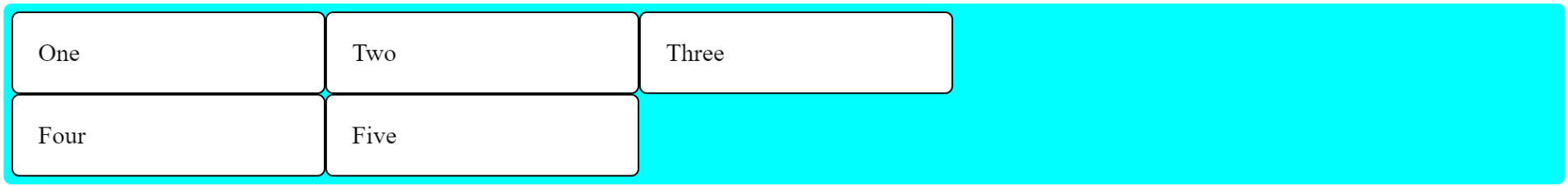
- In het volgende voorbeeld definiëren we een grid container met een grid bestaande uit drie kolommen van elk **200px** breed.
- Aangezien we niet expliciet het aantal en de hoogte van de rijen opgeven, zullen er automatisch rijen worden aangemaakt, waarbij ook de hoogte automatisch bepaald wordt.
- Er is in dit voorbeeld geen expliciete plaatsing van de grid items, waardoor de grid items in ‘HTML source order’ in de grid geplaatst zullen worden.

HTML

```html
<div class="grid-container">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

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



Grid 01: grid container (codepen.io)

grid-template-rows en grid-template-columns

- De waarde voor grid-template-columns (en grid-template-rows) kan een **<track-list>** zijn of een <auto-track-list>
- Mogelijke track sizes bij een **<track-list>** zijn:
  - <length>
  - <percentage>
  - <flex>
  - min-content
  - max-content
  - minmax(min,max)
  - fit-content(value)
  - auto

```
/* Voorbeelden
<track-list> */
grid-template-columns: 20ch 200px  200px;
grid-template-columns: 20% 60% 40%;
grid-template-columns: 100px 1fr 2fr;
grid-template-columns: minmax(100px, 200px);
grid-template-columns: fit-content(40%);
grid-template-columns: auto 100px 1fr;
```


HO GENT

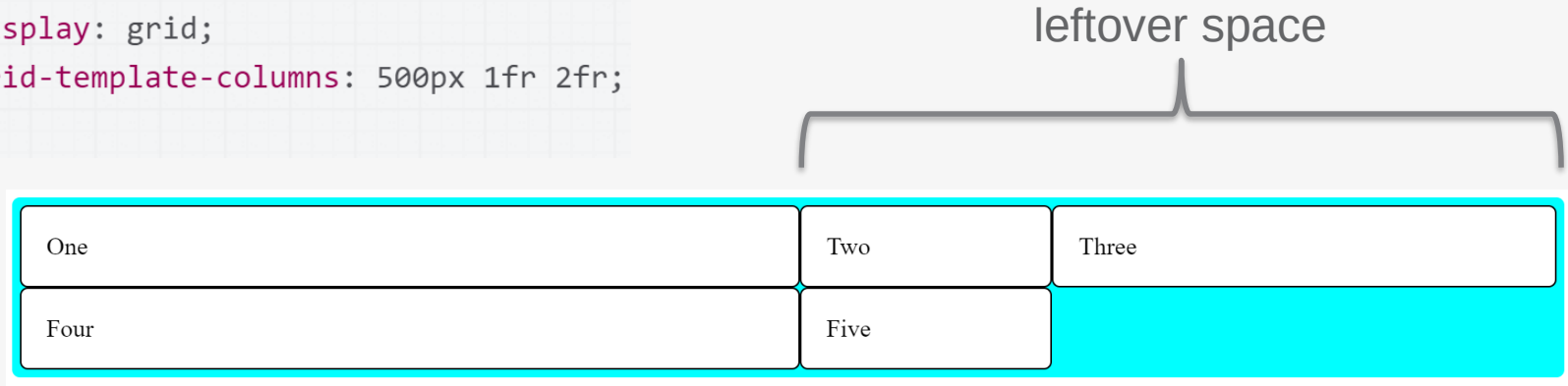
De fr-eenheid (<flex> CSS data type)

- Met de *fr-eenheid* kan je ‘flexible grid tracks’ (rijen of kolommen) maken.
- De *fr-eenheid* stelt een fractie voor van de ‘leftover space’. De ‘leftover space’ is de ruimte die niet ingenomen wordt door ‘non-flexible grid tracks’.

HO GENT

```
/* Voorbeeld met één non-flexible column track (500px) en twee
flexible column tracks.
De ‘leftover space’ wordt verdeeld over de tweede en de derde
kolom. Hiervoor wordt de ‘leftover space’ eerst in drie gedeeld
en hiervan krijgt de tweede kolom één deel en de derde kolom
twee delen.*/
```

```
.grid-container {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```



[Grid 02: fr-eenheid (codepen.io)]

```
/* Voorbeeld met één kolom met breedte van 2fr en twee kolommen met een breedte van 1fr. De beschikbare ruimte wordt dus in vier gedeeld. De eerste kolom krijgt twee delen en de twee overige kolommen elk één deel.*/
```

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


```
/* Voorbeeld met drie kolommen van gelijke breedte, die automatisch groeien of krimpen afhankelijk van de beschikbare ruimte*/
```

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

HO GENT

repeat()-functie

Met de repeat()-functie kan je een (deel van een) ‘track-list’ herhalen.

- Zo kan je

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

  korter schrijven als:

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

- Een meer uitgebreid voorbeeld:

```
grid-template-columns: 20px repeat(4, 1fr 2fr);
```

  is hetzelfde als:

```
grid-template-columns: 20px 1fr 2fr 1fr 2fr 1fr 2fr 1fr 2fr;
```


HO GENT

`grid-auto-rows` en `grid-auto-columns`

- In de vorige voorbeelden hebben we de kolommen (column tracks) expliciet gedefinieerd met de `grid-template-columns`-property en de rijen werden impliciet gecreëerd.
- Het instellen van de hoogte van deze impliciet gecreëerde rijen doe je via `grid-auto-rows`.
  (Voor de kolommen is er `grid-auto-columns`).
- De initial value voor deze properties is **auto**. Impliciet gecreëerde rijen (en kolommen) zijn dus standaard ‘auto-sized’.

