CSS Grid

Introducere

Modulul CSS Grid Layout oferă un sistem de layout bazat pe grid (se pot numi în limba română rețea, pentru că este construita pe două direcții, "două dimensiuni") cu rânduri și coloane, ceea ce face mai ușor să proiectezi pagini web fără a fi nevoie să utilizați float și position.

În trecut, setul de instrumente pentru developer-i arăta astfel:

- **Floats** - Un pic cam nebun, imprevizibil, greu de gestionat ca o grilă bidimensională.
- **Tables** - Este prea rigid, nu e "responsive", nu e frumos, dar a rezolvat câteva momente dificile și în continuare ajută dacă depuneți efortul necesar și aveți timp ☺.
- **Positioning** - De asemenea, prea specifică și rigidă, care nu se adaptează la structurile fluide.

Acestea sunt bune și funcționează, dar în aspecte mai complexe sau mai "responsive" pot fi limitate și frustrante. Naiba, ele pot fi frustrante în aspecte simple, de asemenea.

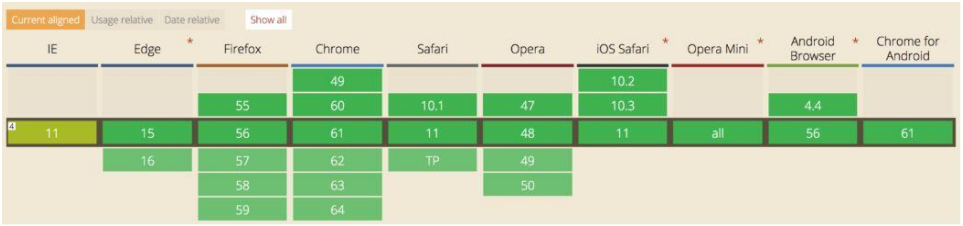
Flexbox și CSS Grid

Ambele sunt instrumente relativ noi care au fost introduse în specificații CSS pentru a aborda mai bine problemele complexe de aspect.

Acestea deblochează noi posibilități pentru problemele vechi, permit lucruri care nu au fost posibile înainte și rezolvă problemele reale cu care ne confruntăm la dezvoltare rapidă a web-ului.

Compatibilitate | Flexbox = Flexible + Box

Lansat: versiunea veche 2009, versiunea curentă 2012.



Compatibilitate | Grid layout

Lansat: martie 2017, lansat pe Edge Oct 2017.

*FAQ: Flexbox vs. Grid layout?

Flexbox: o singură dimensiune

Când vă gândiți la Flexbox trebuie să vă gândiți la scenarii atunci când construiți un aspect unidimensional. De exemplu, dacă aveți nevoie de un rând cu câteva coloane în el. Sigur cu Flexbox puteți crea o grămadă de rânduri cu coloane, dar aceste coloane nu au nicio cunoaștere sau relație una cu cealaltă. Ceea ce vreau sa spun este ca Flexbox excelează la "micro layout": alinierea, comanda, ambalarea și distribuirea spațiului între elementele dintr-un singur container.

CSS Grid: două dimensiuni

Excelențe la împărțirea unei pagini în regiuni majore sau la definirea relației în ceea ce privește dimensiunea, poziția și stratul dintre elementele HTML.

Flexbox vs. Grid...

Amplasarea elementelor, spațiul alb, configurațiile complexe ale grid-ului, menținând în același timp structuri performante al layout-ului lichid ... toate făcute cu ușurință cu CSS Grid.

Deci, ce știm până acum credeți că CSS Grid-ul înlocuiește Flexbox-ul?

→ NU ... dar sunt câteva diferențe importante:

**Flexbox este în esență pentru a stabili elemente într-o singură dimensiune - într-un rând *SAU* o coloană.

**CSS Grid se referă la structura elementelor în două dimensiuni - rânduri *ȘI* coloane care trebuie să se raporteze una la cealaltă.

...ele pot fi folosite chiar și în tandem!

