GRID LAYOUT

Waldemar Korłub

Aplikacje i Usługi Internetowe
KASK ETI Politechnika Gdańska

Pozycjonowanie elementów na stronie

- Atrybuty position/float
  - Rozwiązania z czasów przed rewolucją mobilną
- FlexBox
  - Jednowymiarowy kontener
    - Orientacja pozioma LUB pionowa
- Grid Layout
  - Dwuwymiarowy kontener
  - https://caniuse.com/#search=grid%20layout

Grid Layout

- Umożliwia pozycjonowanie elementów na dwuwymiarowej siatce wierszy i kolumn
- Wymiary wierszy i kolumn można definiować:
  - Przy użyciu jednostek bezwzględnych (np. px)
  - Proporcjonalnie do dostępnej przestrzeni
    - np. procentowo
    - Nowa jednostka na potrzeby gridu: fr
- Elementy mogą zajmować wybraną liczbę kolumn **oraz wierszy**
- W przypadku nachodzących na siebie elementów wykorzystywany atrybut z-index

Kontener

- Kontener dla gridu definiowany za pomocą atrybutów:
  - display: grid

    lub
  - display: inline-grid

```
<div class="wrapper">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

Arkusz CSS:

```
.wrapper {
  display: grid;
}
```

Kolumny i wiersze

- Kolumny i wiersze definiowane za pomocą atrybutów:
  - grid-template-columns
  - grid-template-rows
- Przykładowo:

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 400px 200px;
}
```

Jednostka fr

- Rozmiary wierszy i kolumn można definiować z użyciem wszystkich miar długości
- Na potrzeby Grid Layout wprowadzono nową jednostkę: fr
- Jednostki fr reprezentują ułamek dostępnej przestrzeni
- Umożliwiają proporcjonalny podział dostępnego miejsca pomiędzy wiersze/kolumny

Jednostka fr

- Przykładowo:

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 3fr 2fr;
}
```

- Zdefiniowane 3 kolumny
- Dostępna szerokość zostanie podzielona między kolumny w proporcjach 1:3:2
- Jeśli wrapper ma szerokość 960px:
  - Pierwsza kolumna: 960/(1+3+2)*1 = 160
  - Druga kolumna: 960/(1+3+2)*3 = 480
  - Trzecia kolumna: 960/(1+3+2)*2 = 320

Jednostka fr

- Jednostki fr można łączyć z innymi, przykładowo:

```
.wrapper {
  display: grid;
  grid-template-columns: 100px 3fr 2fr;
}
```

- Pierwsza kolumna ma szerokość 100px niezależnie od szerokości gridu
- Pozostała szerokość gridu jest dzielona pomiedzy drugą i trzecią kolumnę w proporcjach 3:2

Składnia repeat()

- W przypadku powtarzających się deklaracji można zastosować składnię repeat(), np.:

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
}
```

  - Definicja gridu o 12 kolumnach równej szerokości

- Składnię repeat() można łączyć z innymi definicjami:

```
.wrapper {
  display: grid;
  grid-template-columns: 30px repeat(4, 1fr) 10%;
}
```

Wiersze

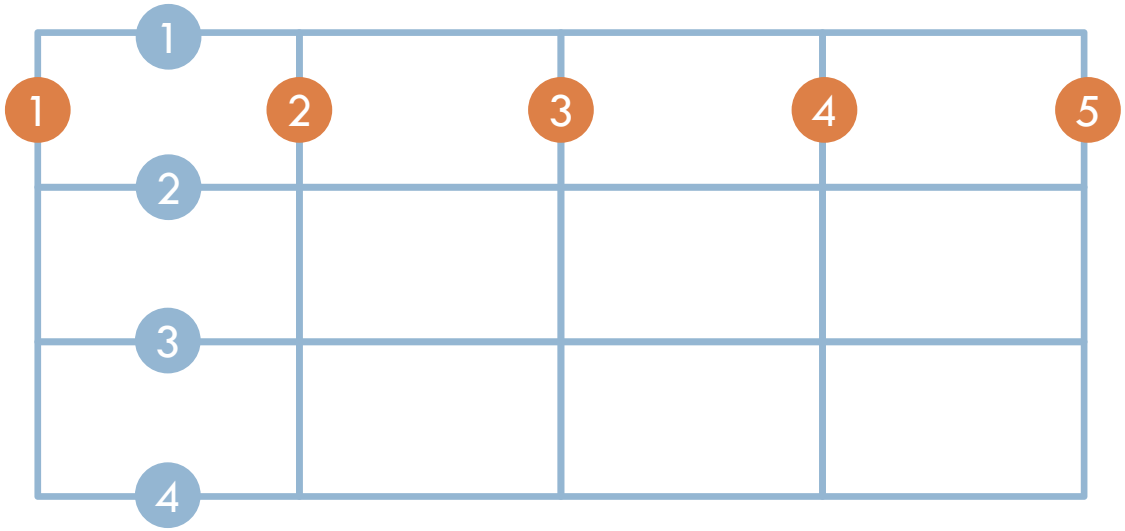
- Wiersze można zdefiniować jawnie za pomocą atrybutu grid-template-rows
- W przypadku jego braku mamy do czynienia z niejawną definicją
  - Wiersze są dodawane automatycznie tak, aby grid zmieścił całą zawartość
- Wymiary niejawnych wierszy:
  - Predefiniowana wysokość:
    grid-auto-rows: 200px;
  - Składnia minmax():
    grid-auto-rows: minmax(100px, auto);

