User Interfaces 1
CSS

Layout: weergavemodel

KdG Karel de Grote Hogeschool

Weergavemodel

("visual presentation model")

- De natuurlijke flow (~ normal flow)
- Grid flow (~ CSS Grid)
- Flexbox flow (~ CSS Flexbox)

De natuurlijke flow (~ normal flow)

("visual presentation model")

- **Block level elements:**
  - Worden onder elkaar geplaatst
  - Je kan de grootte instellen (en border, padding, margin)
  - Voorbeelden: `h1, p, li, section, …`
- **Inline level elements:**
  - Worden naast elkaar geplaatst
  - Je kan de grootte niet instellen: ze hebben de minimale grootte obv de inhoud/content
  - Voorbeelden: `img, span, input, button, …`

⟶ de standaard/default waarde van de eigenschap `display` bepaalt of een element 'block' of 'inline' is

display: mogelijke waardes

- `display:block;`
  - Elementen worden onder elkaar geplaatst.
  - Elementen hun hoogte en breedte kan ingesteld worden.
- `display:inline;`
  - Elementen worden naast elkaar geplaatst.
  - De breedte en hoogte kan niet ingesteld worden.
- `display:inline-block;` → soort "tussenin"
  - Elementen worden naast elkaar getoond.
  - Breedte en hoogte **wel** instelbaar!
- `display:none;`
  - Element wordt niet getoond.

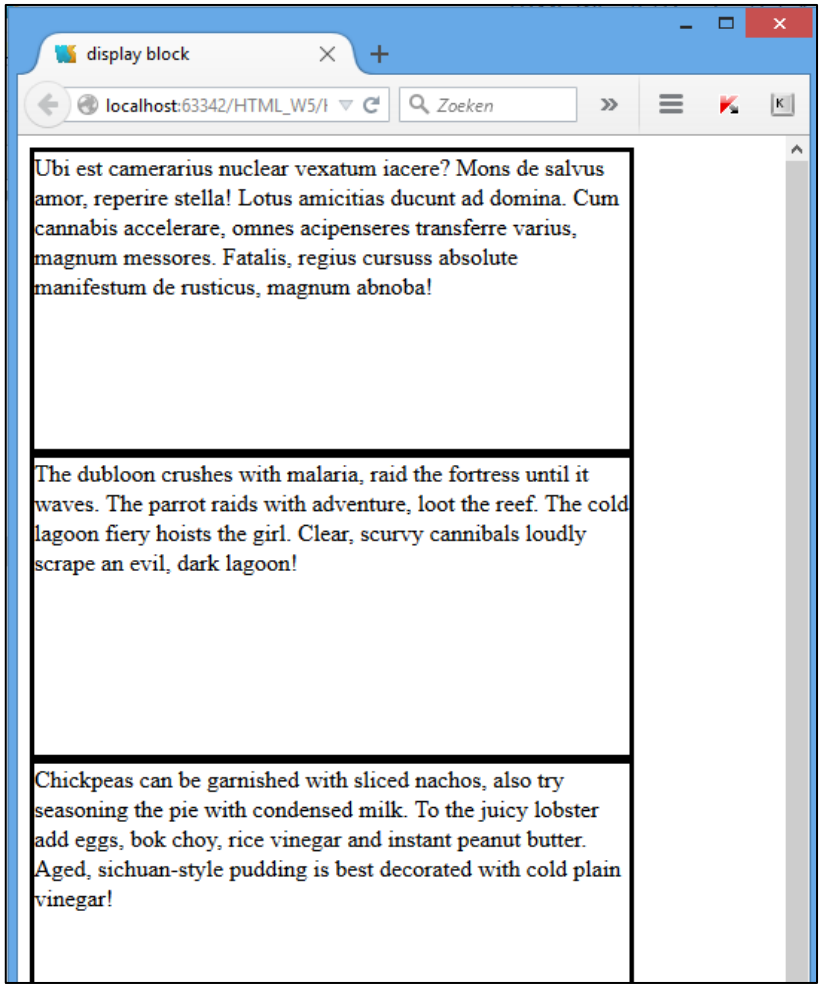
Er bestaan nog andere waardes:
Zie http://www.w3schools.com/cssref/pr_class_display.asp

display: block;

```
section {
    display: block;
    width: 400px;
    height: 200px;
    border: solid;
}
```

Een block-level element neemt standaard de beschikbare ruimte (~breedte) in zijn ouderelement in, maar je kan de breedte (en de hoogte) instellen.
De elementen komen onder elkaar te staan (behalve in geval van 'floating').

Een **section** is default een block-level element, dus `display:block` hoefde hier niet te staan.