CSS Grid Layout exemplu practic

Mai sus am ajuns la concluzia că se poate și chiar se recomandă, folosirea Grid-ului împreună cu Flexbox. În exemplul următor vom discuta cum, unde și de ce alegem sau trebuie să utilizăm așa:

Sursă cod pentru cursul nr.6: https://github.com/AtiSUN/Modul-II-Grid

CSS Grid și Flexbox folosit împreună: https://codepen.io/Atisun/pen/VqQxLG

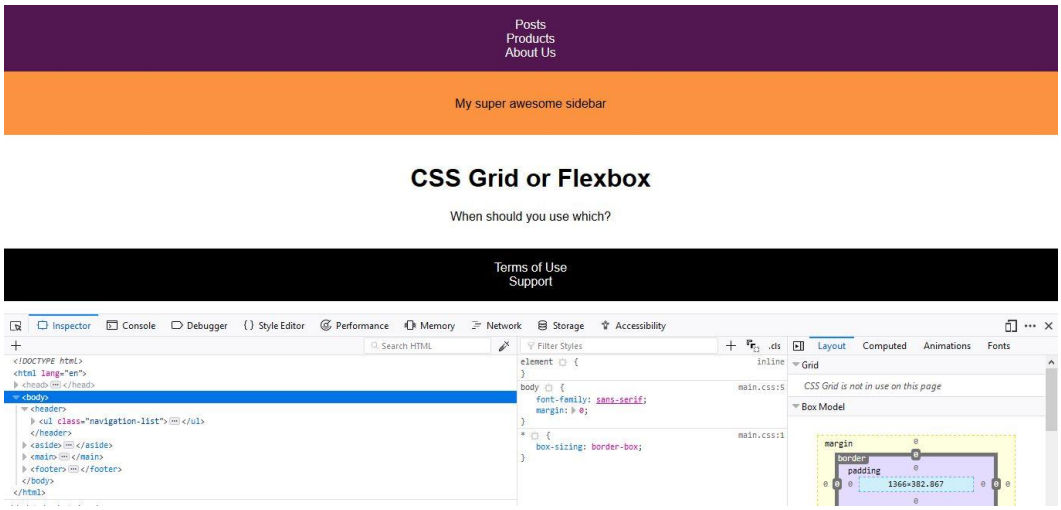
```
<header>
  <ul class="navigation-list">
    <li class="navigation-item"><a href="#">Posts</a></li>
    <li class="navigation-item"><a href="#">Products</a></li>
    <li class="navigation-item"><a href="#">About Us</a></li>
  </ul>
</header>

<aside>
  <p>My super awesome sidebar</p>
</aside>

<main>
  <h1>CSS Grid or Flexbox</h1>
  <p>When should you use which?</p>
</main>

<footer>
  <ul class="navigation-list">
    <li class="navigation-item"><a href="#">Terms of Use</a></li>
    <li class="navigation-item"><a href="#">Support</a></li>
  </ul>
</footer>
```

Atașat găsiți mai jos un print-screen cu patru secțiuni (1. mov = header; 2. portocaliu = sidebar “aside”; 3. alb = main; 4. negru = footer):



Elementele de mai sus sunt diviziuni/secțiuni, de aceea ocupă 100% din lățimea ecranului și datorită block flow-ului se așează automat unul sub celălalt. Cu ajutorul CSS GRID putem să le transformăm într-o rețea alcătuită din coloane și rânduri pentru a manipula elementele mai ușor și în aceeași timp să rămână flexibile.

Acest lucru reușim să îl obținem cu proprietatea `display: grid;`, atât de simplu am transformat `<body>` în grid layout.

1. Grid-Template

După stabilirea proprietății `display: grid;` următorul pas este să alocăm numărul de rânduri și coloane.

Folosind proprietățile `grid-template-columns:` și `grid-template-rows:` stabilim exact layout-ul necesar al proiectului actual.

Dorim ca sidebar-ul să fie lângă conținutul principal, pentru asta trebuie să împărțim grid-ul nostru în două coloane, asta vom face cu proprietatea `grid-template-columns: 300px 1fr;` unde fr este o fracțiune, adică spațiu rămas împărțit egal între fracțiuni. În cazul nostru avem o singură fracțiune asta înseamnă că din lățimea 100% scădem 300 de pixel și ce rămâne este ocupat de coloana numărul doi. Putem să combinăm px | fr | % | em | rem | pt.

Acum urmează să împărțim și rândurile din grid: `grid-template-rows: 4rem auto 4rem;` stiind că rem este rootem și este egal cu 16px ***4rem*16px=64px***. Cu valoriile de mai sus am stabilit trei rânduri, mai exact primul rând având înălțimea 4rem=64px, al doilea auto (adică umplă spațiu disponibil) și al treilea (footer-ul) la fel ca primul (header | meniu) 4rem.

