Technologia i rozwiązania

# Responsive Web Design

## Nowoczesne strony WWW na przykładach

I Ty możesz stworzyć nowoczesną witrynę WWW!

Thoriq Firdaus

Helion

[PACKT] PUBLISHING

Tytuł oryginału: Responsive Web Design by Example

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-9190-6



First published in the English language under the title „Responsive Web Design by Example".



Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/reweno
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

# Spis treści

3

ROZDZIAŁ

# Ulepszanie strony portfolio przy użyciu CSS3

*W poprzednim rozdziale napisaliśmy strukturę HTML5 naszej strony portfolio. W tym rozdziale dodamy do niej arkusze stylów CSS3. Zaczniemy od zdefiniowania paru prostych efektów wizualnych, które zostały niedawno wprowadzone w CSS3, takich jak cienie tekstu i bloków oraz zaokrąglenie rogów.*

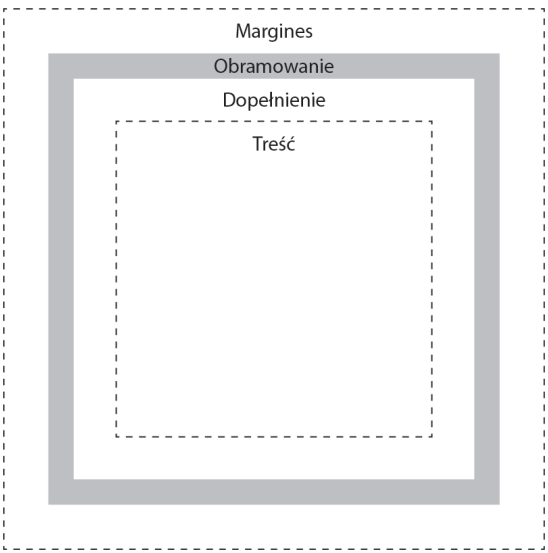
*Następnie utworzymy bardziej skomplikowany efekt, polegający na zmianie obrazu po najechaniu na niego kursorem. Przed pojawieniem się technologii CSS3 można to było osiągnąć wyłącznie przy użyciu JavaScriptu.*

W tym rozdziale:

- Sformatujesz styl strony — zaczniemy od nagłówka i nawigacji, potem przejdziemy do obszaru treści, a skończymy na stopce, wykorzystując podczas pracy różne nowe własności CSS3.
- Utworzysz funkcję filtrującą portfolio przy użyciu CSS3.
- Stworzysz efekt zmiany obrazu po najechaniu kursorem przy użyciu przekształceń i przejść CSS3.
- Przystosujesz wygląd strony do ekranów o różnych szerokościach przy użyciu zapytań medialnych CSS3.

# Model polowy CSS

Element blokowy HTML jest w istocie polem. Na pole to składa się treść właściwa elementu oraz jego marginesy, dopełnienie i obramowanie zdefiniowane przy użyciu CSS, jak widać na poniższym rysunku:




Więcej informacji na temat różnic między elementami blokowymi i śródliniowymi oraz listę elementów można znaleźć na poniższych stronach:

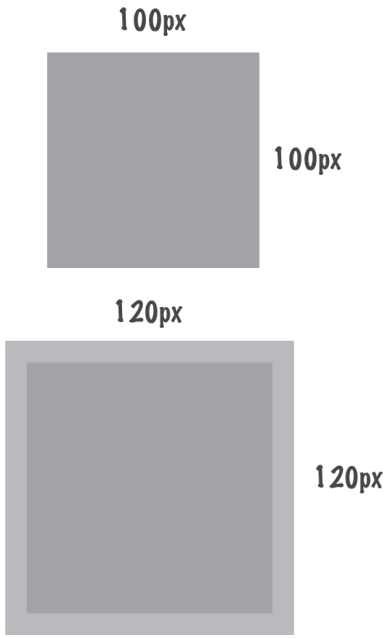
- Elementy blokowe: *https://developer.mozilla.org/en-US/docs/HTML/Block-level_elements*.
- Elementy śródliniowe: *https://developer.mozilla.org/en-US/docs/HTML/Inline_elements*.

Przed pojawieniem się CSS3 określanie właściwości pola elementu było nieco utrudnione. Załóżmy na przykład, że tworzymy kwadrat o boku 100 pikseli:

```
div {
    width: 100px;
    height: 100px;
}
```

Taki kod przeglądarka zinterpretuje jako zwykły kwadrat (pierwszy rysunek na następnej stronie).