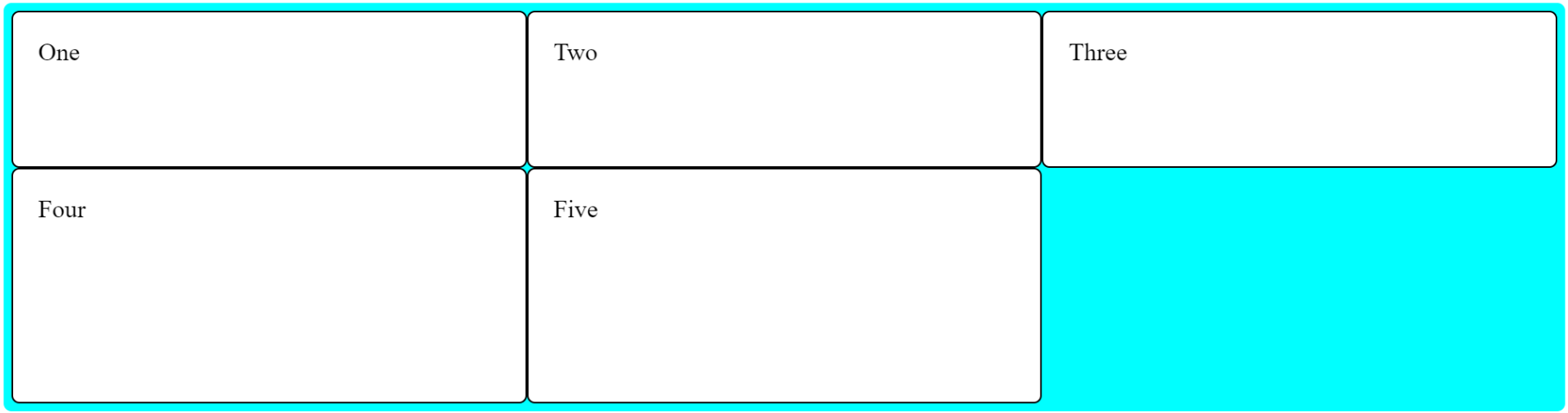
HO GENT

HTML

```html
<div class="grid-container">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

```css
/* de hoogte van de eerste rij stellen we in op 100px en eventuele bijkomende impliciet gecreëerde rijen krijgen een hoogte van 150px*/
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px;
  grid-auto-rows: 150px;
}
```



[Grid 03: implicit grid (codepen.io)]

`grid-auto-rows` en `grid-auto-columns`

- Bij **grid-auto-rows/grid-auto-columns** kunnen we dezelfde ‘track sizes’ gebruiken als bij **grid-template-rows/grid-template-colums**.
- Voorbeeld `minmax()`:

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

In dit voorbeeld zal de rijhoogte steeds aangepast worden aan de inhoud (`auto`), maar de rijhoogte zal nooit kleiner worden dan `100px`.

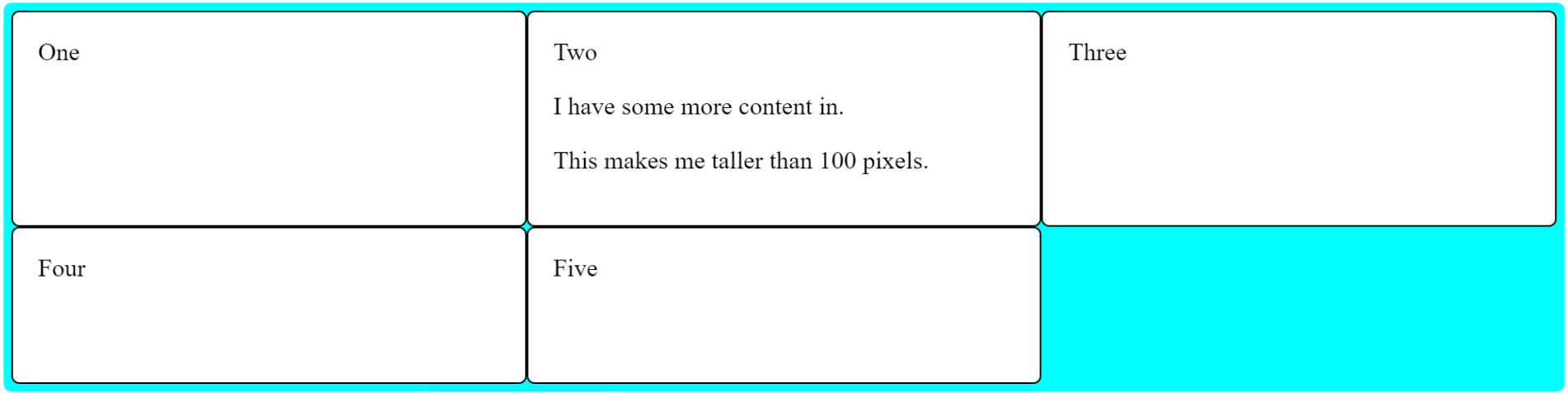
HO GENT

HTML

```html
<div class="grid-container">
 <div>One</div>
  <div>Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

```css
/* de rijhoogte van de rijen is minimaal 100px en maximaal de hoogte van de inhoud.*/
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);
}
```



Grid 04: minmax()-notatie (codepen.io)

`repeat()` met `auto-fill` en `auto-fit`

- De waarde voor **grid-template-columns** (en **grid-template-rows**) kan niet alleen een <track-list> zijn (zie vroeger), maar ook een **<auto-track-list>**.
- Bij een **<auto-track-list>** gaan we bij de `repeat()`-functie, als eerste argument i.p.v. een getal op te geven, de waarde `auto-fill` of `auto-fit` gebruiken.

