CSS osa 2 – Pseudovalitsimet ja asettelu

CS-C1180 Verkkojulkaisemisen perusteet
28.1.2020
Pekka Pulli

CSS-elementtien valinta osa 2

CSS-pseudoelementit

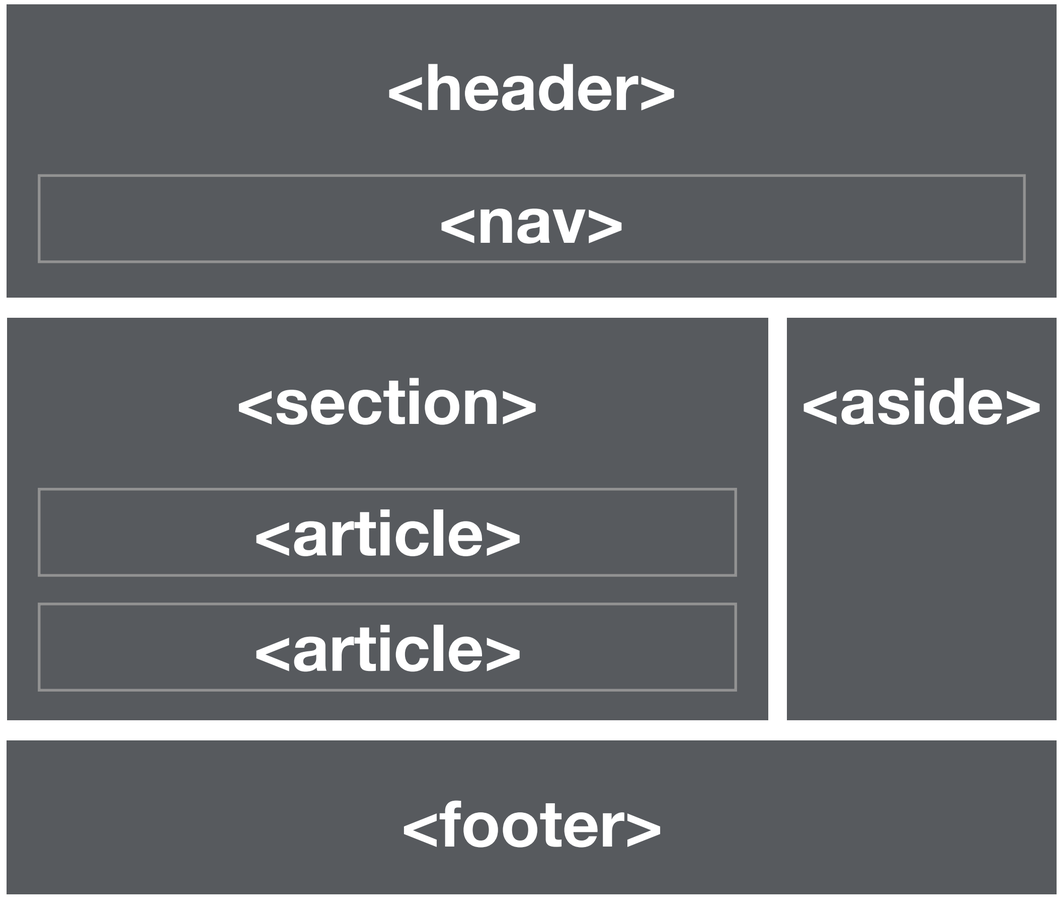
- Unohtui viime materiaalista: CSS-elementtejä voi valita myös loogisin säännöin tai tilan perusteella.
- Esimerkiksi `p:first-of-type` valitsee ensimmäisen <p>-elementin ympäröivän elementin sisältä.
- Hyödyllisimpiä pseudoluokkia:
  - `:hover` aktivoituu, kun elementin päälle tuo hiiren, esim. `button:hover`
  - `:visited` a-elementeille eli linkeille, joissa on jo käyty, esim. `a:visited`
- Lista pseudoluokista https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes
- Valintapeli TAAS http://flukeout.github.io/