Ale będzie tak tylko dopóty, dopóki nie zdefiniujemy dopełnienia, marginesu albo obramowania. Jako że pole elementu ma cztery boki, zdefiniowanie dopełnienia o rozmiarze 10 pikseli (`padding: 10px`) spowoduje zwiększenie szerokości i wysokości elementu o 20 pikseli — po 10 z każdej strony (drugi rysunek na następnej stronie).



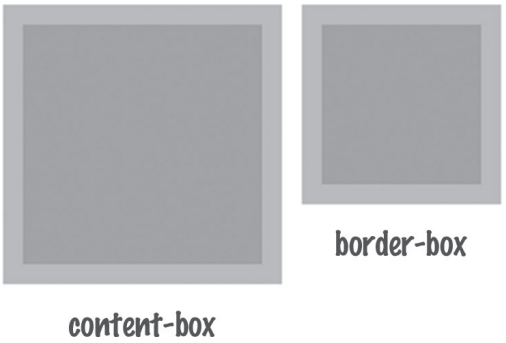

Nie dość, że zajmowane jest dodatkowe miejsce, to jeszcze obszar marginesu znajduje się poza elementem, to znaczy nie należy do jego obszaru. Jeśli więc zdefiniujemy elementowi kolorowe tło, kolor ten nie będzie widoczny w obszarze zajmowanym przez margines.

## Własność CSS3 box-sizing

W CSS3 dostępna jest własność `box-sizing`, za pomocą której można kontrolować właściwości modelu polowego.

| Wartość | Opis |
|---|---|
| `content-box` | Jest to standardowe ustawienie modelu polowego. Wartość ta sprawia, że dopełnienie i obramowanie pola nie są wliczane do jego zdefiniowanych szerokości i wysokości, jak zostało pokazane na rysunku w poprzednim podrozdziale. |
| `border-box` | Ta wartość jest przeciwieństwem poprzedniej. Sprawia, że dopełnienie i obramowanie są wliczane do zdefiniowanych szerokości i wysokości pola. |

Wróćmy do naszego przykładu. Ustawimy własność `box-sizing` na `border-box`. Dzięki temu niezależnie od dopełnienia i obramowania szerokość i wysokość elementu będą wynosić po 100 pikseli. Na poniższym rysunku przedstawiam różnicę między dwoma opisywanymi ustawieniami:




## Czas na działanie — definiowanie własności box-sizing

Otwórz plik *styles.css* i na samym początku wpisz poniższą regułę:

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

### Co uzyskaliśmy?

Użyliśmy selektora uniwersalnego CSS (*) w celu zastosowania własności `border-box` do wszystkich elementów blokowych na stronie, aby ułatwić sobie ustawianie ich szerokości i wysokości.

> Jako pierwszy poradę tę opublikował Paul Irish, główny programista skryptu Modernizr i HTML5 Boilerplate. Więcej informacji o tej metodzie można znaleźć w jego wpisie na stronie *http://www.paulirish.com/2012/box-sizing-border-box-ftw/*.

# Jednostki miary CSS

W specyfikacji CSS opisano kilka jednostek miary. Na naszej stronie będziemy używać głównie jednostek `px`, `em` oraz procentów.

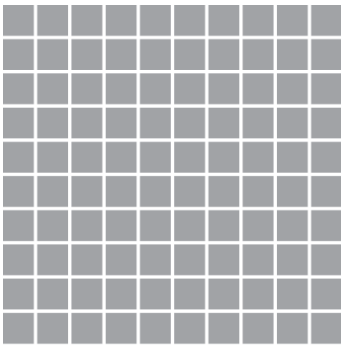
## Piksele

Piksel (`px`) to bezwzględna jednostka, która jest używana chyba najczęściej ze wszystkich dostępnych jednostek. Za jej pomocą można precyzyjnie określić rozmiary elementów. W specyfikacji

CSS znajdującej się pod adresem *http://www.w3.org/TR/css3-values/#reference-pixel* piksel jest zdefiniowany jako:

> *Rozmiar kątowy jednego piksela na urządzeniu o gęstości 96 dpi.*

Według tej definicji 1 piksel CSS na ekranie o gęstości 96 dpi jest równy 1 pikselowi urządzenia. Zatem 10 pikseli w CSS odpowiada 10 pikselom urządzenia.

