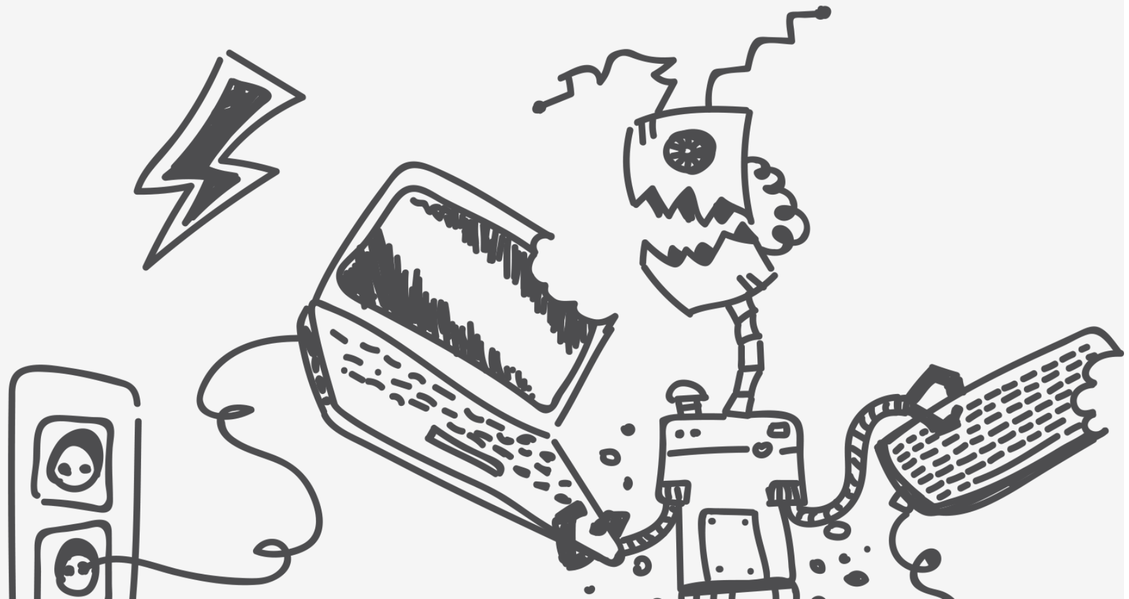


Grid Layout

Grid

Co to takiego?

- **Grid** (tzw. siatka) to układ strony, w którym treść ułożona jest w kolumnach i rzędach.
- Na węższych ekranach kolumny z reguły są wyświetlane jedna pod drugą, każda z nich stanowi wtedy nowy rząd – w celu lepszej prezentacji i czytelności treści.
- Grid powinien być również skalowalny tak, aby suma szerokości kolumn w jednym rzędzie, niezależnie od ich liczby, nie przekraczała szerokości ekranu. Najlepiej do określenia szerokości kolumny użyć zatem jednostek względnych takich jak procenty czy emy.

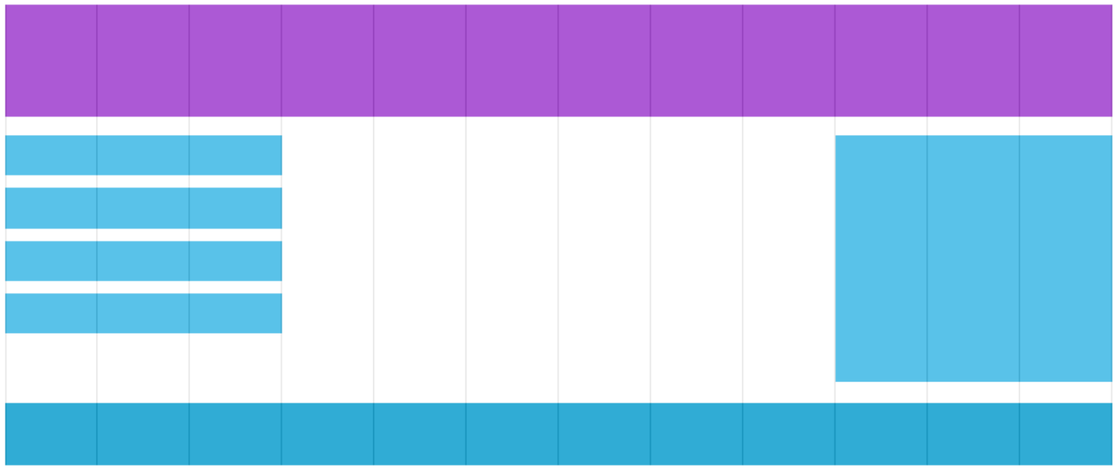
Grid

Po co używać grida?

- Większość nowoczesnych projektów graficznych zakłada ustawienie elementów w kolumnach.
- Aby łatwo ustawić trzy elementy obok siebie w wersji desktopowej (ale w wersji mobilnej wyświetlić je jeden po drugim), należy właśnie użyć **grida**.
- Użycie **grida** znacznie przyspiesza tworzenie stron i aplikacji – gdy mamy już stworzone odpowiednie klasy **grida** możemy pozycjonować elementy HTML za ich pomocą a nie każdy element oddzielnie.

Grid ułatwia pozycjonowanie elementów

- Oparcie layoutu strony na siatce znacznie ułatwia pozycjonowanie elementów.