CSS-asettelu

Sivun elementtien asettelu

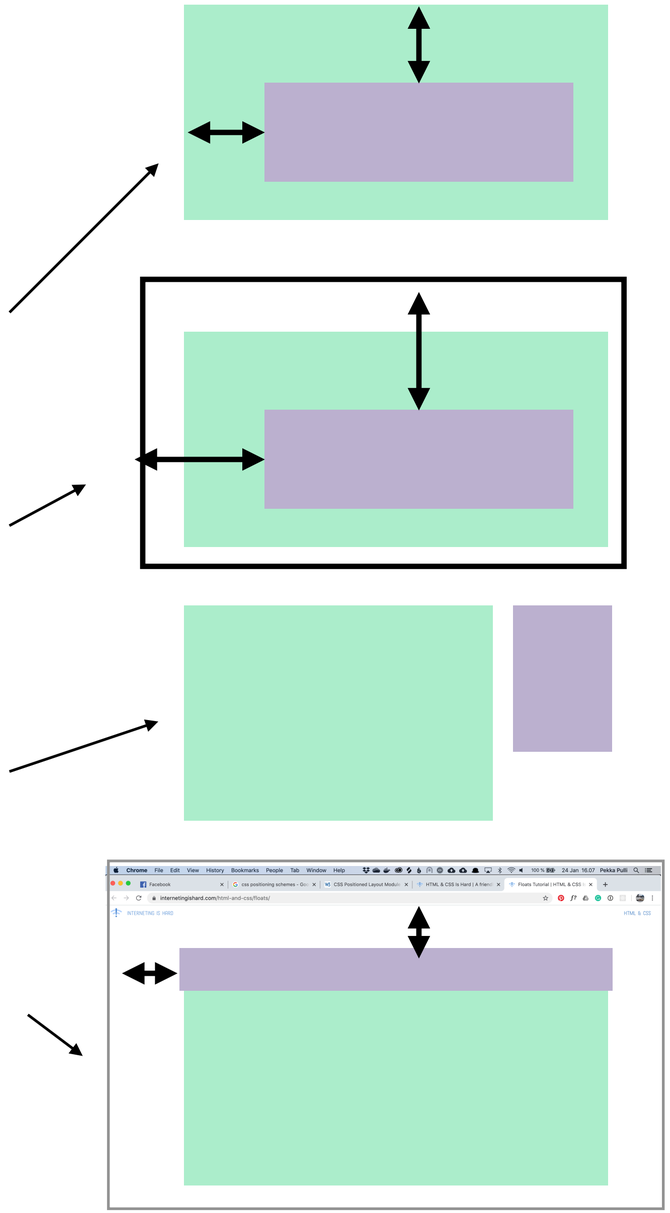
- Usein leveydet asetetaan itse (`width`, `max-width`) ja korkeus määrittyy sisällön mukaan, mutta korkeus saatetaan haluta määrittää silloin kun halutaan tehdä vaikka tasan ruudun korkuinen elementti

- Jos haluaa vain latoa elementtejä päällekäin, riittää laittaa vain elementtejä listaan. Kaikki muu tarvitsee css-muotoilua.



Asettelumoodit

- Position-attribuutti, esim. `position: absolute;`
  - `Relative`: Elementin paikka suhteutetaan ympäröivän elementin paikkaan (yleensä marginaaleilla)
  - `Absolute`: Jokaiselle elementille asetetaan paikka suhteessa joko sivuun tai ympäröivään elementtiin, jos ympäröivälle elementille on asetettu `position: relative;`
  - `Float`: Elementti “kelluu” sisällön vasemmalla tai oikealla puolella. Nykyään tämä toteutetaan useimmiten flexboxilla tai gridillä
  - `Fixed`: Elementti asetellaan suhteessa selainikkunaan, jolloin se pysyy paikallaan riippumatta skrollauksesta