HO GENT

Voorbeeld: repeat(auto-fill,…)

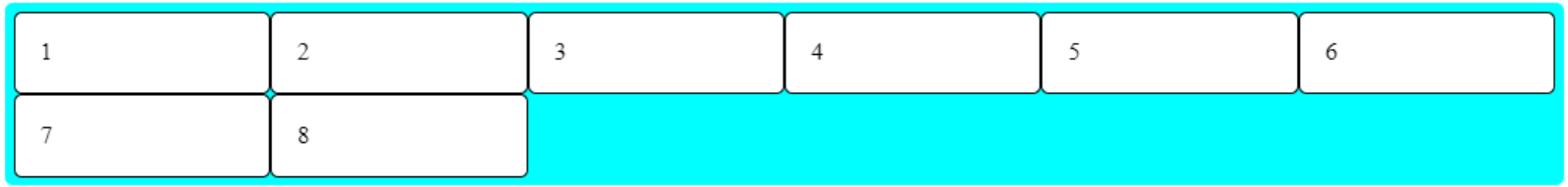
```
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```

We geven zelf geen vast aantal kolommen op bij de `repeat`()-functie maar laten de browser het aantal kolommen bepalen (`auto-fill`).
Voorbeeld: bij 800px is er niet genoeg ruimte voor 6 kolommen van 150px (6*150px=900px) en een kolom mag niet smaller zijn dan 150px, dus maakt de browser 5 kolommen.
Door het toevoegen van de max-waarde **1fr** wordt bovendien de overgebleven ruimte verdeeld over de 5-kolommen.

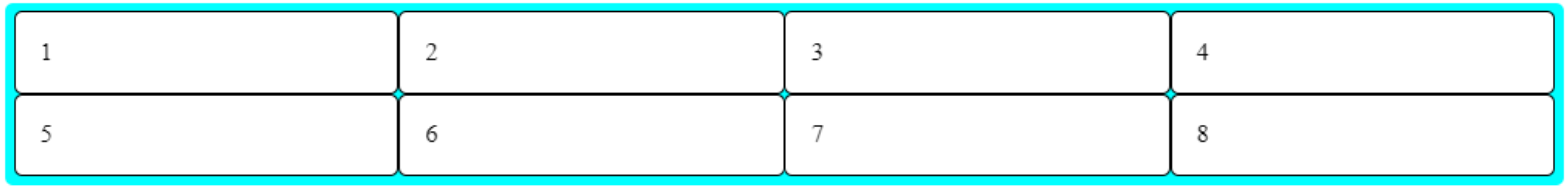
HO GENT

Voorbeeld: repeat(auto-fill,...)

```
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```




```
grid-template-columns: repeat(4, 1fr);
```



[Grid 05: repeat(auto-fill, ...) (codepen.io)]

Meer info over auto-fill en auto-fit

- `Auto-fill` en `auto-fit` werken op dezelfde manier, behalve als er lege tracks zijn.
- Een goede video over hoe `auto-fill` en `auto-fit` werken en wat het verschil is tussen de twee vind je op de website van https://gridbyexample.com van Rachel Andrew.

  Videolink:

  https://gridbyexample.com/video/series-auto-fill-auto-fit/

HO GENT

3. positioneren van grid items op de grid

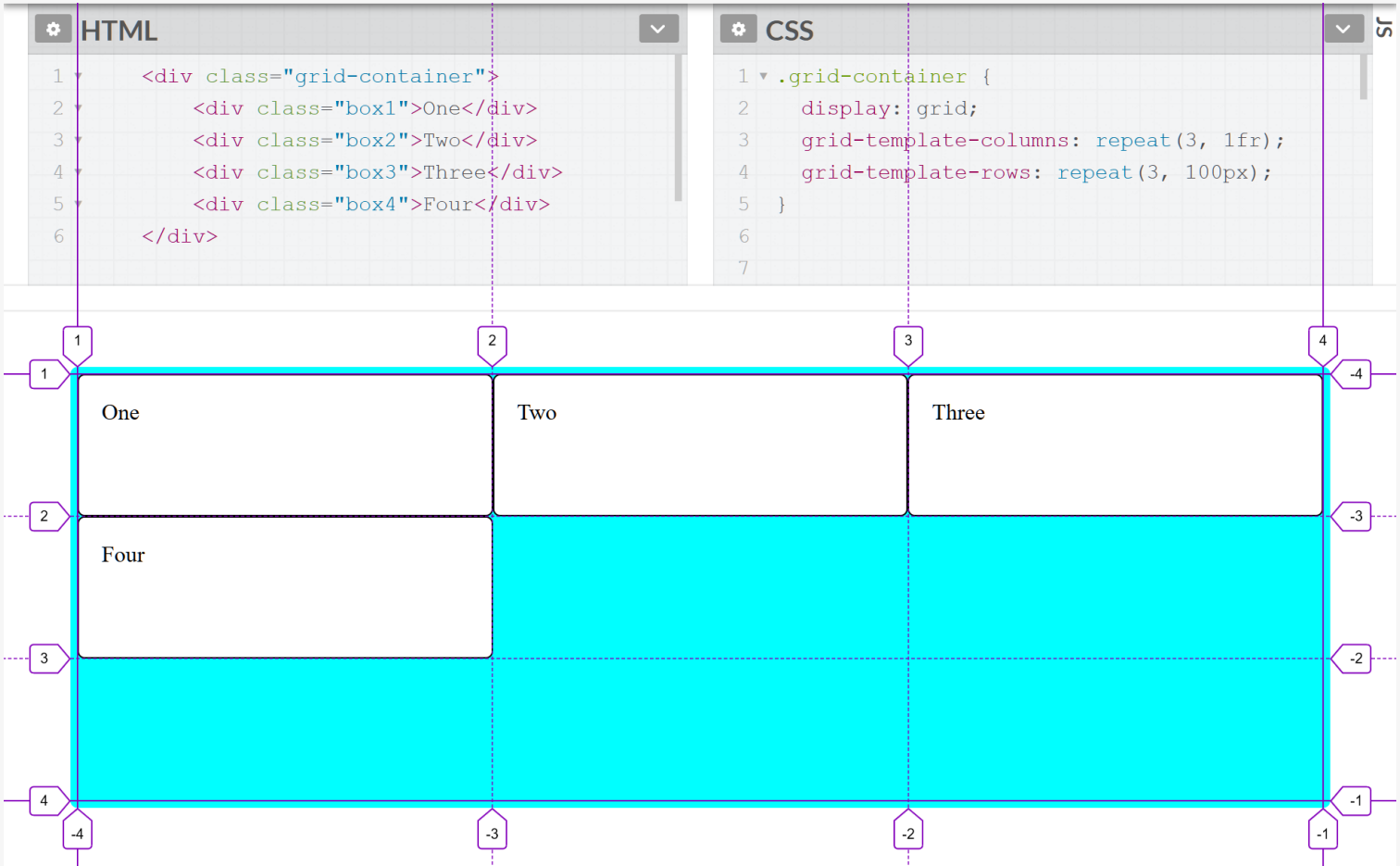
Tot nu toe hebben we enkel de grid, met andere woorden de grid structuur (de rijen en de kolommen) gedefinieerd. We gaan nu kijken naar de mogelijkheden om de grid items te positioneren op de grid. Hierbij kunnen grid items ook meerdere rijen en kolommen overspannen.