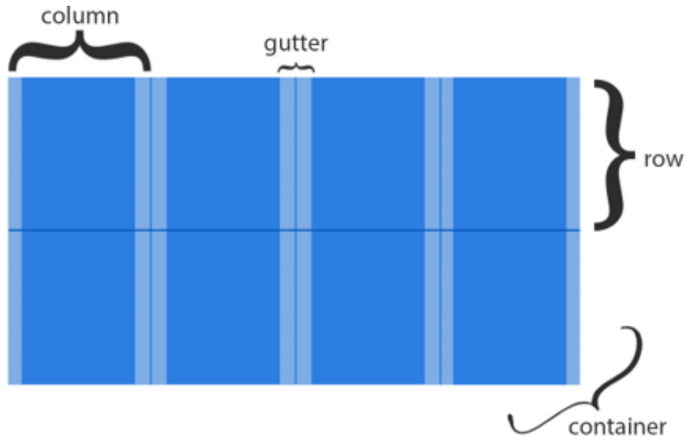
Przykład: http://suprb.com/apps/gridalicious

Grid

Konstrukcja grida

Grid składa się z następujących elementów:

- container,
- rows,
- columns,
- gutters (odstępy między kolumnami).



Jak zrobić swój grid?

Container

Celem kontenera jest ustawienie szerokości dla całego grida.

Najlepiej ustawić szerokość na 100%, ale warto też dodać `max-width` i wpisać wybraną przez siebie szerokość.

Nie zapominajmy też o ustawieniu poziomych marginesów na `auto` aby nasz kontener zawsze znajdował się na środku ekranu.

```
.container {
  width: 100%;
  max-width: 990px;
  margin: 0 auto;
  padding: 0 15px;
}
```

Jak zrobić swój grid?

Row

Celem wiersza jest przetrzymywanie kolumn oraz pilnowanie, aby kolumny nie opływały innych kolumn w innych wierszach.

Dla wiersza ustawiamy zatem `display: flex;` aby zmusić elementy do układania obok siebie.

Dodamy również `flex-wrap: wrap;` po to by kolumny po przekroczeniu 100% szerokości rzędu spadły pod siebie.

```
.row {
  display: flex;
  flex-wrap: wrap;
}
```

Jak zrobić swój grid?

Column – pozycjonowanie

Warto ustawić wszystkim klasom rozpoczynającym się od `col-`, `min-height: 1px`, aby zapobiec nachodzeniu na siebie kolumn, gdy będą puste.

```
[class*='col-'] {
  min-height: 1px;
}
```

Jak zrobić swój grid?

Column – szerokość

Aby ustawić szerokość jednej kolumny, musimy podzielić szerokość całego kontenera przez liczbę wszystkich kolumn.

Przykład

Chcemy użyć dwunastu kolumn.

Nasz kontener ma **100%** szerokości, więc:

100/12 = 8,333

Szerokość bazowa jednej kolumny to 8,333%.

```
$columns: 12;
$column-base-width: 100% / $columns;

[class*='col-'] {
  min-height: 1px;
  width: $column-base-width;
}
```

Jak zrobić swój grid?

Column – szerokość

Teraz obliczymy szerokość innych kolumn:

8,333 * N

W naszym przypadku posiadamy sześć kolumn - liczymy szerokość dla wszystkich możliwości w przedziale **1 - 12**.

```
.col-1 {
  width: 8.33333%;
}
.col-2 {
  width: 16.66667%;
}
.col-3 {
  width: 25%;
}

// Kolumny od 4 do 10

.col-11 {
  width: 91.66667%;
}
.col-12 {
  width: 100%;
}
```

Jak zrobić swój grid?

Column – szerokość

Oczywiście nie będziemy robić tego ręcznie! Z pomocą przychodzi Sass i możliwość użycia pętli `@for`. Niech to ona stworzy nam zadane klasy na podstawie wartości początkowej - `1` i końcowej - `$columns` (w naszym przykładzie `12`).

Taka pętla wytworzy nam określoną ilość klas o podanych parametrach.

```
$columns: 12;
$column-base-width: 100% / $columns;

@for $i from 1 through $columns {
  .col-#{$i} {
    width: $column-base-width * $i;
  }
}
```

Jak zrobić swój grid?

Gutters – odstępy

Ustawienie odstępów mogłoby być problematyczne ale na szczęście możemy ustawić odpowiedni `box-sizing`.

```
* {
  box-sizing : border-box;
}

[class*='col-'] {
  min-height: 1px;
  width: $column-base-width;
  padding: 12px;
}
```

Media queries – przykładowa konfiguracja

Po prawej znajduję się wykorzystanie `media queries` wraz z **gridem** - na ekranach do **550px** szerokości ustawiamy szerokość kolumn na **100%** - kolumny ustawiają się wtedy pod sobą.

```
@media (max-width: 550px) {
  [class*='col-'] {
    width: 100%;
  }
}
```

Grid i responsywność

RWD

- Treść rozmieszczona w oparciu o grid dopasuje się do danego urządzenia – takie jest założenie responsywności.
- Jeśli elementy umieszczone obok siebie staną się mało czytelne, to powinny wyświetlać się jeden pod drugim a ich szerokość dopasować do ekranu.

Responsywna siatka

- Tworzenie responsywnej siatki może być procesem skomplikowanym i czasochłonnym.
- Wymaga wnikliwego podejścia, odpowiedniego zaplanowania i dobrego opanowania technik CSS.
- Dobrze przygotowana siatka jest narzędziem uniwersalnym i może być wielokrotnie wykorzystana.
- Można również skorzystać z siatek, które zostały już stworzone i sprawdzone.

Frameworki zawierające grid

Dwa najpopularniejsze i najbardziej rozbudowane frameworki to:

- Bootstrap,
- Foundation.

Najważniejsze różnice między frameworkami to:

- możliwość zagnieżdżania kolumn,
- szerokość domyślnych punktów granicznych.