Web technology

Les 9: CSS - flexbox - tabellen

Kristof Michiels

In deze presentatie

- CSS flexbox
- Werken met tabellen

CSS flexbox

Flexbox

- Module heet voluit CSS Flexible box layout module
- Zorgt voor controle over het plaatsen van elementen op één as (horizontaal of verticaal)
- Geschikt voor menubalken en andere navigatie, produktlijstjes, afbeeldingsgalerijen, ...

Flexbox

- Elementen kunnen uitrekken of krimpen binnen hun containers
- Zo verhinderen we teveel of te weinig ruimte: super voor responsive web design
- Andere mogelijkheden:
  - Alle elementen zelfde hoogte te geven
  - Eenvoudig horizontaal of verticaal centreren
  - Volgorde van de elementen te veranderen (los van HTML-volgorde)

Een flexbox container aanmaken

- Flexbox is - net als inline, block en grid - een layout mode
- Een flexbox container aanmaken doe je dus met *display:flex*
- Op de volgende slide staat een basisvoorbeeld:
  - Standaard worden de divs als block elementen onder elkaar geplaatst
  - Als we de flexbox aanzetten gaan ze automatisch horizontaal in één rij staan
  - De elementen worden standaard in een rij van links naar rechts geplaatst

Een flexbox container

```
<div class="een">1</div>
<div class="twee">2</div>
<div class="drie">3</div>
<div class="vier">4</div>
<div class="vijf">5</div>
```

```
body {
    display: flex;
}
```

De "flow" controleren met *flex-direction*

- Je kan binnen een flex-container controleren op welke manier de binnenste elementen geplaatst worden
- De opties die we hier hebben worden aangestuurd door de *flex-direction* eigenschap
- De waarden die we kunnen gebruiken: row (standaard) | column | row-reverse | column-reverse
- Bij *row* is de as horizontaal, bij *column* is de as verticaal

```
body {
    display: flex;
    flex-direction: column;
    }
```

Uitsmeren over meerdere lijnen met *flex-wrap*

- Met *flex-wrap* bepaal je of de flexbox zich over meerdere lijnen mag uitstrekken
- De waarden die we kunnen gebruiken: nowrap (standaard) | wrap | wrap-reverse
- Standaard wordt alles op één lijn gezet

```
body {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
```

Flex-direction en *flex-wrap* verkort...

- Doe je met flex-flow
- Waarden: flex-direction flex-wrap

```
div.mijndiv {
    display: flex;
    height: 35rem;
    flex-flow: column wrap;
}
```

Uitlijnen van elementen in de flexbox container

- Uitlijnen op de hoofdas doen we met *justify-content*
- Je past het toe op de flex-container
- Waarden: flex-start (standaard) | flex-end | center | space-between | space-around

```
div.mijndiv {
    display: flex;
    justify-content: flex-start;
}
```

Uitlijnen op de kruis-as

- Hiermee bedoelen we in de hoogte (bij row) en links/rechts (bij column)
- Uitlijnen op de kruis-as doen we met *align-items*
- Je past het toe op de flex-container
- Waarden: stretch (standaard) | flex-start | flex-end | center | baseline

```
div.mijndiv {
    display: flex;
    align-items: center;
}
```

Individuele elementen uitlijnen

- Doe je op de kruis-as met *align-self*
- Waarden: flex-start | flex-end | center | baseline | stretch (standaard)

```
div {
    align-self: flex-start;
}
```

Meerdere flexlijnen uitlijnen

- Doe je met align-content
- Je bepaalt hiermee hoe je meerdere flexlijnen uitspreid over de kruis-as
- Waarden: flex-start | flex-end | center | space-around | space-between | stretch (standaard)

```
body {
    display: flex;
    flex-direction: row;
    height: 60vh;
    align-content: center;
    flex-wrap: wrap;
    border: 0.1rem solid grey;}
```

Flexbox op de navigatiebalk

```
<nav>
    <ul>
          <li class="logo">My logo</li>
          <li><a href="#">Navigation 1</a></li>
          <li><a href="#">Navigation 2</a></li>
          <li><a href="#">Navigation 3</a></li>
    </ul>
</nav>
```

Flexbox op de navigatiebalk

```
ul {
    display: flex;
    align-items: center;
    background-color: teal;
    padding: .5em;
    margin: 0;
    color: white;
    height: 3rem;}
li {
    margin: 0 1rem;}
li.logo {
    margin-right: auto;}
```

Bepalen hoe elementen "flexen" in de container

- We kunnen hiermee automatisch elementen laten groeien en krimpen
- Doen we met de flex-eigenschap
- Deze eigenschap wordt op de elementen gezet, niet op de container (kan individueel)
- Waarden: none | 'flex-grow flex-shrink flex-basis'
- Standaard: 0 1 auto;
- 1 staat voor "zet aan", 0 voor "zet uit"
- De flex-basis: zet de startafmeting. Je kan dit ook gebruiken om de width te zetten
- Je kan bepaalde elementen grotere waarden meegeven

Shortcuts voor de flex eigenschap

- *flex:initial* : zelfde als flex: 0 1 auto; Elementen zullen niet automatisch groeien, maar wel krimpen
- *flex: auto* : zelfde als flex: 1 1 auto; Elementen zullen groeien en krimpen
- *flex: none* : zelfde als flex: 0 0 auto; Niet krimpen en niet groeien
- *flex: getal* : zelfde als flex: getal 1 0px; Absolute flex. Getal bepaalt hoeveel vrije ruimte het element krijgt