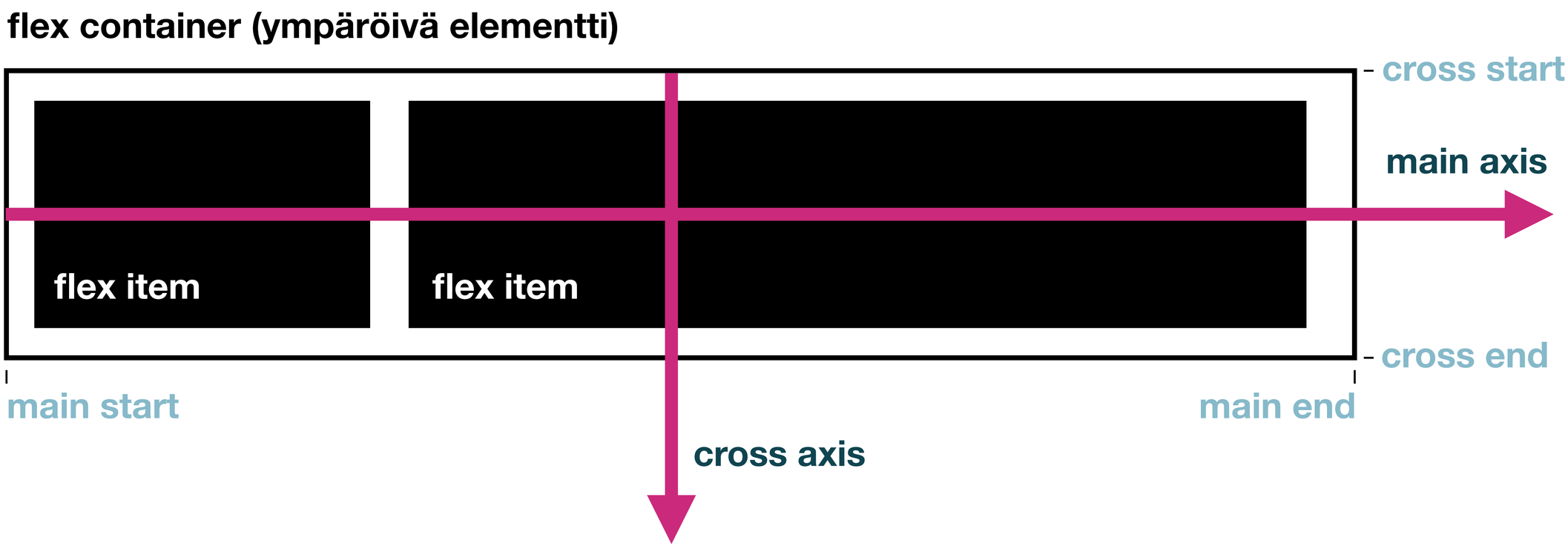
Display-moodit

- `Display`-attribuutilla voi määrittää, miten selain tulkitsee elementin asettelua. Esim. `display: flex;`. Jokaisella html-elementillä on oletusarvoinen display-arvo, esim. `<p>`:llä `display: block;`. Muutama tärkein display-attribuutin arvo:
  - `none`: elementtiä ei tuoda ollenkaan ruudulle
  - `block`: Täysleveä elementti, jolle voi asettaa leveyden ja korkeuden.
    - `flex` ja `grid`: Block-elementtejä, joilla on sisäiset asettelusäännöt (käydään läpi kohta)
  - `inline`: Näkyy rivillä muun sisällön ohessa ja vie vain niin paljon tilaa kuin tarvitsee.
  - `inline-block`: Inline-elementti, jolle voi asettaa leveyden ja korkeuden.

Flexbox

- Flexbox layout on monipuolinen asettelumenetelmä – sillä pystyy tekemään helposti skaalautuvan modernin sivuston, mutta parhaiten se soveltuu sivuston yksittäisiin komponentteihin ja sivun osien asetteluun

- Flexboxin ajatus on, että komponentille luodaan säännöt, joiden avulla se täyttää sille annetun tilan.


flex container (ympäröivä elementti)
cross start
main axis
flex item
flex item
cross end
main start
main end
cross axis

Ympäröivän elementin säännöt

- `display: flex;` asettaa elementin toteuttamaan Flexbox layoutia
- `flex-direction: row;` → tai `column;` ↓
- `justify-content`: sisällön asettelu pääsuunnassa, esim. `space-between` tai `flex-start`
- `align-items:` sisällön asettelu ristikkäisessä suunnassa, esim. `center`

Flexboxin sisällä olevan elementin säännöt

- `flex-grow: <numero>`: kuinka paljon elementti kasvaa, kun ympäröivä elementti kasvaa. 0: ei kasva, 1-N: kasvaa suhdeluvun verran (esim. 2 kasvaa kaksinkertaisesti 1:een nähden)

- `flex-shrink: 0 tai 1`: Määrää kutistuuko elementti jos ympäröivä elementti kutistuu

- `flex-basis`: Elementin oletuskoko flexbox layoutin suunnassa. Esim. `50%`, `5rem`, `auto` tai `content`.

