CSS Grid - Въведение

Какво е и кога се появява CSS?

CSS или иначе казано – Cascading Style Sheets е език използван за стилизиране на HTML файлове. CSS "казва" на браузъра как да се заредят елементите в страницата.

Появата на CSS се дължи на необходимостта от стилизиране и персонализиране на уеб страниците. С появата си CSS позволява промяна на всеки един HTML елемент. Идеята за него се заражда през 1994г. и получава първата си версия две години по-късно през 1996г.

Браузърите обаче не внедряват бързо тази нова технология, а чак през 2002г. се появява първия браузър с внедрена пълна CSS спецификация.

Структура

CSS файловете се състоят от така наречени "правила", делящи се на 2 типа – селектори и декларационния блок. Всяко CSS правило е препоръчително да завършва с ';'. Макар и използването им да не е напълно задължително, тяхната липса е възможно да доведе до проблеми в стилизацията на елементите.

Какво е селектора?
Селекторът е низ, който идентифицира един или повече елементи на страницата, следвайки специален синтаксис.

Какво е декларационен блок?
Декларационния блок от друга страна съдържа една или повече декларации, на свой ред съставени от двойка свойство и стойност.

Как изглежда CSS?

Както казахме по-рано правилата в CSS са два типа селектор и декларация. Декларацията е честта съдържаща различни правила, всяко от които е изградено от свойство и стойност.

```
p {
  font-size: 20px;
}
```

Тук p е селектора и прилага правилото, че на дадения елемент задава стойността 20px на свойството font-size.

Даден селектор може да се прилага на повече от един елемент:

```
h1, h2, h3 {
  color: blue;
  font-family: Arial;
}
```

Селекторите могат да се прилагат освен на HTML тагове като елементи, така и на елементи, които имат като атрибут class или id.

Относно форматирането на тези правила няма специфични изисквания. Те могат да бъдат на един ред, на много редове, да съдържат много табове и тн.

```
/* Едно и също правило, форматирано по различни начини */
p { color: red; font-size: 20px; }

p {
  color: red;
  font-size: 20px;
}

p {
  color: red;
  font-size: 20px;
}
```

Какво е CSS Grid?

Стига толкова обаче за CSS идва реда да си говорим в конкретика за CSS Grid. CSS Grid Layout или иначе казано "решетка" е двуизмерна система за оформление, базирана на мрежа, която в сравнение с всяка система за уеб оформление от миналото напълно променя начина, по който проектираме потребителски интерфейси.

Преди появата на CSS Grid подреждането на елементите върху страниците се осъществяваше чрез таблици и след това, чрез така наречените floats позиционирания и inline-block, но тези хакове не са пълноценно решение на проблемите свързани с всички видове позиционирания. Много готин метод на подравняване е Flexbox, но не е напълно ефективен, защото предоставя само еднопосочно позициониране.

Точно заради нуждата от инстумент за двуизмерно позициониране на елементите се появява и CSS Grid. Въпреки че идеята за подравняване на елементи към колони е приложена на някои страници още през 2011 г., този стандарт се поддържа едва от около 2017 г. .[6]

Малко CSS Grid термини

Решетъчен контейнер – grid container
Елементът, върху който display: grid се прилага. Това е пряк родител на всички елементи на мрежата.

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 100px;
}
```

Решетъчен елемент – grid item
Децата (т.е. преките наследници) на контейнера на мрежата.