2. Denumirea liniilor despărțitoare

Liniile despărțitoare sunt stabilite de numărul coloanelor și a rândurilor. Dacă deschidem proiectul în Mozilla Firefox și inspectăm orice element, imediat observăm o diferență destul de mare. Mozilla pe lângă suportul cel mai mare pana în ziua de azi pentru grid, mai are un extra ajutor in developer tool.

Mergând mai departe, vreau să vă arăt cum este reprezentată acest grid direct pe proiect:

Deocamdată această opțiune găsiți doar în Mozilla Firefox.

Revenind la denumirea despărțitoarelor, nu suntem obligați să folosim doar numerele pe care sunt afișate pentru coloane și rânduri.

Folosind sintaxa de împărțirea coloanelor vă arăt cum putem să denumim/redenumim despărțitoarele:

```
grid-template-columns: [col-1-start] 300px [col-1-end col-2-start] 1fr [col-2-end];
```

3. Poziționarea elementelor

După stabilirea numărul coloanelor și rândurilor putem să stabilim pentru fiecare element HTML cât spațiu să ocupe din grid-ul stabilit.

În cazul nostru vrem ca header și footer să ocupe câte două coloane, adică de la linia 1 pana la linia 3.

Cu proprietatea `grid-column-start: col-1-start;` stabilim de unde să înceapă elementul și cu proprietatea `grid-column-end: col-2-end;` specifică până unde să întinde. Folosim selectorul de clase pentru header și footer exact cum puteți vedea mai jos:

```
header {
	background: #521751;
	grid-column-start: col-1-start;
	grid-column-end: col-2-end;
}

footer {
	background: black;
	grid-column-start: col-1-start;
	grid-column-end: col-2-end;
}
```

4. Proprietăți de Grid Conteiner

Proprietatea `justify-content` este utilizată pentru alinierea întregii ***grid*** în interiorul containerului.

```
justify-content: space-evenly;
justify-content: space-around;
justify-content: space-between;
justify-content: center;
justify-content: start;
justify-content: end;
```

Proprietatea `align-content` este utilizată pentru alinierea verticală a grilei întregi în interiorul containerului.

```
align-content: center;
align-content: space-evenly;
align-content: space-around;
align-content: space-between;
align-content: start;
align-content: end;
```

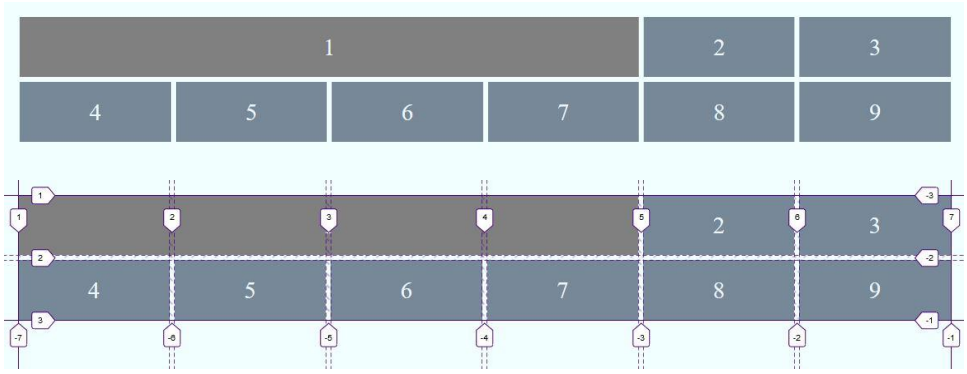
5. Proprietăți de Grid Item

Children elements (Items)