Flexbox-resurssit

- Referenssi: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Peli! https://flexboxfroggy.com/

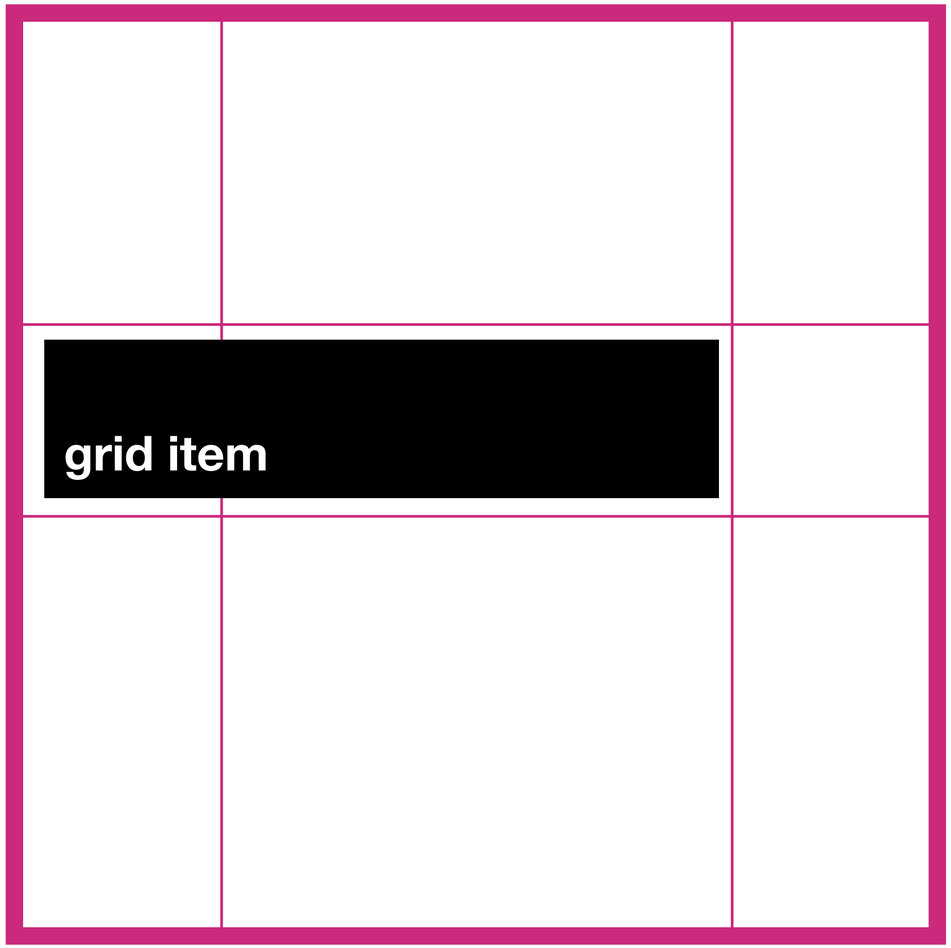
Grid

- Grid on toinen monipuolinen asettelumenetelmä, jota tuetaan hyvin moderneissa selaimissa
- Flexbox asetteli sisältöä yhdessä pääsuunnassa, grid on kaksiulotteinen järjestelmä. Siksi se sopii erityisesti kokonaisen sivun tai esimerkiksi ruudukossa olevien elementtien asetteluun.
- Grid ei toimi täysin IE 11:ssa, mutta IE 11 on väistymässä ja Grid alkaa olla hyvä vaihtoehto verkkosivujen asetteluun