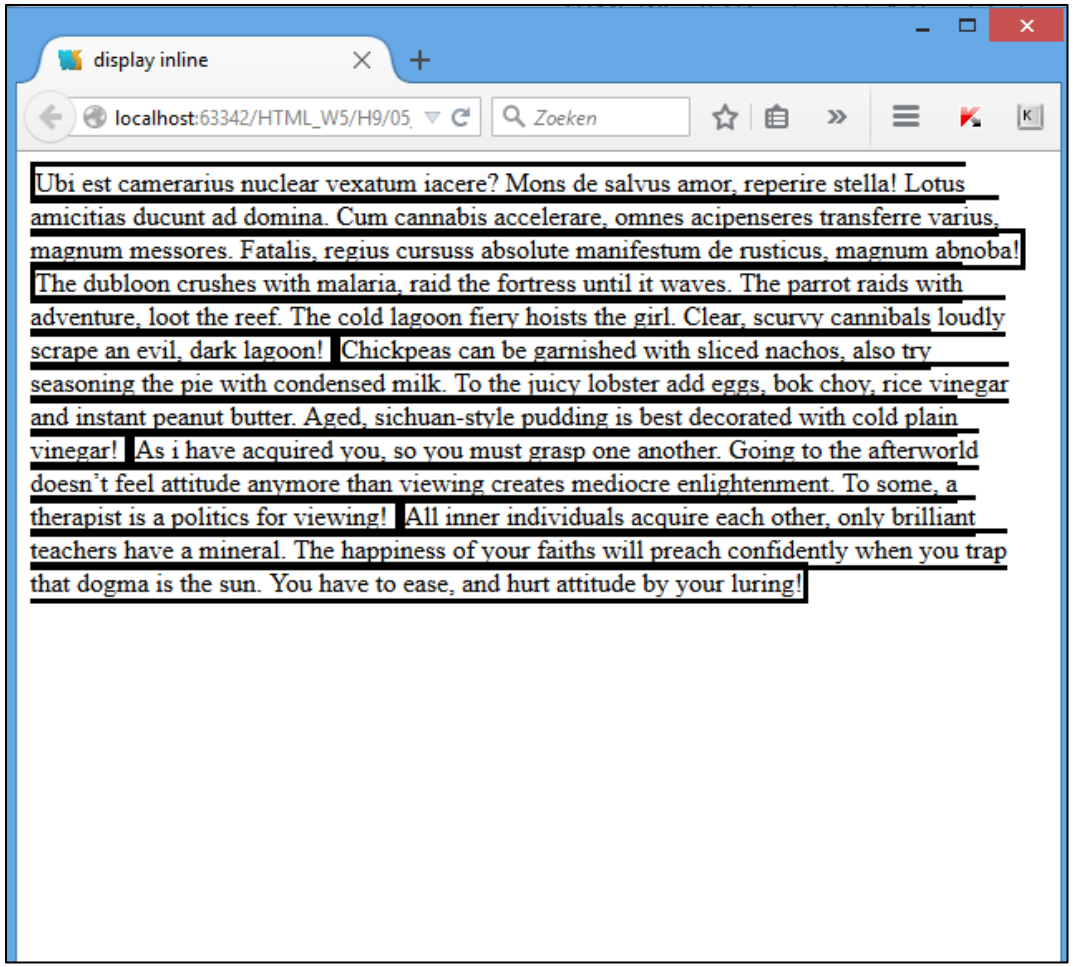


Voorbeeld: H9/06_display_block

display: inline;

```
section {
    display: inline;
    width: 400px;
    height: 200px;
    border: solid;
}
```

Maak je de sections **inline**, dan komen ze als één lange tekst achter elkaar te staan. De breedte en hoogte die je in de css instelt worden genegeerd.

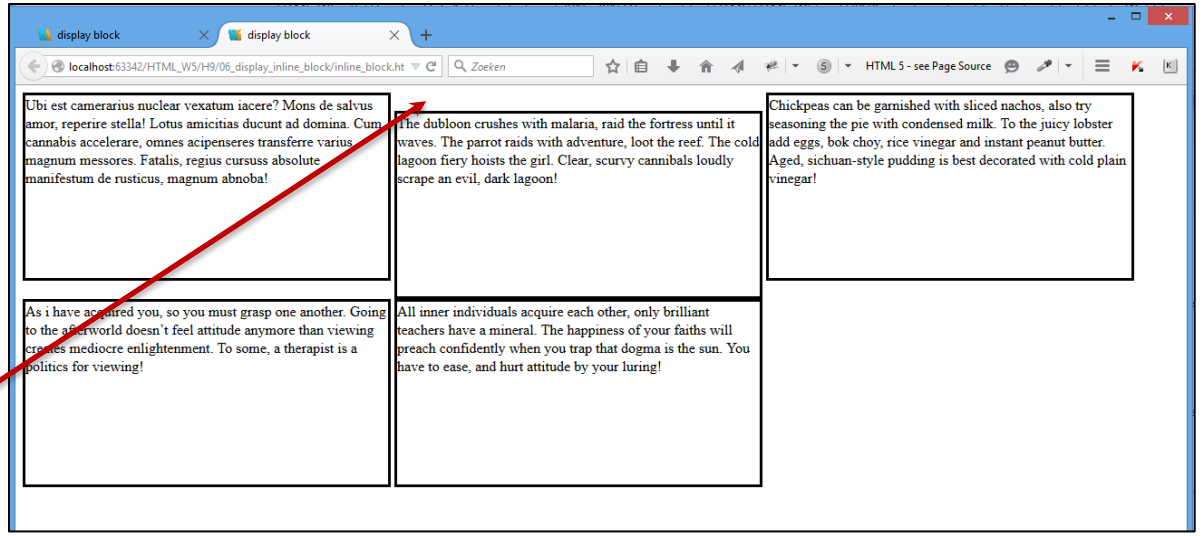


Voorbeeld: H9/07_display_inline

display: inline-block;

```
section {
    display: inline-block;
    width: 400px;
    height: 200px;
    border: solid;
}
```