De volgorde van flex elementen aanpassen met order

- Soms handig bij responsive web design
- Vooral nuttig indien je flexbox gebruikt voor layout en wij gaan dat zelden doen
- Je gebruikt hiervoor de eigenschap order
- Werkt op zelfde manier als z-index bij positioning
- Elk element heeft standaard 0. Je kiest getal zelf. Negatief getal kan ook

```
.een {order: 2;}
.twee {order: 1;}
.drie {order: 3;}
.vier {order: 5;}
.vijf {order: 4;}
```

Werken met tabellen

Een tabel

County	Establishments	Revenue per Establishment	Rank	Employees per Establishment	Rank
Marion	95	$8,236,547	1	43	3
Monroe	17	$6,273,647	2	52	1
Lake	15	$4,113,667	3	31	5
Madison	5	$3,605,800	4	49	2
Hamilton	24	$3,330,292	5	20	9
Delaware	6	$3,279,333	6	32	4
Wayne	6	$2,350,667	7	31	6
Elkhart	18	$2,300,944	8	27	7
Boone	7	$2,089,143	9	9	10
La Porte	10	$1,264,000	10	20	8
City or Town					
Indianapolis	91	$8,511,132	1	44	3
Fort Wane	13	$5,239,615	2	48	2
Lafayette	5	$5,185,200	3	51	1
Carmel	13	$3,792,231	4	17	4

*Counties and cities for which publishing industry revenue was available. Others were either not available or not disclosable

Source: IBRC, using 2002 Economic Census data

Geduld! Met CSS maak je ook deze tabel nog veel mooier

Wanneer/waarom tabellen gebruiken?

- HTML tabellen zijn gecreëerd om informatie geordend in rijen en kolommen te kunnen toevoegen aan webpagina's
- Vbn: statistieken, schema's, produktvergelijkingen,...
- Tabellen kunnen van alles bevatten: getalinformatie, maar ook teksten, afbeeldingen, ... elke andere vorm van HTML!
- Vroeger: tabellen gebruikt om layouts te maken...

Minimale tabelstructuur

De elementen waarmee we een tabel gaan vormen:

```
<table>...</table> <!-- De tabel zelf (rijen en kolommen)-->
<tr>...</tr> <!-- Een tabel-rij -->
<th>...</th> <!-- Een tabel-hoofding -->
<td>...</td> <!-- Een tabel-cel-data -->
```

Een eenvoudige tabel

- Elke tabel bestaat uit een table-element
- Daarin ga je met tr-elementen rijen definiëren
- Binnen die rijen ga je telkens de kolommen beschrijven
- Je zorgt er hierbij voor dat elke rij evenveel kolommen bevat

Table headers: <th>

- Worden anders weergegeven dan <td> elementen
- Belangrijk omdat ze informatie bevatten over de elementen die eronder worden weergegeven
- Maken een tabel "toegankelijk"
- Niet verplicht maar gebruik ze altijd!

Cellen samenvoegen/uitsmeren

- Ook *cell spanning* genoemd. Dwz dat je een cel *uitsmeert* over meerdere rijen of kolommen
- Je kunt er je tabelstructuren complexer en gesofisticeerder mee maken
- Je opmaak wordt er iets minder gemakkelijk te lezen door
- Je kan een cel uitsmeren door gebruik te maken van de colspan en rowspan attributen

Colspans

- Gebruik je in het td of th element
- Je smeert er cellen mee uit naar rechts
- colspan="2" betekent dat het element dan 2 kolommen plaats inneemt
- Jij moet er op toe zien dat het totaal aantal kolommen steeds klopt
- Je hebt in dit geval dan één kolom minder nodig voor deze rij

Rowspans

- Werken op dezelfde manier als colspans
- Maar ze smeren de cel naar beneden uit over meerdere rijen
- rowspan="3" betekent dat het element dan 3 rijen plaats inneemt
- Jij moet er op toe zien dat het totaal aantal rijen steeds klopt
- Je hebt in dit geval dan twee rijen minder nodig voor deze kolom

Tabellen en toegankelijkheid

- We voorzien hiervoor een tabelbeschrijving met <caption>
- *Captions* worden meestal boven de tabel getoond (al kan dit met CSS aangepast worden: caption-side: bottom)
- Beschrijft de informatie in de tabel en kan ook verwijzen naar de structuur van de tabel
- Wanneer aanwezig: eerste element binnen de tabel
- https://www.w3schools.com/tags/tag_caption.asp

Headers met cellen verbinden: scope

- Om aan te geven op welke data de header-beschrijving slaat
- Toegankelijkheidsexperten raden aan dat elk th element een scope attribuut heeft
- Waarden: row, column
- https://www.w3schools.com/tags/att_th_scope.asp

Content groeperen in een table: thead, tbody en tfoot

- https://www.w3schools.com/tags/tag_thead.asp

Kolommen groeperen: colgroup, col

- Je kan kolommen beschrijven met class en id met een colgroup element
- Kolomgroepen worden beschreven aan het begin van een tabel, net onder de caption
- Ze bevatten geen informatie, ze beschrijven enkel hoe de kolommen geschikt zijn binnen tabel
- https://www.w3schools.com/tags/tag_colgroup.asp

Tabellen vormgeven

- Wat de vormgeving betreft kunnen we beroep doen op onze bestaande CSS skills

```
table, th, td {
    border: .1rem solid black;
    padding: .5rem;
    text-align: center;}

th {
    background: blue;
    color: white;
    width: 12.5rem;}

tr:nth-child(even) {
    background: lightgrey;}
```

Tabellen vormgeven

- Zoals elk tekstelement op een html-pagina kunnen we de inhoud van tabellen vormgeven:
  - met de font, text en background-eigenschappen
- Om de ruimte binnen een cel aan te passen gebruik je padding op het td of th-element
- Om de width van een individuele kolom aan te passen gebruik je een class-naam of werk je met nth-child

Webtech - CSS - les9 - kristof.michiels01@ap.be