Grid Lines

- Als je een grid definieert dan krijgen de lijnen waaruit de grid bestaat automatisch nummers. Je kan deze lijnnummers gebruiken om een ‘grid item’ expliciet te positioneren op de grid.
- De Grid Inspector in Mozilla Firefox is een goede tool om te leren werken met grid. Op de volgende dia zijn via de *Grid Inspector* in de *Firefox Developer Tools* de *Grid lines* en hun *Line numbers* zichtbaar gemaakt.
- De getallen verwijzen naar de lijnen en niet naar de kolommen of de rijen. Zo bevat een 3-column-layout 4 lijnen.
- Naast de positieve lijnnummers zijn er ook negatieve lijnnummers, waarbij er achterwaarts geteld wordt.

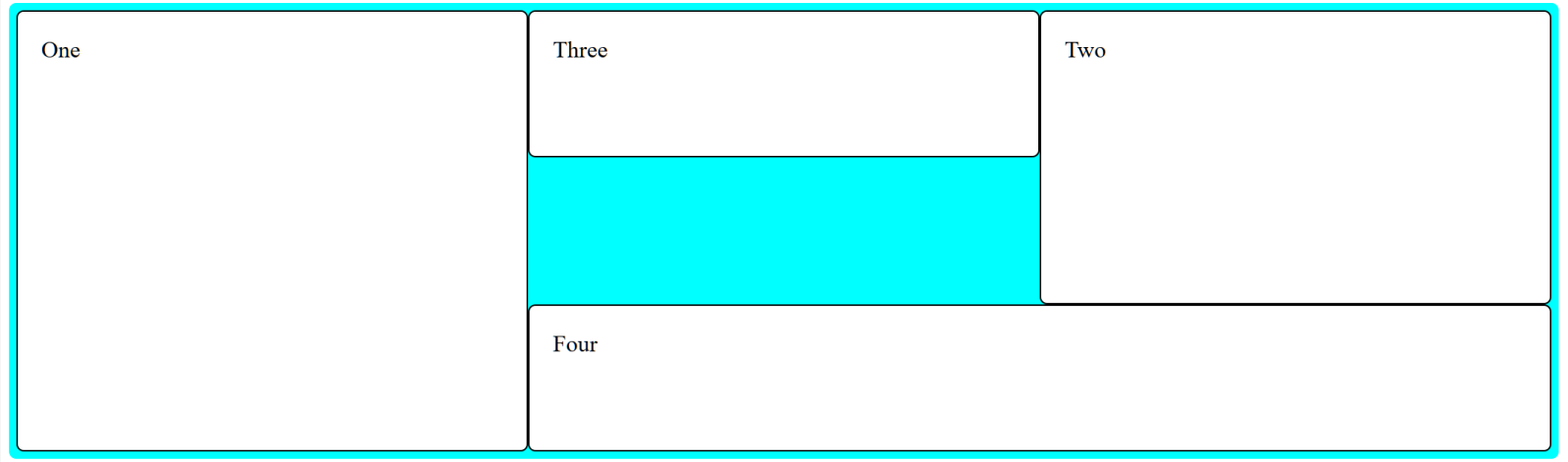
HO GENT

Afbeelding met Grid Lines en Line Numbers zichtbaar via de Firefox Grid Inspector



Grid Items positioneren op de Grid met Line Numbers

```
.box1 {grid-column-start: 1;grid-column-end: 2;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-column-end: 4;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-column-end: 3;grid-row-start: 1;grid-row-end: 2;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;grid-row-end: 4;}
```



[Grid 06: position grid Items with grid lines (codepen.io)]

Default Spans

- In het vorige voorbeeld werd elke ‘grid-row-end’ en ‘grid-column-end’ expliciet vermeld. Als het item echter maar één track (rij of kolom) overspant, mag je de ‘grid-row-end’ of ‘grid-column-end’ waarden weglaten. Het vorige voorbeeld kan dus ook geschreven worden als:

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```


HO GENT

Het sleutelwoord ‘span’

- In plaats van het ‘end’ lijnnummer op te geven, kan je ook het aantal tracks opgeven dat je wilt overspannen.
- Voorbeeld:

```
grid-row-start: 1;grid-row-end: 4;}
```

kan je dus ook schrijven als

```
grid-row-start: 1;grid-row-end: span 3;}
```

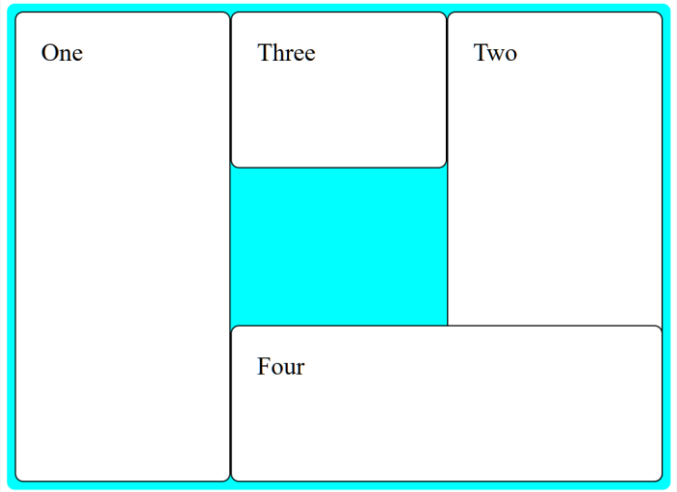

HO GENT

Overlappende grid items

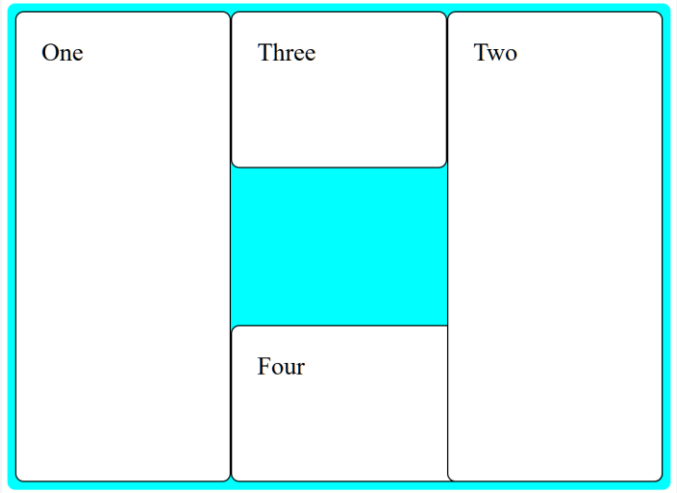
- Grid items kunnen overlappen.
- Als we in het voorbeeld op de volgende dia `.box2` laten eindigen bij rijlijn 4 i.p.v. bij rijlijn 3, dan zit `.box2` gedeeltelijk verscholen achter `.box4`. Wensen we `.box2` naar de voorgrond te brengen dan moeten we de `z-index`-property aanpassen.

