Stylowanie po znacznikach

```
a {
    text-decoration: none;
    color: #ec008c;
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}

ul {
    padding-left: 10px;
    list-style-type: none;
}

h2 {
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}
```

```
.concerts__info  span  p {
    margin: 0;
    letter-spacing: 1px;
}
```

1. Nadawajcie elementom klasy zamiast stylować po znacznikach
2. Lepiej nadać znacznikowi „p” klasę, niżeli tworzyć tak skomplikowany selektor
3. Niepotrzebnie używać tak wysokich poziomów specyficzności

Niepoprawne wartości własności CSS

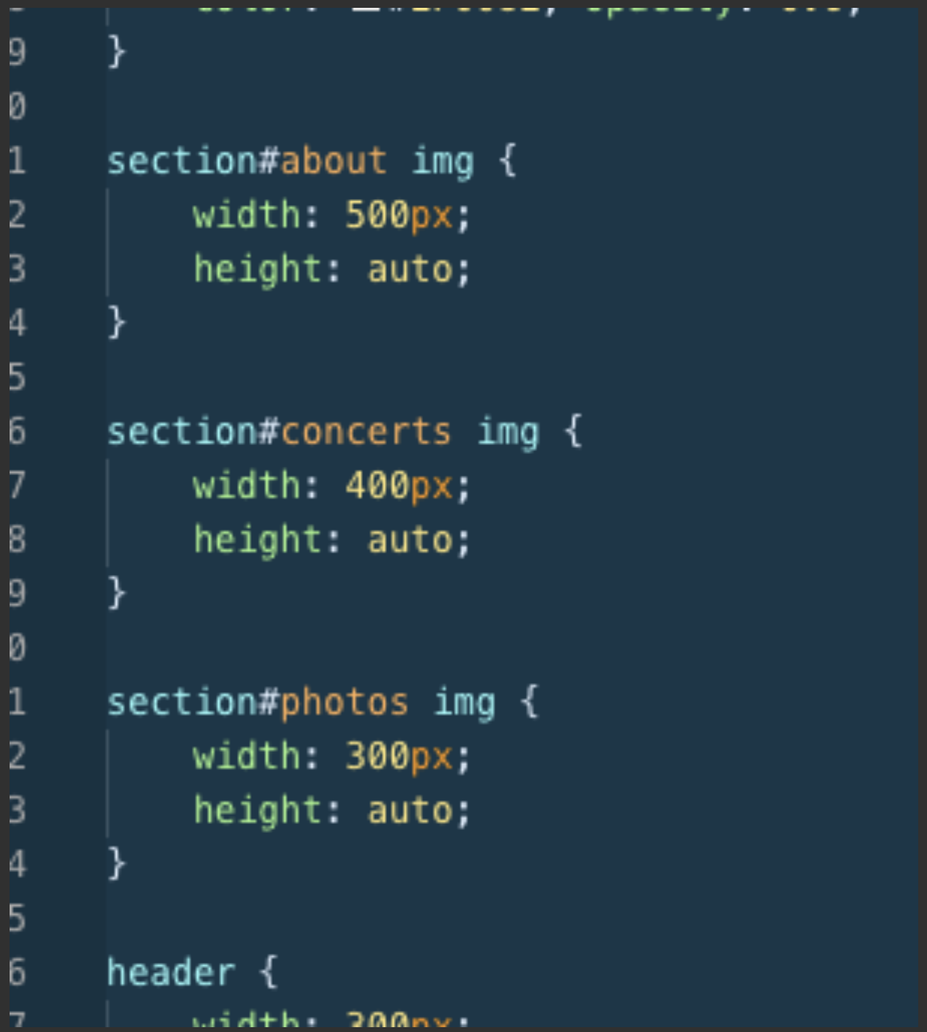
```css
.page-content
{
    position: relative;
    left: 25%;
    margin-left: 220px;
    height: min-height;
}
```

```css
.content-section about-section {
  position:right;
  width:100px;
  height:100px;

}
```

1. Sprawdzajcie czy wartość, którą stosujecie, można nadać określonej własności - używajcie do tego MDN
2. Po dzisiejszych zajęciach będziecie potrafili użyć do tych celów DevTools

Nadużywanie ID



```
}

section#about img {
    width: 500px;
    height: auto;
}

section#concerts img {
    width: 400px;
    height: auto;
}

section#photos img {
    width: 300px;
    height: auto;
}

header {
```

1. Unikajcie używania ID, chyba, że czujecie, że jest niezbędne
2. Takie sytuacje to: id wykorzystywane do nadania animacji, formularza kontaktowego
3. PS: tutaj widać też przykład zbyt wysokiej specyficzności

Nadawanie indywidualnych czcionek

```css
a {
    text-decoration: none;
    color: #ec008c;
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}

ul {
    padding-left: 10px;
    list-style-type: none;
}

h2 {
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}
```

1. Nadawajcie czcionkę np. Całemu body lub div'owi, który trzyma w środku całą stronę
2. Jeśli na stronie stosowanych jest więcej niż jedna czcionka, to ogólną powinna być ta, która najczęściej występuje