```
<div class="container">
  <div class="item"> </div>
  <div class="item">
    <p class="sub-item"> </p>
  </div>
  <div class="item"> </div>
</div>
```

Grid line
Разделителните линии, които изграждат структурата на решетката. Те могат да бъдат или вертикални („линии на мрежата на колони“), или хоризонтални („линии на мрежата на редове“) и да се намират от двете страни на ред или колона.

Grid Cell
Пространството между два съседни реда и две съседни колонни линии на мрежата.

Grid Track
Пространството между две съседни линии на мрежата. Можете да мислите за тях като за колоните или редовете на мрежата.

Grid Area
Областта на мрежата може да бъде съставена от произволен брой клетки на мрежата.

История и еволюция на Grid системите

Какво прави display?

Свойството display на даден обект определя как той се изобразява от браузъра. Възможните стойности на display включват: block, inline, none, contents, flow, flow-root, table, flex, grid, list-item, inline-block, inline-table, inline-flex, inline-grid, inline-list-item

Какво правят основните стойности?

Inline

Елементът, върху който display: grid се прилага. Това е пряк родител на всички елементи на мрежата. В този пример container е мрежовият контейнер.

inline-block
Подобно на inline, но с inline-block width и height се прилага, както посочите.

block
Както споменахме, обикновено елементите се показват в един ред, с изключение на някои елементи, включително: div, p, section, ul, които се задават като block от браузъра. С display: block елементите се подреждат един след друг, вертикално и всеки елемент заема 100% от страницата.

None
Използването display: none кара елемент да изчезне. Все още е там в HTML, но просто не се вижда в браузъра.

Предшествениците на Grid - floating и cleaning

Float - Плаването е CSS свойство, което кара някой елемент да "плава" по самата уеб страница. Този тип хак се използва за подходящото позициониране на елементите по уебстраницата, но на моменти този float може да бъде доста досаден (говоря от личен опит). Възможните стойности на float са left, right, none. Примерна употреба е, ако имаме 3 картинки, които искаме да подравним отдясно, то ще използваме float: right. По този начин тези картинки ще се подредят една до друга отдясно, докато не се запълни цялата ширина на реда и тогава елементите слизат на долния ред. Ако искаме картинките да бъдат подравнени в ляво, но вместо хоризонтално да бъдат подредени вертикално, то ще добавим clear: left. По същия начин възможните стойности за clear са left, right, both, none.

Появата на Grid и Flexbox

С появата на Flexbox и CSS Grid всички проблеми от типа създадени от досадната употреба на clear и float се изчистват. Нека си поговорим малко за това какво представлява Flexbox. В сравнение с CSS Grid (който е двуизмерен), flexbox е едноизмерен модел на оформление. Той ще контролира оформлението въз основа на ред или колона, но не заедно едновременно. Основната цел на flexbox е да позволи на елементите да запълнят цялото пространство, предлагано от техния контейнер, в зависимост от някои правила, които задавате.

Основи на CSS Grid

Какво представлява CSS Grid Layout?

CSS Grid е фундаментално нов подход за изграждане на оформления с помощта на CSS. Към април 2019 г., всички основни браузъри (с изключение на IE, който никога

няма да има поддръжка за него) вече поддържат тази технология, покривайки 92% от всички потребители.

CSS Grid vs. Flexbox

CSS Grid не е конкурент на Flexbox, защото те работят по различен начин. CSS Grid работи с 2 измерения (редове И колони), докато Flexbox работи с едно измерение (редове ИЛИ колони).

Създаване в детайли

Оформлението на CSS Grid се активира върху контейнерен елемент (който може да бъде div или всеки друг HTML елемент), чрез настройка display: grid.

Основни CSS Grid свойства

Свойства за контейнера
display: grid - дефинира grid контейнер
grid-template-columns - дефинира колоните
grid-template-rows - дефинира редовете
grid-template-areas - дефинира областите
grid-gap - разстояние между елементите
justify-items - хоризонтално подравняване
align-items - вертикално подравняване

Свойства за елементите
grid-column - позиция на колона
grid-row - позиция на ред
grid-area - име на област
justify-self - хоризонтално подравняване
align-self - вертикално подравняване

Специални единици и функции
fr - фракционна единица
repeat() - повтаряне на стойности
minmax() - минимална и максимална стойност
auto-fill - автоматично запълване
auto-fit - автоматично напасване

Подравняване и разпределение
justify-content - хоризонтално разпределение
align-content - вертикално разпределение
place-content - комбинация от двете
place-items - подравняване на всички елементи

Най-основните свойства на контейнера са grid-template-columns и grid-template-rows.

grid-template-columns и grid-template-rows

Тези свойства определят броя на колоните и редовете в grid-а и също така задават ширината на всяка колона/ред.