Gridin käyttö: ympäröivä elementti

- `display: grid;` asettaa elementille grid layoutin
- Elementille määritellään viivasto, joka määrittää sisällä olevien elementtien paikat ja koot
- Sisällä olevien elementtien paikkoja määritellään viittaamalla viivoihin numerolla tai antamallasi viivan nimellä
- Sisältöä voi asetella myös flexbox-henkisesti attribuuteilla kuten `justify-content`, `align-content`, `justify-items` ja `align-items`

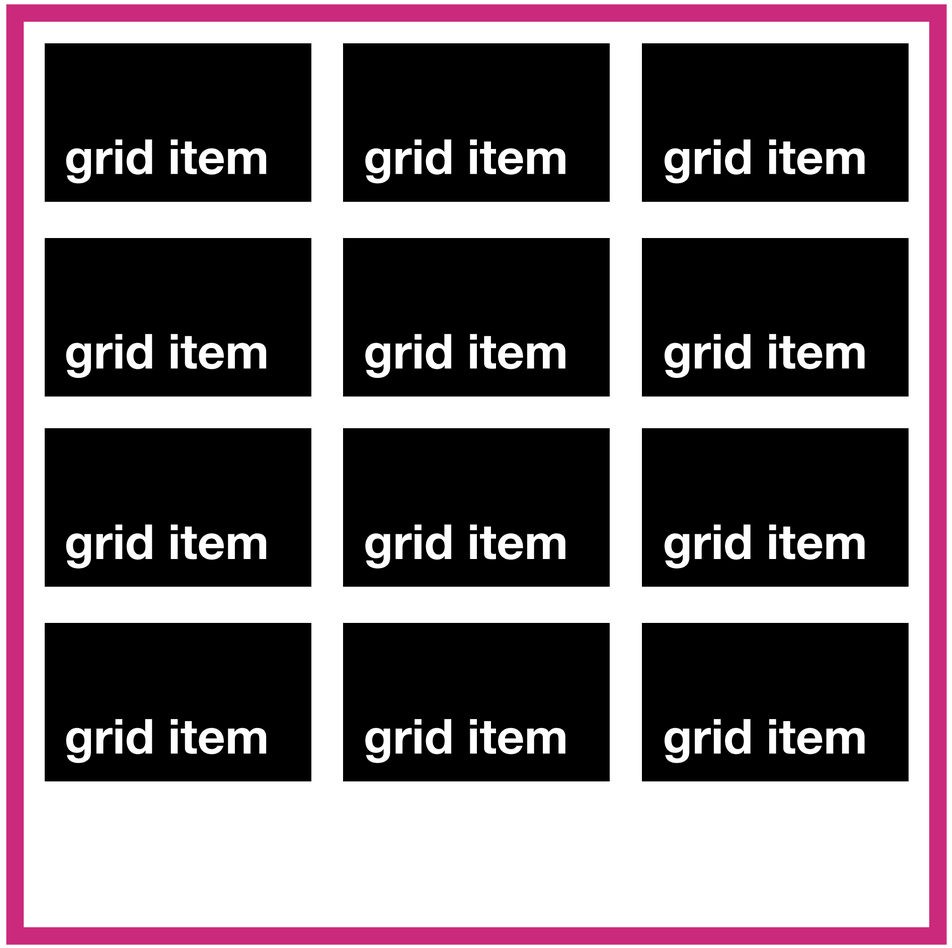
grid container (ympäröivä elementti)




```
.grid {
  display: grid;
  grid-template-columns: 25% [center] auto 25%;
  grid-template-rows: 200px 100px auto;
}
```

- `grid-template-columns` ja `grid-template-rows` määrittävät gridin asettelun raamiviivat.
- Arvot asetetaan järjestyksessä vasemmalta oikealle ja ylhäältä alas
- viivan voi nimetä hakasulkeilla, esim. yllä `[center]`