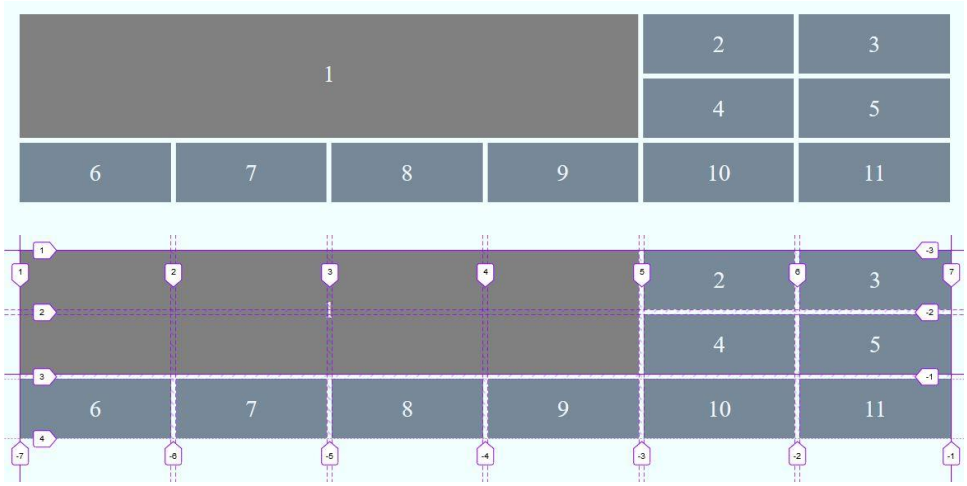
Un grid container conține elemente de grid (grid items).

Implicit, un container are un element *grid* pentru fiecare coloană, în fiecare rând, dar puteți modela elementele *grid* astfel încât acestea să cuprindă mai multe coloane și / sau rânduri.

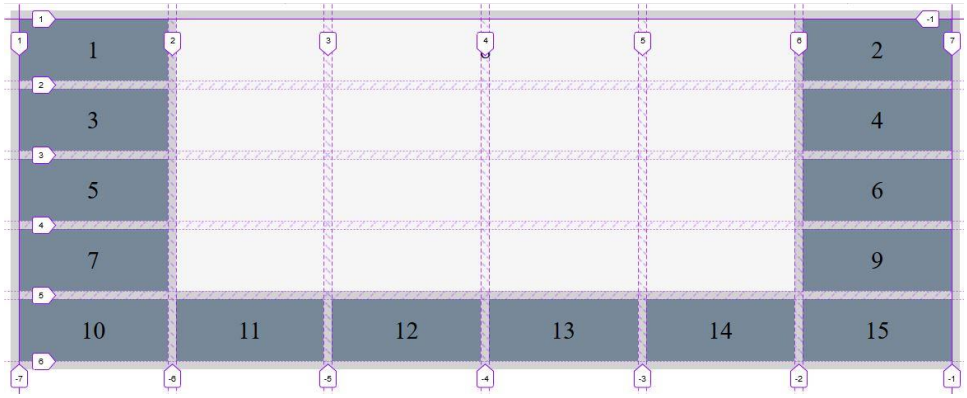
Proprietatea `grid-column` definește coloanele în care se plasează un element. Definiți locul în care elementul va începe și unde se va termina elementul.



Proprietatea `grid-row`: definește rândul în care se plasează un element. Definiți locul în care elementul va începe și unde se va termina elementul.

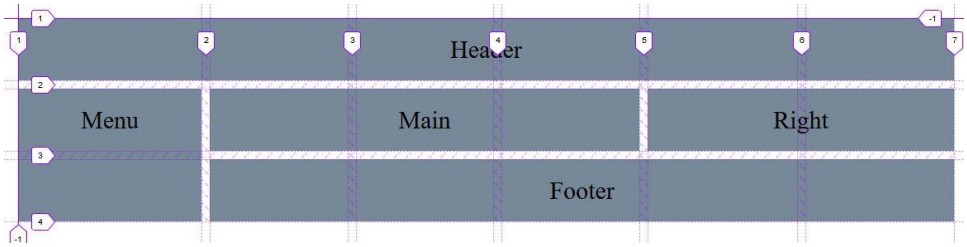


Proprietatea `grid-area`: poate fi folosită ca proprietate sintagmă pentru grid-start-start, grid-column-start, grid-end-end și grid-column-end proprietăți.



Naming Grid Items

Proprietatea `grid-area`: poate fi, de asemenea, utilizată pentru a atribui nume elementelor *grid*.