```
.container {
        display: grid;
        /* 4 колони */
        grid-template-columns: 1fr 1fr 1fr 1fr;
        /* 2 реда */
        grid-template-rows: 100px 100px;
        /* разстояние между елементите */
        gap: 10px;
      }
```

Автоматични размери

Много пъти може да имате фиксиран размер на горния колонтитул, фиксиран размер на долния колонтитул и основното съдържание, което е гъвкаво по височина, в зависимост от дължината му. В този случай можете да използвате ключовата дума auto:

```
.container {
        display: grid;
        /* фиксиран размер за header и footer */
        grid-template-rows: 100px auto 100px;
        /* две колони с различно съотношение */
        grid-template-columns: 1fr 2fr;
      }
```

Различни размери на колони и редове

```
.container {
  display: grid;
  /* фиксирани и гъвкави колони */
  grid-template-columns: 100px 200px 1fr;
  /* различни височини на редовете */
  grid-template-rows: 50px 150px;
  /* разстояние между елементите */
  gap: 15px;
}
```

Grid в действие - практически примери

След като видяхме интерактивния пример и примерни имплементации на различни типове Grid-ове е време да надникнем към 8 модерни CSS Layout-и

01. Супер центриран place-items: center

HTML

```
<div class="parent" >
  <div class="child" contenteditable>
    :)
  </div>
</div>
```

CSS

```
.parent {
  display: grid;
  place-items: center;

  background: lightblue;
  width: 500px;
  height: 500px;

  resize: both;
  overflow: auto;
}

.child {
  // etc.
  padding: 0.5rem;
  border-radius: 10px;
  border: 1px solid red;
  background: lightpink;
  font-size: 2rem;
  text-align: center;
}
```

02. Странична лента grid-template-columns: minmax(<min>, <max>) ...

HTML

```
<div class="sidebar" contenteditable>
  Min: 150px
  <br/>
  Max: 25%
</div>
<p class="content" contenteditable>
  Lorem ipsum dolor sit amet consectetur adipi</p>
```

CSS

```
body {
  display: grid;
  grid-template-columns: minmax(150px, 25%) 1fr;
  padding: 0;
  margin: 0;
}

.sidebar {
  height: 100vh;

  //   etc.
  background: lightpink;
  font-size: 2rem;
  text-align: center;
}

.content {
  padding: 2rem;
}

body {
  font-family: system-ui, serif;
}
```

03. Палачинков стек grid-template-rows: auto 1fr auto

HTML

```
<header><h1>Header.com</h1></header>
<main></main>
<footer>Footer Content — Header.com 2020</footer>
```

CSS

```
body {
  display: grid;
  height: 100vh;
  grid-template-rows: auto 1fr auto;
}

// etc

header {
  background: lightpink;
  padding: 2rem;
```

```
}

main {
  background: coral;
}

footer {
  background: wheat;
  padding: 2rem;
  text-align: center;
}

body {
  font-family: system-ui, sans-serif;
}
```

04. Класически Свети Граал grid-template: auto 1fr auto / auto 1fr auto

HTML

```
<header><h1 contenteditable>Header.com</h1></header>
<div class="left-sidebar" contenteditable>Left Sidebar</div>
<main contenteditable></main>
<div class="right-sidebar" contenteditable>Right Sidebar</div>
<footer contenteditable>Footer Content — Header.com 2020</footer>
```

CSS

```
body {
  display: grid;
  height: 100vh;
  grid-template-rows: auto 1fr auto;
}

// etc

header {
  background: lightpink;
  padding: 2rem;
}

main {
  background: coral;
}
```

```
footer {
  background: wheat;
  padding: 2rem;
  text-align: center;
}

body {
  font-family: system-ui, sans-serif;
}
```

05. 12-колонна мрежа grid-template: auto 1fr auto / auto 1fr auto

HTML

```
<div class="span-12">Span 12</div>
<div class="span-6">Span 6</div>
<div class="span-4">Span 4</div>
<div class="span-2">Span 2</div>
```

CSS