HO GENT

```css
.box2 {
  grid-column-start: 3;
  grid-row-start: 1;
  grid-row-end: 4;
}
```




```css
.box2 {
  grid-column-start: 3;
  grid-row-start: 1;
  grid-row-end: 4;
  z-index: 1;
}
```



Grid items plaatsen

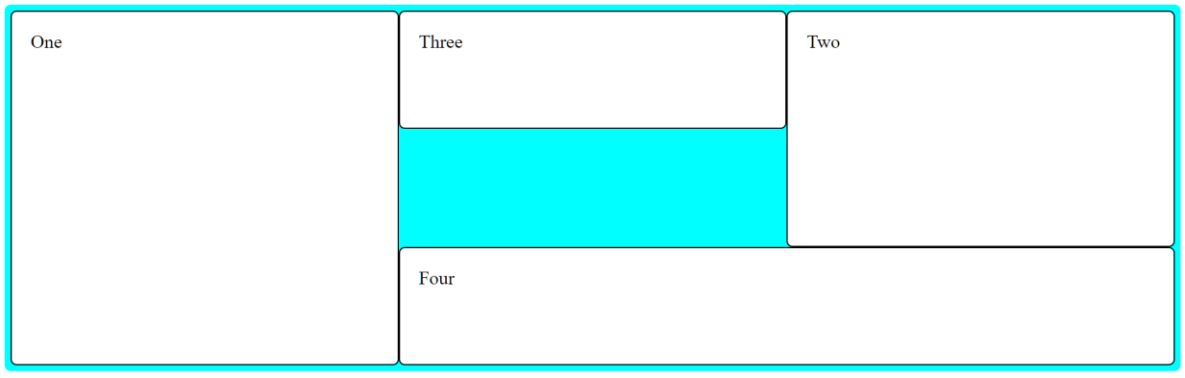
- Om *grid items* in de *grid* te plaatsen bestaan er naast de grid-placement properties `grid-column-start`, `grid-column-end`, `grid-row-start` en `grid-row-end` ook de shorthands `grid-column`, `grid-row` en `grid-area`.

grid-area			
grid-column		grid-row	
grid-column-start	grid-column-end	grid-row-start	grid-row-end

Bron: https://www.w3.org/TR/css-grid-1/#common-uses

- Op de volgende dia's vind je, hoe je Voorbeeld 06 korter kan schrijven met deze properties. Meer info over het plaatsen van grid items vind je o.a. in de specification: https://www.w3.org/TR/css-grid-1/#placement

HO GENT

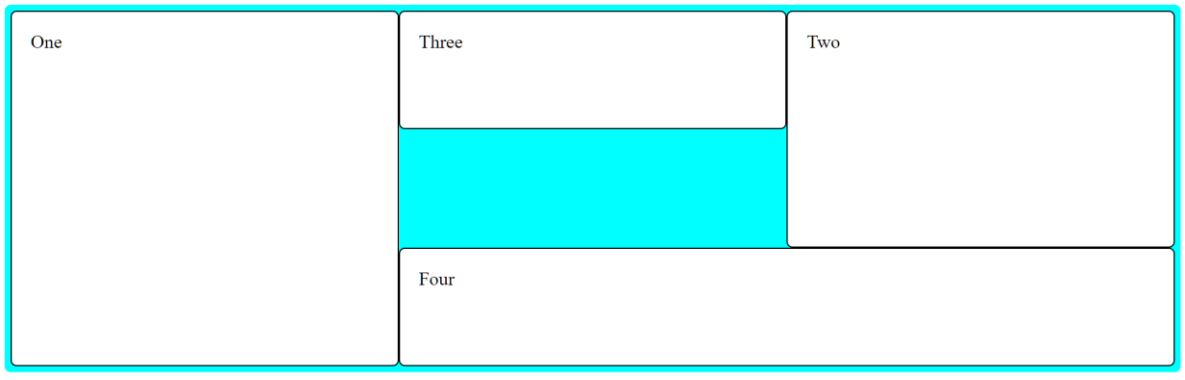


Voorbeeld 06…

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```

…kan korter geschreven worden met de `grid-column` en `grid-row` shorthand properties

```
.box1 {grid-column: 1;grid-row: 1 / 4;}
.box2 {grid-column: 3;grid-row: 1 / 3;}
.box3 {grid-column: 2;grid-row: 1;}
.box4 {grid-column: 2 / 4;grid-row: 3;}
```



Voorbeeld 06…

```
.box1 {grid-column-start: 1;grid-row-start: 1;grid-row-end: 4;}
.box2 {grid-column-start: 3;grid-row-start: 1;grid-row-end: 3;}
.box3 {grid-column-start: 2;grid-row-start: 1;}
.box4 {grid-column-start: 2;grid-column-end: 4;grid-row-start: 3;}
```

…kan ook korter geschreven worden met de `grid-area` property

```
.box1 {grid-area: 1 / 1 / 4 / 2;}
.box2 {grid-area: 1 / 3 / 3 / 4;}
.box3 {grid-area: 1 / 2 / 2 / 3;}
.box4 {grid-area: 3 / 2 / 4 / 4;}
```

```
/* De volgorde van de waarden
voor de grid-area property is:
  grid-row-start
  grid-column-start
  grid-row-end
  grid-column-end */
```

Achterwaarts tellen

- Het is ook mogelijk om achterwaarts te tellen bij het opgeven van de lijnnummers. Zo kan je in Voorbeeld 06 het lijnnummer 4 vervangen daar lijnnummer -1 en kan

```
.box1 {grid-column:1; grid-row: 1 / 4;}
```

ook geschreven worden als

```
.box1 {grid-column:1; grid-row: 1 / -1;}
```


HO GENT

Gutters

- Witruimte tussen kolommen en rijen, genoemd ‘Gutters’, creëer je met `column-gap` en `row-gap` of de shorthand `gap`.
- **Opmerking** Sommige browsers gebruiken nog de oude notaties met de prefix `grid-` nl. `grid-column-gap`, `grid-row-gap` en `grid-gap`, maar de meeste browser-creators hebben hun browser reeds aangepast.