Brak font-stack'ów

```
a {
    text-decoration: none;
    color: #ec008c;
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}

ul {
    padding-left: 10px;
    list-style-type: none;
}

h2 {
    font-family: "OpenSans","Lucida Console", monospace;
    letter-spacing: 3.8px;
}
```

1. Są niezbędne, jeśli nie chcecie by wasz tekst został wyświetlony z wyjściową czcionką przeglądarki - nie jest to ładne

Wykorzystywanie float

```
.band01_image {
    float: right;
    height: 500px;
    width: 900px;
   margin: 20px;
   padding: 20px;

}
```

1. Nie wykorzystujecie float. Jest to przestarzała technologia stosowana w specyficznych przypadkach
2. Zamiast tego, po dzisiejszym wykładzie, będziecie korzystać z flex'a

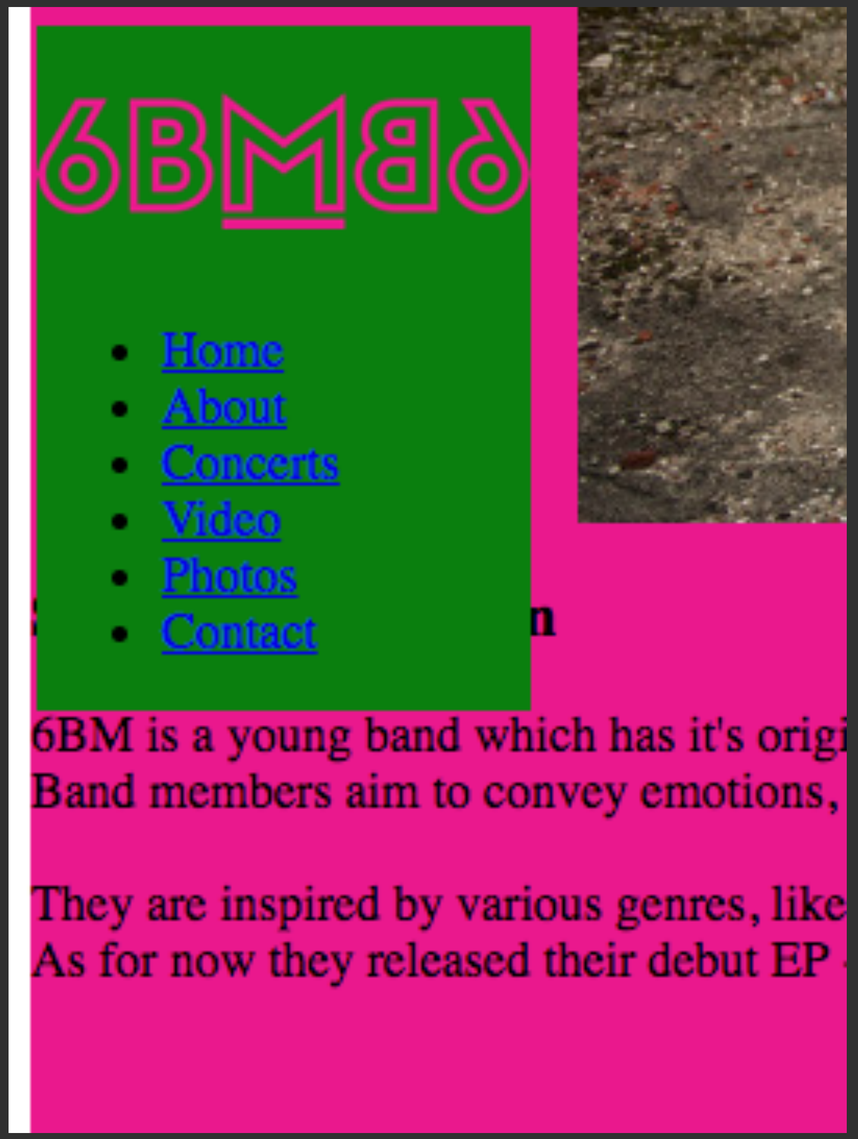
Brak własności wspierających dla background-image

1. Wiele z was nie wykorzystywało własności takich jak: size, position, repeat dla background-image
2. Powoduje to, że wiele zdjęć kurczy się lub rozjeżdża przy zmianach rozdzielczości

```
#main-page, #concerts, #photos {
    background-image: url("../img/background.svg");
    background-size: cover;
}

#about, #video, #contact {
    background-color: #25072e;
}
```

Nie pozostawianie miejsca na menu/footer



1. Menu i footer nie powinny najeżdżać na treść strony
2. Można to rozwiązać pozostawiając miejsce, tzn odsuwając treść strony, tak by menu/footer na nią nie nachodziły
3. Jedno z możliwych rozwiązań załączam w codepen poniżej

Codepen: space-for-menu

Height zamiast min-height

1. Wykorzystanie height, sprawi, że gdy content strony będzie zajmował więcej niż jej height, to ‚wyleje się' nam on
2. Min-height sprawi, że container będzie utrzymywał jakąś minimalną wysokość, a jeśli treść będzie miała jej więcej, to rozciągnie się z nią