```
body {
  display: grid;
  height: 100vh;
  grid-template-columns: repeat(12, 1fr);
}

// etc

div {
  display: grid;
  place-items: center;
}

.span-12 {
  background: lightpink;
  grid-column: 1 / 13;
}

.span-6 {
  background: lightblue;
  grid-column: 1 / 7;
}

.span-4 {
```

```
  background: coral;
  grid-column: 4 / 9;
}

.span-2 {
  background: yellow;
  grid-column: 3 / 5;
}

body {
  font-family: system-ui, sans-serif;
}
```

06. RAM (Повторение, Автоматично, Минимум-Максимум) grid-template-columns: repeat(auto-fit, minmax(<base>, 1fr))

HTML

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

CSS

```
body {
  display: grid;
  height: 100vh;
  grid-gap: 1rem;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}

// etc

div {
  display: grid;
  place-items: center;
  background: lightpink;
}

body {
  font-family: system-ui, sans-serif;
}
```

07. Ограничаване на стила clamp(<min>, <actual>, <max>)

HTML

```
<div class="card">
<h1>Title Here</h1>
<div class="visual"></div>
<p>Descriptive Text. Lorem ipsum dolor si</p>
<br/>
<p>Lorem ipsum dolor sit amet cons</p>
</div>
```

CSS

```
body {
  display: grid;
  place-items: center;
  height: 100vh;
}

.visual {
  height: 100px;
  width: 100%;
  background: wheat;
  margin: 0.5rem 0;
}

.card {
  width: clamp(45ch, 50%, 75ch);
  display: flex;
  flex-direction: column;
  background: lightpink;
  padding: 1rem;
}

body {
  font-family: system-ui, sans-serif;
}

h1 {
  font-size: 1.5rem;
}
```

08. Зачитане на пропорциите aspect-ratio: <width> / <height>

HTML

```
<div class="card">
```

```
  <h1>Title Here</h1>
  <div class="visual"></div>
  <p>Descriptive Text. Lorem ipsum dolo</p>
</div>
```

CSS

```
body {
  display: grid;
  place-items: center;
  height: 100vh;
}

.visual {
  aspect-ratio: 16/9;
  background: wheat;
  margin: 0.5rem 0;
}

.card {
  width: 80%;
  display: flex;
  flex-direction: column;
  background: lightpink;
  padding: 1rem;
}