HO GENT

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 50px 50px;
  column-gap: 10px;
  row-gap: 1rem;
}
```

Grid 07: gutters (codepen.io)

Grid-template-areas property

HO GENT

grid-template-areas property

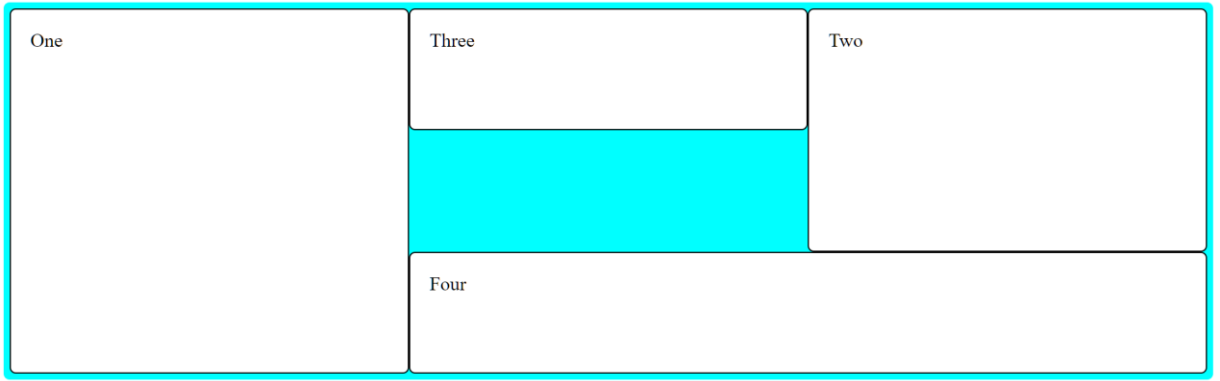
- Een andere manier om grid items in de grid te plaatsen is met de `grid-template-areas` property. Hiermee creëer je namen voor grid areas die je dan kan gebruiken in bijv. de `grid-area` property om grid items te plaatsen in een grid area.
- Op de volgende dia vind je Voorbeeld 06 herschreven met `grid-template-areas`. Merk op dat
  - er een rij gecreëerd wordt voor elke string.
  - je meerdere rijen of kolommen kan overspannen door de naam te herhalen;
  - je voor cellen zonder naam een punt `.` gebruikt.
  - de waarde van `grid-template-areas` de structuur van de grid visualiseert.

HO GENT

Voorbeeld 06 herschreven met de `grid-template-areas` property

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 100px);
  grid-template-areas:
  "one three two"
  "one .     two"
  "one four  four";
}


.box1 {grid-area: one;}
.box2 {grid-area: two;}
.box3 {grid-area: three;}
.box4 {grid-area: four;}
```



[Grid 08: grid-template-areas (codepen.io)]

Opmerking ‘Named Grid Lines’

- Je kan niet alleen namen geven aan grid areas, maar ook aan grid lines. Het werken met 'Named Grid Lines’ zullen we echter niet bespreken. Wie hierin toch geïnteresseerd is zie: MDN - Layout using named grid lines

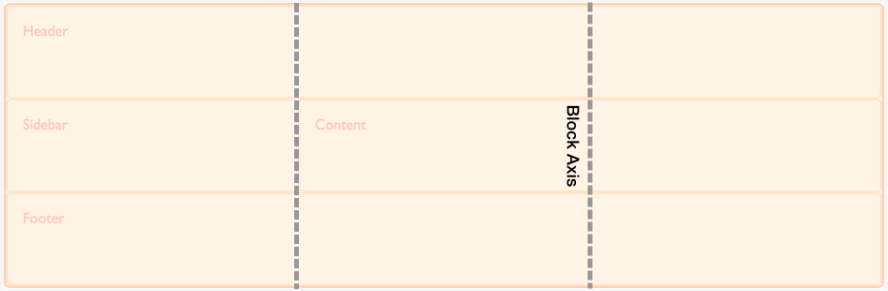
HO GENT

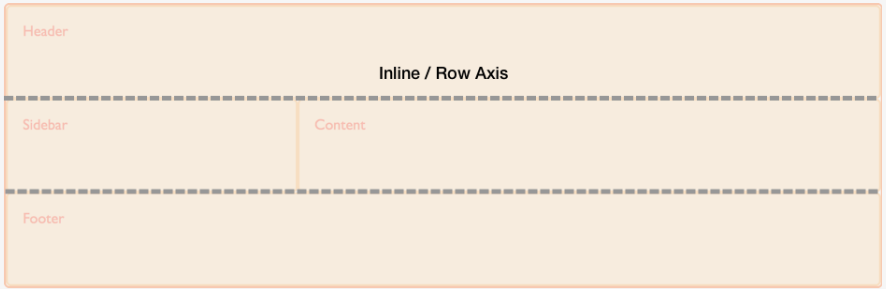
Box alignment in CSS Grid Layout

BRON: MDN Box alignment in CSS Grid Layout (https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Box_Alignment_in_CSS_Grid_Layout)

HO GENT

Box alignment in CSS Grid Layout





Om items uit te lijnen langs de *block axis* gebruik je de properties align-items en align-self. Items uitlijnen langs de *inline axis* doe je met justify-items and justify-self.