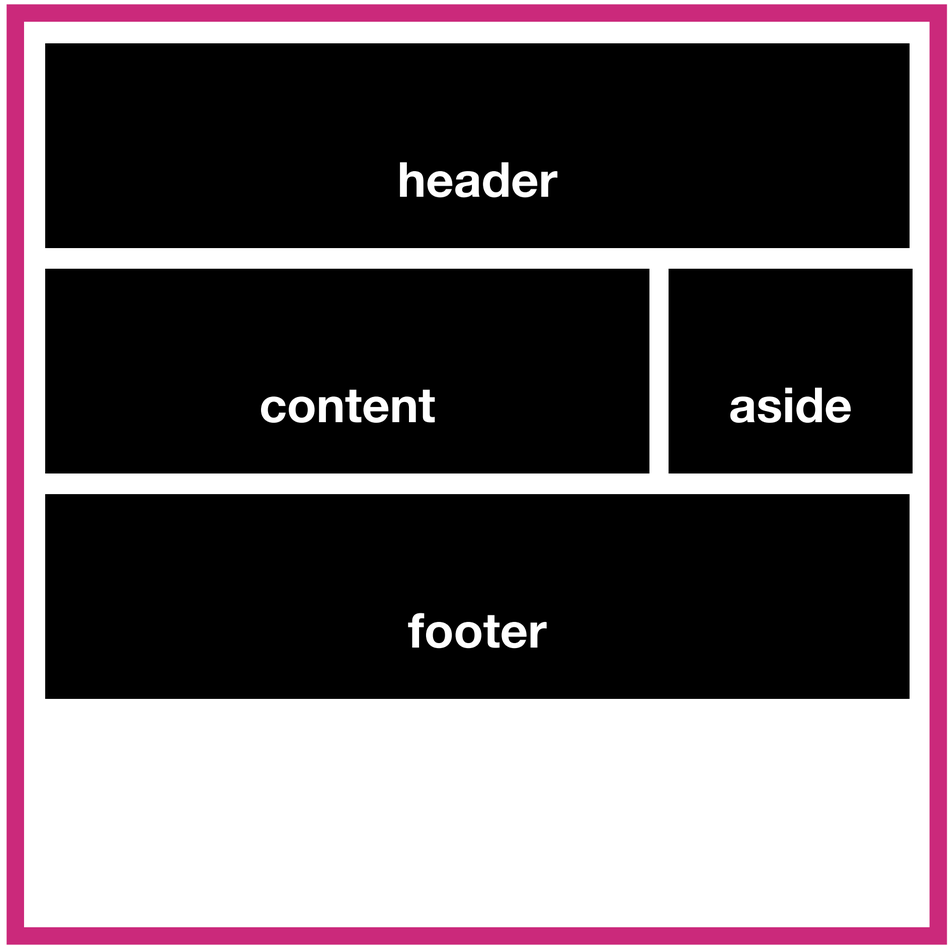
grid container (ympäröivä elementti)




```
.grid {
  display: grid;
  grid-template-columns:
    repeat(auto-fill, minmax(240px, 1fr));
  grid-gap: 30px;
}
```

- `repeat`-parametrilla voi tehdä toistuvia elementtejä ja täyttää ruudukon helposti
- `grid-gap` määrittää elementtien välit. Välin voi asettaa molemmissa suunnissa, esim. `grid-gap: 10px 15px;`
- `1fr` tarkoittaa yhtä osuutta (fraction) jäljelle jäävästä tilasta

grid container (ympäröivä elementti)




```
.grid {
  display: grid;
  grid-template-columns: 70% 30%;
  grid-gap: 30px;
  grid-template-areas:
      "header header"
      "content aside"
      "footer footer";
}
```