Exemplu CSS Grid cu Flexbox

Exemplu pentru Flexbox și CSS Grid folosit în tandem: https://codepen.io/Atisun/pen/PXQqWm

grid-explained.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
       <meta charset="utf-8">
       <title>CSS Grid & Flexbox</title>
       <link rel="stylesheet" href="css-for-grid-explained.css">
  </head>
  <body>
       <main class="deals">
             <article class="sale-item">
                 <h1>Computer Starter Kit</h1>
                 <p>This is the best computer money can buy, if you don’t have much
            money.</p>
                 <ul>
                    <li>Computer</li>
                    <li>Monitor</li>
                    <li>Keyboard</li>
                    <li>Mouse</li>
                 </ul>
              <img src="imgs/computer.jpg" alt="You get: a white computer with
            matching peripherals">
                 <button>BUY NOW</button>
             </article>
             <article class="sale-item"></article>
             <article class="sale-item"></article>
             <article class="sale-item"></article>
             <article class="sale-item"></article>
       </main>
    </body>
</html>
```

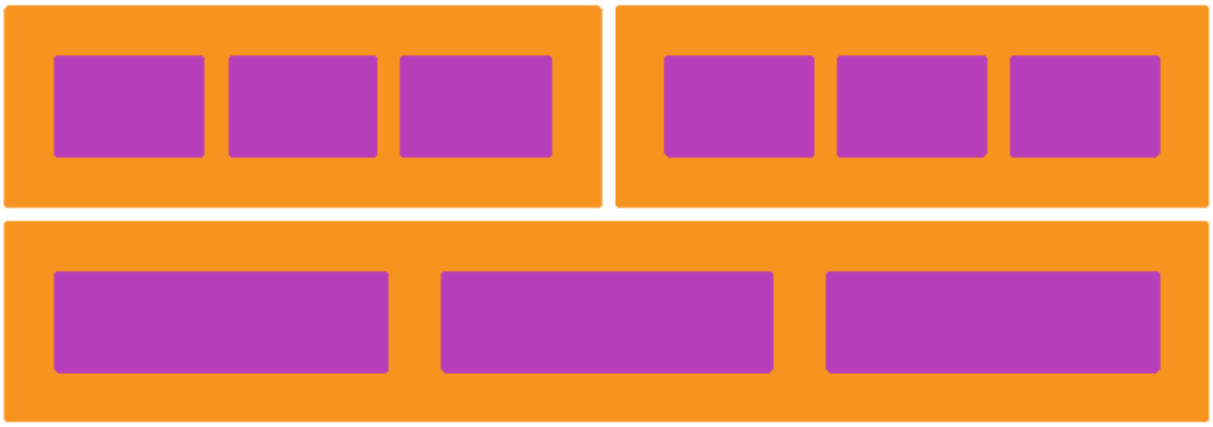
css-for-grid-explained.css

```
* {
  box-sizing: border-box;
}
body {
  padding: 1rem;
}

/* Just for the fallback layout */
main {
  max-width: 500px;
  margin: 0 auto;
}
article {
  margin: 1rem 0;
}

/* Now lets do a Grid-based layout */
@supports (display: grid) {
  main {
       max-width: 10000px;
       margin: 0;
  }
  article {
       margin: 0;
  }
  .deals {
       display: grid;
       grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
       grid-gap: 1rem;
  }
}
.deals {
    font-family: Avenir, Roboto, Helvetica, san-serif;
    font-size: 80%;
}
.sale-item {
    display: flex;
    flex-flow: column;
    border: 1px solid black;
    border-radius: 1rem;
    padding: 2rem;
}
.sale-item > h1 {
    margin: 1rem 1rem 0;
}
.sale-item > ul {
    margin: 0 0 1rem;
}
.sale-item > p {
    margin: 0.25em 1rem 1rem;
}
.sale-item > img {
    order: -1;
    align-self: center;
}
```

Exemplu



Elementele portocalii sunt elemente de grid items și flex parents

Elementele mov sunt flex items

Elementele portocalii sunt elemente de flex items și grid parents

Elementele mov sunt grid items

Interoperabilitate

"Nu este vina ta, dar este problema ta"

CSS Grid ar trebui să fie utilizat în locul potrivit, la momentul potrivit. Trebuie să faceți compromisuri:

- Prezentați un aspect simplificat pentru browser-ele mai vechi.
- Utilizați-l în zone cu impact ridicat.

Un alt exemplu de CSS Grid Layout.

Diviziunii cu clasa grid-container a fost transformat în

Datorită proprietății "`grid-template-columns:`" cu valori "`auto auto auto;`" seamănă foarte mult cu un tabel:

HTML

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

CSS

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #862486;
  padding: 10px;
}
.grid-item {
  background-color: orange;
  border: 1px solid rgba(134, 36, 134, 0.8);
  color: black;
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

1	2	3
4	5	6
7	8	9