HO GENT

Theoriebestanden (voorbeelden)

- Open in Visual Studio Code de map **02-css-grid-box-alignment**

HO GENT

We gebruiken de voorbeelden van de MDN-website. Deze voorbeelden gebruiken allemaal de volgende basis HTML en CSS …

HTML

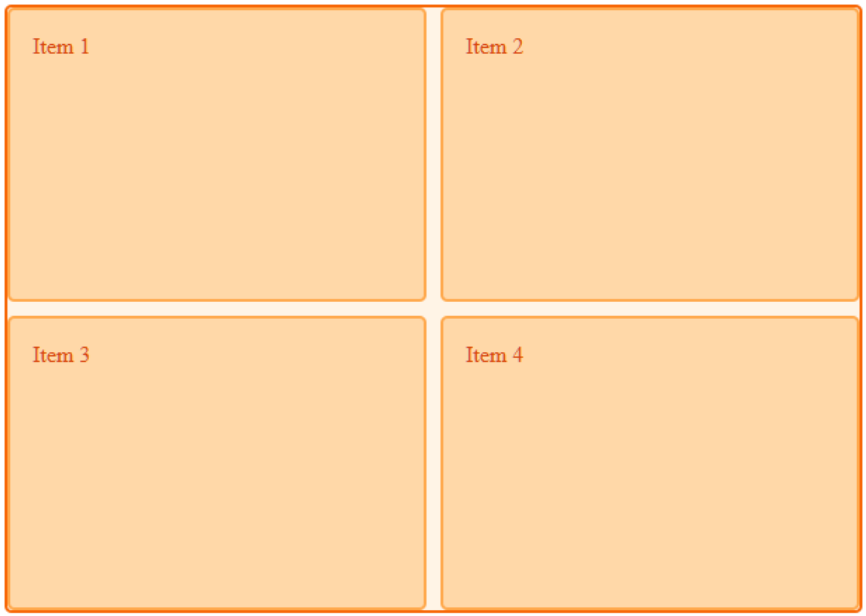
```
<div class="wrapper">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div>
```

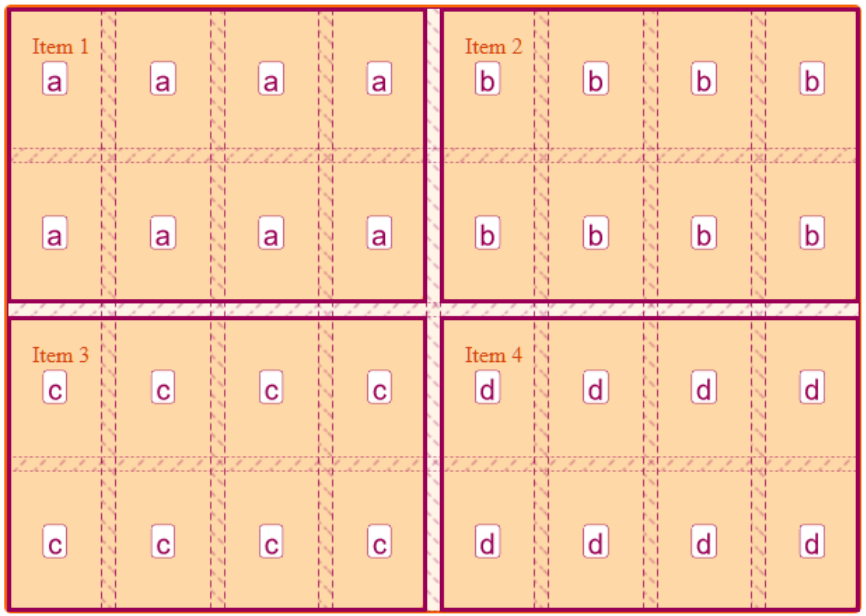
CSS

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(8, 1fr);
  grid-gap: 10px;
  grid-auto-rows: 100px;
  grid-template-areas:
    "a a a a b b b b"
    "a a a a b b b b"
    "c c c c d d d d"
    "c c c c d d d d";
}

.item1 {grid-area: a;}
.item2 {grid-area: b;}
.item3 {grid-area: c;}
.item4 {grid-area: d;}
```

… met het onderstaande als resultaat. In de rechtse afbeelding is de Firefox Grid Inspector gebruikt om de grid te visualiseren.
In dit startvoorbeeld zijn er geen expliciete Box alignment properties ingesteld;





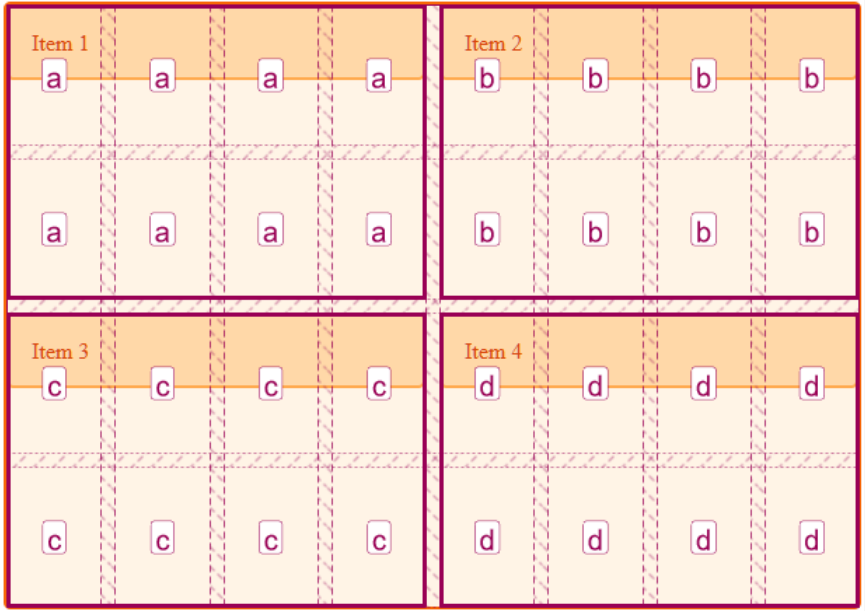
Block axis: align-self en align-items

- De `align-self` property stelt de uitlijning in van een grid item binnen zijn grid area/cell langs de block axis.
- De default voor `align-self` komt overeen met `stretch`.
  Bij elke andere waarde worden de afmetingen van het item automatisch berekend zodat de inhoud er net in past.
- De `align-items` property stelt de `align-self` property in voor alle grid-items. De `align-items` property stel je in op de grid container.

HO GENT

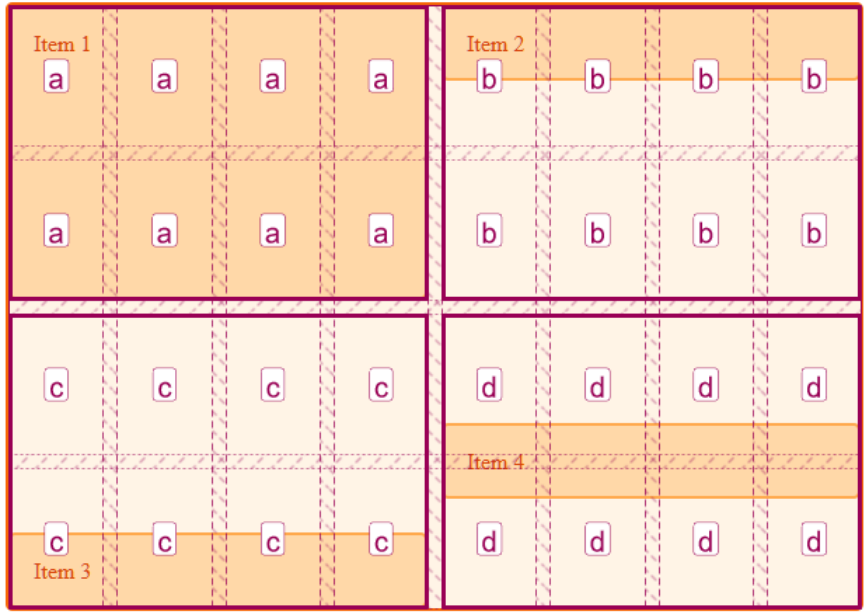
Voorbeeld 1a: items uitlijnen langs de block axis met `align-items`