inline-block zorgt ervoor dat de elementen wel hun grootte behouden, en toch naast elkaar komen te staan. Dit lijkt (en is ook vaak) de ideale oplossing voor veel lay-out problemen, maar toch heb je soms ook hier vreemd gedrag (waarom staat dat tweede paragraafje een beetje lager bijvoorbeeld?). Goed testen is de boodschap!



Voorbeeld: H9/08_display_inline_block

overflow

- Als de box niet groot genoeg is om de inhoud te bevatten, wat moet er dan gebeuren?
  - `overflow:visible;` → inhoud gewoon tonen (kan dus overlappen). Dit is de default.
  - `overflow:hidden;` → alles wat buiten de box valt wordt weggelaten
  - `overflow:scroll;` → we tonen altijd scrollbars
  - `overflow:auto;` → we tonen alleen scrollbars indien nodig

→ Opmerking: Je kan ook apart werken met `overflow-x` en `overflow-y` …

`visibility`

- Geeft de mogelijkheid om een element onzichtbaar te maken, maar in tegenstelling tot **`display:none`** wordt de ruimte van het element wel gereserveerd

  - `visbility:visible;` → element is zichtbaar (standaard)
  - `visibility:hidden;` → element wordt visueel verborgen

Grid flow

CSS Tricks: A Complete Guide to Grid
https://css-tricks.com/snippets/css/complete-guide-grid/

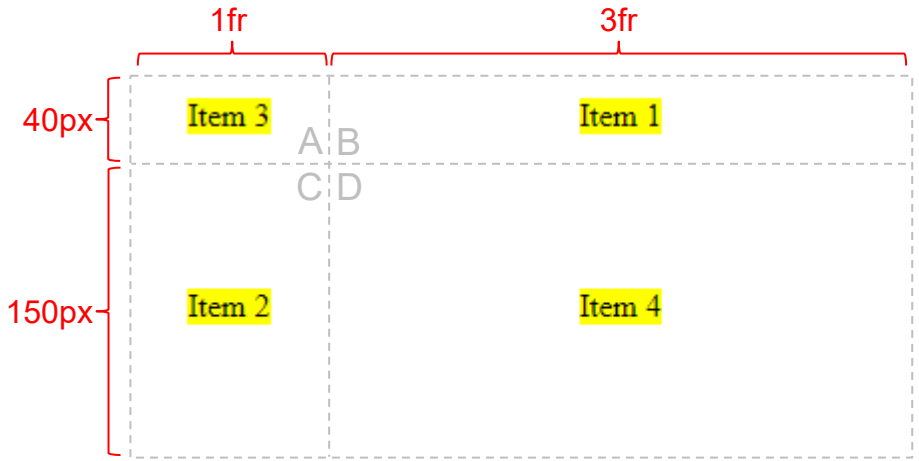
Dit is een 2 dimensionale lay-out structuur, waarbij je voor het grid-element (~container) de directe kinderelementen (~items) in deze structuur plaatst (= 'content placement').

```
ul {
    display: grid;
    grid-template-areas: 'A B'
                         'C D';
    grid-template-columns: 1fr 3fr;
    grid-template-rows: 40px 150px;
    justify-content: center;
    align-items: center;
}

li:nth-child(1) { grid-area: B; }
li:nth-child(2) { grid-area: C; }
li:nth-child(3) { grid-area: A; }
li:nth-child(4) { grid-area: D; }

ul { list-style: none;
     margin: 0; padding: 0; }
li { background-color: yellow; }
```

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
</ul>
```



Flexbox flow

CSS Tricks: A Complete Guide to Flexbox
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Dit is een 1 dimensionale flow structuur, waarbij je voor het flexbox-element (~container) de directe kinderelementen (~items) in een basis richting laat plaatsen (= 'content flow').

```
ul {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    align-items: center;
    gap: 40px;
}

li:nth-child(1) { height: 35px; }
li:nth-child(2) { height: 20px; }
li:nth-child(3) { height: 50px; }
li:nth-child(4) { height: 30px; }

ul { list-style: none;
     margin: 0; padding: 0; }
li { display: inline-block;
     background-color: yellow; }
```

```
<ul>
    <li>Item 1 content</li>
    <li>Item 2</li>
    <li>Item 3 other content</li>
    <li>Item 4 more content</li>
</ul>
```