body {
  font-family: system-ui, sans-serif;
}

h1 {
  font-size: 1.5rem;
}
```

Ограничения и добри практики

Поддръжка на браузъри

CSS Grid е сравнително нова технология, която се поддържа от всички модерни браузъри. Към март 2017 г. повечето браузъри вече поддържат нативно CSS Grid без префикси:

- Chrome (включително на Android)
- Firefox
- Safari (включително на iOS)
- Opera

Internet Explorer 10 и 11 също поддържат CSS Grid, но с остарял синтаксис. Това означава, че ако трябва да поддържате IE, ще трябва да използвате по-стария синтаксис или да използвате fallback решения.

Достъпност (Accessibility)

Когато използвате CSS Grid, е важно да се уверите, че вашият код е достъпен за всички потребители. Ето няколко важни аспекта:

Подредба на съдържанието: CSS Grid позволява визуална промяна на подредбата на елементите, но това не трябва да засяга логическия ред на съдържанието. Уверете се, че DOM структурата следва логическия ред на информацията.
Четливост: Избягвайте прекалено сложни grid структури, които могат да затруднят навигацията с клавиатура или екранни четци.
Отзивчивост: При проектиране на отзивчиви layouts, уверете се, че промяната в подредбата не води до объркване или загуба на контекст за потребителите.

Добри практики

Семантична структура
Използвайте семантични HTML елементи заедно с CSS Grid. Това подобрява достъпността и поддръжката на кода.

Прогресивно подобрение
Предоставете fallback стилове за браузъри, които не поддържат CSS Grid. Това осигурява базова функционалност за всички потребители.

Оптимизация на производителността
Избягвайте прекалено сложни grid структури, които могат да забавят рендирането. Използвайте grid-template-areas за по-ясна и поддържаема структура.

Тестване
Регулярно тествайте вашите grid layouts в различни браузъри и устройства. Използвайте инструменти за разработчици за откриване на потенциални проблеми.

Често срещани проблеми

Прекалено сложни grid структури: Това може да доведе до трудности при поддръжка и лоша производителност.
Несъответствие между визуална и DOM структура: Може да създаде проблеми с достъпността и SEO.
Липса на fallback решения: Може да доведе до лошо потребителско изживяване в по-стари браузъри.
Прекалено голяма зависимост от фиксирани размери: Това може да създаде проблеми при отзивчиви дизайни.

Заключение

CSS Grid Layout представлява революционна промяна в начина, по който създаваме уеб интерфейси. Като първия CSS модул, създаден специално за решаване на проблеми с оформлението, той предлага безпрецедентна гъвкавост и мощност.

Ключови предимства

Двуизмерно оформление
За разлика от Flexbox, който работи само в едно измерение, CSS Grid позволява едновременен контрол върху редовете и колоните, което го прави идеален за сложни layouts.

Гъвкавост
Съчетаването на фиксирани и гъвкави размери, автоматично разпределение и мощни функции като minmax() и repeat() позволява създаването на адаптивни дизайни с минимален код.

Поддръжка
С поддръжка от всички модерни браузъри, CSS Grid вече е напълно готов за производствена употреба. Въпреки че IE изисква специален синтаксис, това не е пречка за внедряването му.

Семантика
Grid позволява разделяне на визуалното оформление от HTML структурата, което подобрява достъпността и поддръжката на кода.

Бъдещето на CSS Grid

CSS Grid продължава да се развива с нови функции и подобрения. Някои от най-интересните разработки включват:

Subgrid: Позволява на вложени grid контейнери да наследяват размерите на родителския grid, което решава дългогодишни проблеми с вложени layouts.
Auto-placement: Става все по-мощен с нови ключови думи като dense и подобрена поддръжка за автоматично разпределение на елементи.
Grid Level 2: Включва нови функции за по-добро управление на grid областите и по-голяма гъвкавост при създаване на сложни layouts.

Практически приложения

CSS Grid вече се използва успешно в множество реални проекти:

Съвременни уеб приложения: За създаване на сложни, адаптивни интерфейси с минимален код.
Дизайн системи: За стандартизиране на layouts и подобряване на консистентността.
Отзивчиви уебсайтове: За създаване на layouts, които се адаптират плавно към различни размери на екрана.

Интерактивни приложения: За динамично преоформление на съдържанието без JavaScript.

В заключение, CSS Grid представлява мощна и зряла технология, която променя начина, по който мислим за уеб оформление. С неговата поддръжка от всички модерни браузъри и постоянно развиващите се възможности, той е незаменим инструмент в арсенала на всеки съвременен уеб разработчик.

Източници

[1] Flavio Copes, "The CSS Handbook: A Handy Guide to CSS for Developers", зададен/публикуван April 24, 2019, [https://www.freecodecamp.org/news/the-css-handbook-a-handy-guide-to-css-for-developers-b56695917d11/#css-grid]

[2] Una Kravets, "1 Line Layouts", зададен/публикуван [https://1linelayouts.glitch.me/]

[3] Una Kravets, Jeremy Wagner, Vladimir Levin, "content-visibility: the new CSS property", зададен/публикуван August 5, 2020 [https://web.dev/content-visibility/]

[4] Chris House, "A Complete Guide to Grid", зададен/публикуван Sep 26, 2024 [https://css-tricks.com/snippets/css/complete-guide-grid/]

[5] W3Schools, "CSS Grid Layout", зададен/публикуван [https://www.w3schools.com/css/css_grid.asp]

[6] Mikhaél Minisini, "Web layouts - Part 2: State of the art, Flexbox and Grid.", зададен/публикуван April, 2020, [https://apptitude.ch/en/development/web-layout-part-2-flexbox-and-grid/]