Odstępy między wierszami/kolumnami

- Możliwe jest zdefiniowanie odstępów pomiędzy wierszami/kolumnami w gridzie:
  - Dla kolumn:
    - grid-column-gap: 10px
  - Dla wierszy:
    - grid-row-gap: 10px
  - Zbiorczy atrybut:
    - grid-gap: 10px

Linie gridu

- Struktura gridu jest definiowana poprzez określenie liczby wierszy i kolumn
- Pozycjonowanie elementów w gridzie nie odbywa się względem wierszy/kolumn, ale **względem linii definiujących ich granice**



Pozycjonowanie elementów w gridzie

- Pozycję elementu w gridzie określają atrybuty:
  - grid-column-start
  - grid-column-end
  - grid-row-start
  - grid-row-end
- Wartości powyższych atrybutów odnoszą się do numerów linii gridu
- Jeśli atrybuty nie zostaną zdefiniowane domyślnie element zajmuje jedną komórkę w gridzie
  - …i są lokowane w kolejności definicji w pliku HTML na kolejnych wolnych komórkach gridu

Przykładowy układ

Paczka ”08 grid layout.zip”

Skrócone atrybuty

- grid-column
  - Łączy grid-column-start oraz grid-column-end
  - Przykładowo:
    grid-column: 1 / 3;
  - Odpowiada:
    grid-column-start: 1;
    grid-column-end: 3;
- grid-row
  - Łączy grid-row-start oraz grid-row-end
- grid-area
  - Łączy grid-row-start, grid-column-start, grid-row-end, grid-column-end (w tej kolejności)
  - Grid-area: 2 / 1 / 4 / 3;

Atrybuty pozycjonowania

- Wartości dla atrybutów *-end nie są obowiązkowe
  - Domyślnie element zajmuje jeden wiersz/kolumnę
- Zamiast podawać początek i koniec obszaru można podać początek i liczbę kolumn/wierszy, np.:
  - Składnia span:
    grid-column: 2 / span 2;
- Do linii gridu można odnosić się również za pomocą liczb ujemnych (od prawej do lewej), np.:
  - grid-column: 1 / -1;

Nazwane linie gridu

- Liniom gridu można nadawać nazwy, np.:

```
.wrapper {
  display: grid;
  grid-template-columns:
    [main-start] 1fr [content-start] 1fr [content-end] 1fr [main-end];
  grid-template-rows:
    [main-start] 100px [content-start] 100px [content-end] 100px [main-end];
}
```

- Nazwy można wykorzystać w miejscu numerów linii:

```
.content {
  grid-column: content-start / content-end;
}
```

Szablony

- Grid Layout umożliwia zdefiniowanie szablonu gridu
  - Atrybut grid-template-areas, np.:

```
.wrapper {
    grid-template-columns: repeat(6, 1fr);
    grid-template-areas:
            "head head head head head head"
            "side side main main main advs"
            "foot foot foot foot foot foot";
}
```

- Elementy potomne można przypisać do zdefiniowanych tak obszarów
  - Zamiast przypisywać im linie początkowe/końcowe:

```
.content {
  grid-area: main;
  //zamiast: grid-area: 2 / 3 / 3 / 6;
}
```

Szablony

- Obszary w obrębie szablonu muszą mieć kształt prostokąta
- Liczba kolumn musi zgadzać się z atrybutami grid-template-column
  - W przeciwnym razie szablon zostanie zignorowany przez przeglądarkę
- Mechanizm szablonów jest szczególnie przydatny w połączeniu z progami responsywnymi

Responsywne szablony: *mobile-first*


```
.wrapper {
    display: grid;
    grid-auto-rows: minmax(100px, auto);
    grid-gap: 10px;

    grid-template-columns: 1fr;
    grid-template-areas:
            "head"
            "side"
            "main"
            "advs"
            "foot";
}
```

Responsywne szablony


```
@media (min-width: 768px) {
    .wrapper {
        grid-template-columns: 1fr 3fr;
        grid-template-areas:
                "head head"
                "side main"
                "advs advs"
                "foot foot";
    }
}

@media (min-width: 992px) {
    .wrapper {
        grid-template-columns: repeat(6, 1fr);
        grid-template-areas:
                "head head head head head head"
                "side side main main main advs"
                "foot foot foot foot foot foot";
    }
}
```

Przykładowy układ

Paczka ”08b grid layout.zip”

Grid Layout

- Grid Layout posiada istotne zalety w porównaniu do wcześniejszych rozwiązań:
  - Grid jest responsywny
    - Dla różnych progów responsywnych można wykorzystywać różne liczby kolumn
    - Dostępne wcześniej biblioteki oferowały typowo statyczny grid, np. 12 kolumn
  - Kontener definiuje układ dla elementów potomnych
    - Wcześniej rozmiar kontenera wynikał ze stylów elementów potomnych
  - Elementy mogą rozpościerać się na wiele wierszy gridu
    - Wcześniej typowo każdy wiersz gridu definiowany osobno

Pytania?

Dziękuję za uwagę