```
.wrapper {
  align-items: start;
}
```



Voorbeeld 1b: items uitlijnen langs de block axis met `align-self`

```
.item2 {align-self: start;}
.item3 {align-self: end;}
.item4 {align-self: center;}
```



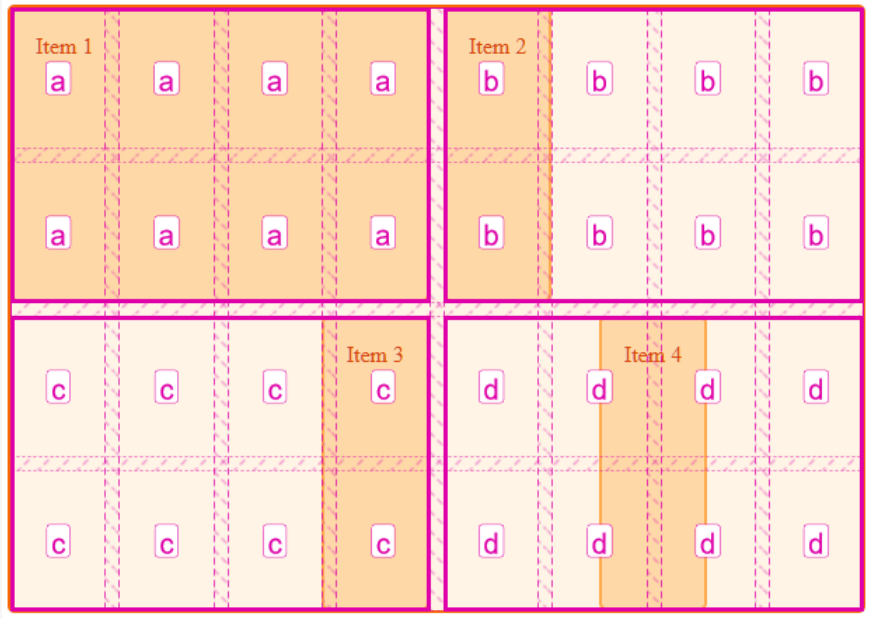
Inline axis: justify-self en justify-items

- De `justify-self` property stelt de uitlijning in van een grid item binnen zijn grid area/cell langs de inline axis.
- De default voor `justify-self` komt overeen met `stretch`.
  Bij elke andere waarde worden de afmetingen van het item automatisch berekend zodat de inhoud er net in past.
- De `justify-items` property stelt de `justify-self` property in voor alle grid items.
  De `justify-items` property stel je in op de grid container.

HO GENT

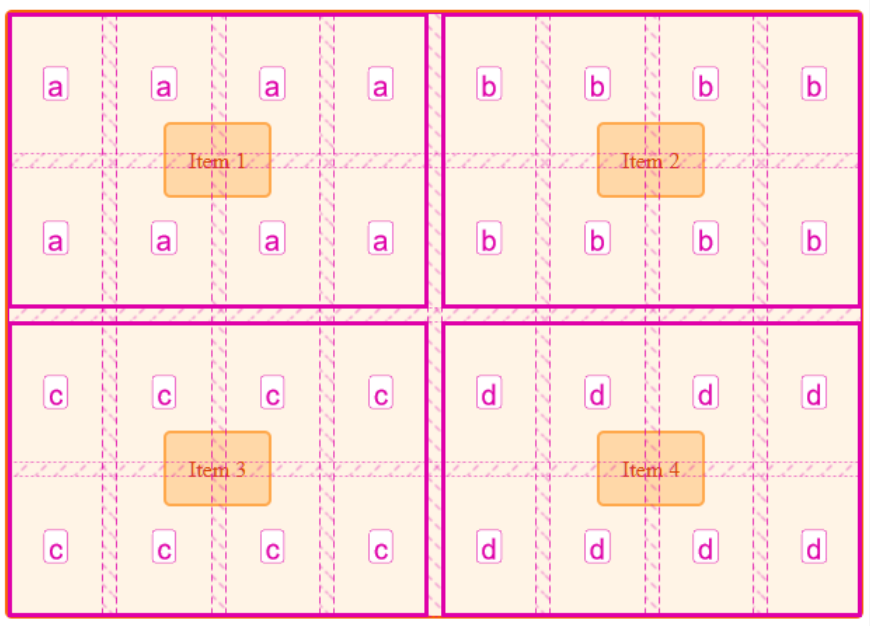
Voorbeeld 2: items uitlijnen langs de inline axis
met `justify-self`

```
.item2 {justify-self: start;}
.item3 {justify-self: end;}
.item4 {justify-self: center;}
```



Voorbeeld 3: items centreren
met `align-items` en `justify-items`

```
.wrapper {
  align-items: center;
  justify-items: center;
}
```



align-content en justify-content

De align-content en de justify-content properties worden gebruikt als de grid tracks (rijen of kolommen) niet de volledige grid container innemen. Je kan in dit geval de tracks zelf uitlijnen binnen de container.

HO GENT

Extra info (geen examenstof)

Nog enige extra info in verband met CSS Grid:

- Ook voor de **grid-template-rows**, **grid-template-columns** en **grid-template-areas** zijn er shorthands, namelijk grid en grid-template.
- Het artikel MDN - Auto-placement in CSS Grid Layout bevat onder andere info over het gebruik van grid-auto-flow.
- CSS Grid level 2 (Working draft) bevat de definitie van subgrid, maar momenteel is dit enkel geïmplementeerd in Firefox (2021-10-25)