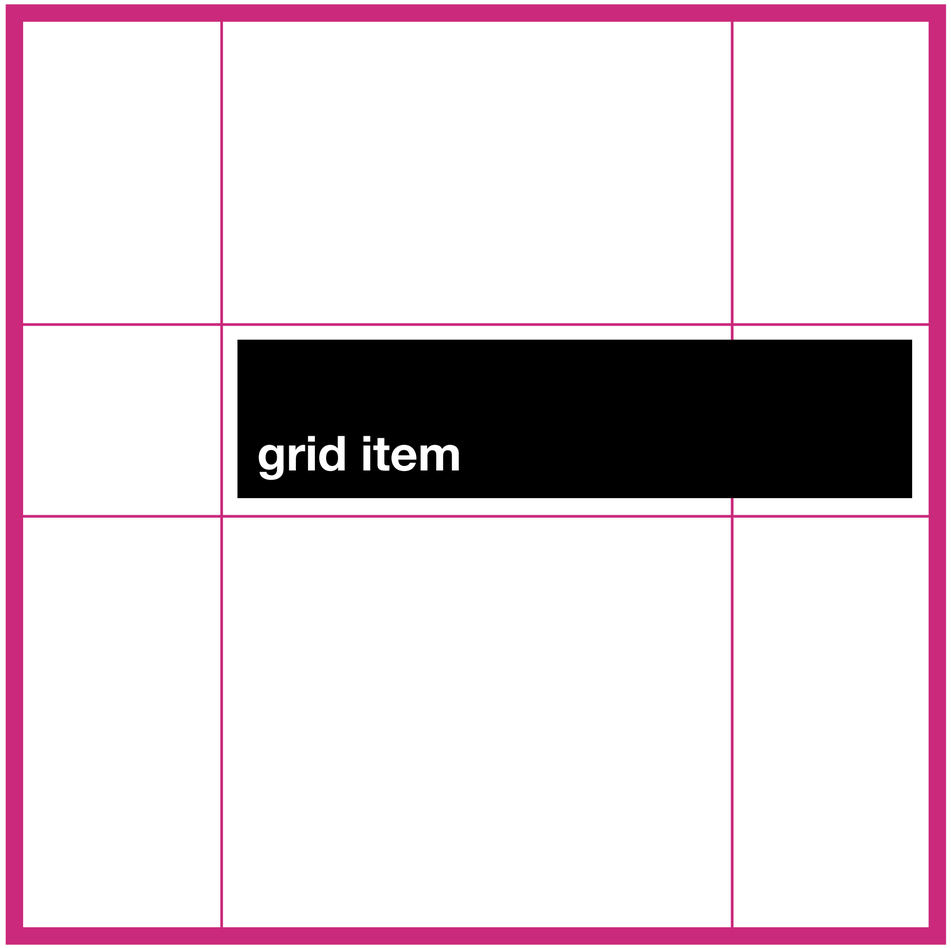
- Gridille voi määritellä myös alueita, joille sisältö on helppo asetella.
- Paikka gridissä määritellään sisältöelementissä

Gridin käyttö: sisältöelementti

- Sisältöelementin paikan voi asetella kolmella eri tavalla
- Helpoin on ilman mitään määrittelyä. Silloin sisältöelementit asettuvat järjestyksessä vapaisiin soluihin
- `grid-column-start, grid-column-end, grid-row-start, grid-row-end`: Sisältöelementin alku- ja loppuviivat
  - Voi määrittää numerona [1-n] tai gridiä määritellessäsi antamallasi nimellä, esim. `[center]`, tai elementille voi antaa useamman solun `span`-avainsanalla. Esim.

```
{
  grid-column-start: 1;
  grid-column-end: span 2;
}
```

grid container (ympäröivä elementti)



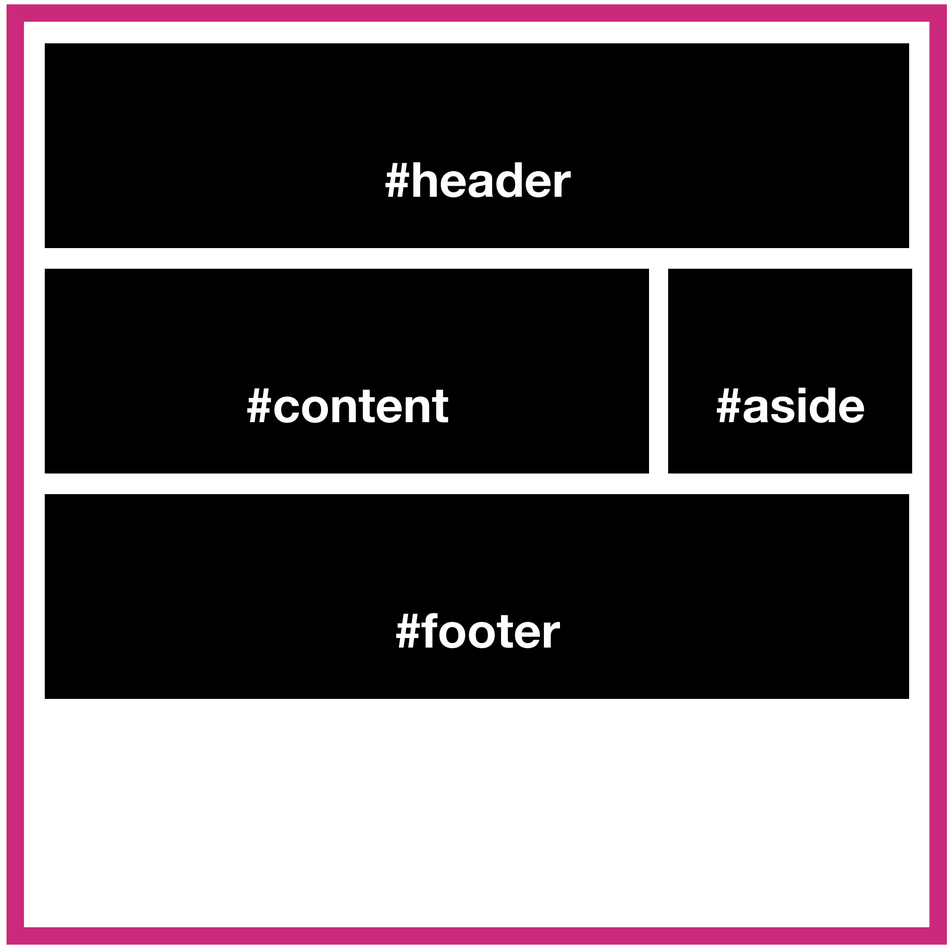

```
.grid {
  display: grid;
  grid-template-columns: 25% [center] auto 25%;
  grid-template-rows: 200px 100px auto;
}

.grid-item {
  grid-column-start: center;
  grid-column-end: span 2;
  grid-row-start: 2;
}
```

Gridin käyttö: sisältöelementti

- Toinen tapa antaa sisältöelementille paikka käyttää aiemmin mainittua `grid-template-areas` -sääntöä.

- Paikka määritellään antamalla sisältöelementille `grid-area`.

grid container (ympäröivä elementti)




```
<body>
  <header id=“header”/>
  <section id=“content”/>
  <aside id=“aside”/>
  <footer id=“footer”/>
</body>
```

```
.grid {
  [...]
  grid-template-areas:
      "header header"
      "content aside"
      "footer footer";
}

#header {
  grid-area: header;
}

#content {
  grid-area: content;
}

#aside {
  grid-area: aside;
}

#footer {
  grid-area: footer;
}
```

Grid-resurssit

- Peli! https://cssgridgarden.com/
- https://css-tricks.com/snippets/css/complete-guide-grid/

Responsiivisuus

Responsiivisuus

- Nykypäivänä verkkosivuston tulee skaalautua saumattomasti eri päätelaitteille
- Tämän voi toteuttaa käyttämällä asemoinnissa suhteellisia kokoja ja toteuttamalla sivustolle sääntöjä sivun leveyden mukaan murtumispisteiden mukaan (breakpoint) ja media queryjen avulla
- Tyypillisiä muuttuvia attribuutteja breakpointeissa: marginaalit, flexboxin orientaatio, flexboxin justify ja align, fonttikoko

Media queryt

```
.box {
  width: 100%;
  max-width: 640px;
  margin: 0 auto;
  background-color: pink;
}

@media (max-width: 680px) {
  .box {
    margin: 0 20px;
    background-color: blue;
  }
}
```

- Referenssi: https://www.w3schools.com/css/css3_mediaqueries_ex.asp

Ensi kerralla (4.2. ja 6.2.)

- Portfoliosivuston tekoa assareiden opastuksella, dl 24.2.

Kurssin jatko

- 25.2. Javascript-luento (Juha-Matti Santala, Futurice)
- 10.3. Videotuotantoluento (Anna Berg, Media Factory)
- 24.3. Javascript-ekosysteemi ja React (Matias Saarinen, Reaktor)