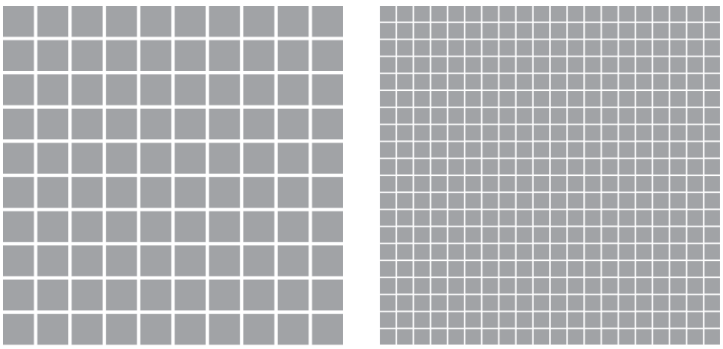


Normalny ekran
(96 dpi)

W tym projekcie rozmiary pól będziemy określać w pikselach.

## Piksele na ekranach o większej gęstości

Ze względu na upowszechnianie się ekranów o coraz wyższych rozdzielczościach przykład opisany w poprzednim podrozdziale jest już nieaktualny. Poniższy rysunek przedstawia ekran o wysokiej rozdzielczości 192 dpi. Element o szerokości i wysokości 10 pikseli na takim ekranie zajmie 20 pikseli urządzenia:



Normalny ekran
(96 dpi)

Ekran o wyższej rozdzielczości
(192 dpi)

Rozmiar elementu na ekranie pozostanie taki sam, ale będzie się on składał z większej liczby pikseli urządzenia.

Na temat piksela i jego relacji z rozdzielczością ekranu toczy się wiele dyskusji, np.:

- Reda Lemeden w swoim wpisie przedstawił trudności związane z projektowaniem dla ekranów o różnych gęstościach (*http://coding.smashingmagazine.com/2012/08/20/towards-retina-web/*).
- Scott Kellum w serwisie A List Apart opisał jednostkę piksel i jej zastosowanie do definiowania rozmiaru elementów na stronach przeznaczonych dla różnych urządzeń i ekranów o różnych rozdzielczościach (*http://alistapart.com/article/a-pixel-identity-crisis*).

# Jednostka em

Jednostka `em` jest względna i odnosi się do rozmiaru wielkiej litery *M* używanego fontu. W CSS `1em` zasadniczo odnosi się do rozmiaru pisma ustawionego w urządzeniu lub dokumencie. Jeśli element nadrzędny ma określony rozmiar pisma w jednostce `em`, to rozmiar pisma jego elementów potomnych będzie ustawiany względem rozmiaru pisma tego elementu nadrzędnego.

W naszym projekcie jednostki `em` będziemy używać do określania rozmiaru pisma zgodnie z zaleceniami W3C (*http://www.w3.org/Style/Examples/007/units.en.html*).

## Konwertowanie pikseli na em

Domyślny rozmiar pisma w systemie Skeleton jest ustawiony w pliku *base.css* na 14 pikseli, więc do tego rozmiaru będą się odnosić ustawienia wyrażone w jednostce `em`.

Powiedzmy, że chcemy się dowiedzieć, ile jednostek `em` odpowiada rozmiarowi 20 pikseli, gdy podstawowy rozmiar pisma wynosi 14 pikseli. Istnieje narzędzie ułatwiające zamienianie jednostek `px` na `em` (i odwrotnie) o nazwie PXtoEM.com (*http://pxtoem.com/*). Na zrzucie ekranu (na następnej stronie) pokazałem sposób jego użycia.

Jak widać na zrzucie, przy podstawowym rozmiarze pisma 14 pikseli rozmiarowi `20px` odpowiada wartość `1.429em`.

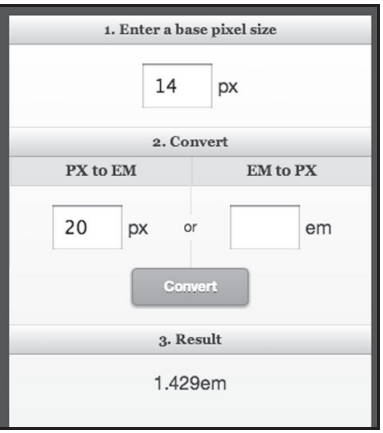
## Samodzielne obliczanie wartości em

Jednostki `px` i `em` można także przeliczać przy użyciu wzorów przedstawionych w poniższej tabeli:

| Rodzaj konwersji | Wzór | Przykład |
|---|---|---|
| `px` na `em` | Rozmiar (px) / podstawowy rozmiar pisma (px) | 20(px) / 14(px) = `1.429em` (po zaokrągleniu) |
| `em` na `px` | Rozmiar (em) * podstawowy rozmiar pisma (px) | 1,429(em) * 14(px) = 20px |

| Percent | Points |
|---|---|
| 37.5% | 5pt |
| 43.8% | 5pt |
| 50.0% | 6pt |
| 56.3% | 7pt |
| 62.5% | 8pt |
| 68.8% | 8pt |
| 75.0% | 9pt |
| 81.3% | 10pt |
| 87.5% | 11pt |
| 93.8% | 11pt |
| 100.0% | 12pt |
| 106.3% | 13pt |
| 112.5% | 14pt |

| Pixels | EMs | Percent | Points |
|---|---|---|---|
| 6px | 0.429em | 42.9% | 5pt |
| 7px | 0.500em | 50.0% | 5pt |
| 8px | 0.571em | 57.1% | 6pt |
| 9px | 0.643em | 64.3% | 7pt |
| 10px | 0.714em | 71.4% | 8pt |
| 11px | 0.786em | 78.6% | 8pt |
| 12px | 0.857em | 85.7% | 9pt |
| 13px | 0.929em | 92.9% | 10pt |
| 14px | 1.000em | 100.0% | 11pt |
| 15px | 1.071em | 107.1% | 11pt |
| 16px | 1.143em | 114.3% | 12pt |
| 17px | 1.214em | 121.4% | 13pt |
| 18px | 1.286em | 128.6% | 14pt |

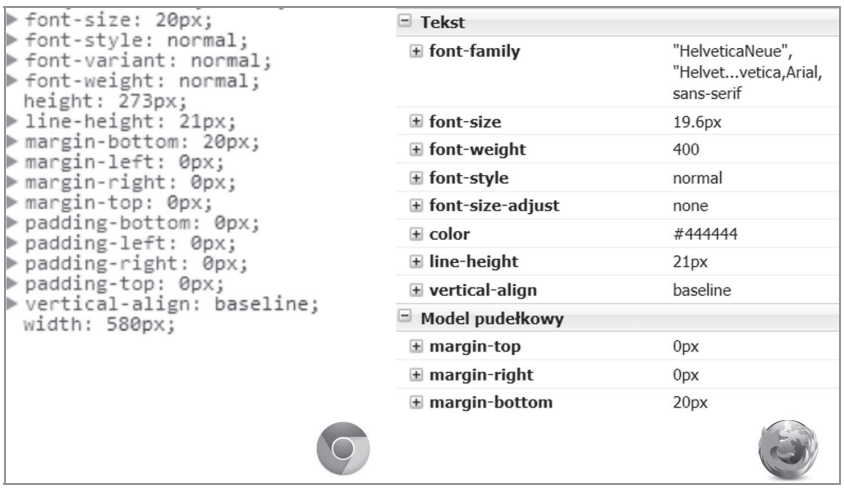



## Dziwne traktowanie jednostki em przez przeglądarki

Przeglądarki w niektórych przypadkach różnie traktują wartości wyrażone w jednostce `em`. W przykładzie z poprzedniego podrozdziału wartość `1.429em` przy podstawowym rozmiarze pisma `14px` zostanie zamieniona na `20px` przez wszystkie przeglądarki (Chrome, Opera, Safari i Firefox).

Ale jeśli usuniemy dwie ostatnie cyfry w celu zaokrąglenia liczby do `1.4em`, to sytuacja się nieco zmieni. W Firefoksie i Operze zostanie zastosowana wartość `19.6px`, a w przeglądarkach opartych na mechanizmie Webkit (Chrome i Safari) zostanie zastosowane zaokrąglenie do `20px`.

Można sprawdzić, jak przeglądarki zamieniają wartości wyrażone w jednostce `em` na piksele w narzędziach programistycznych w okienku *Computed* (wartości obliczone).



## Procenty

Procent to jednostka względna o podobnych właściwościach jak `em`. Podczas gdy `em` odnosi się do rozmiaru pisma, procent odnosi się do długości elementu nadrzędnego, niezależnie od zastosowanej do jej określenia jednostki. Jeśli na przykład element nadrzędny ma wysokość `100px`, to ustawienie szerokości na `100%` jego elementu potomnego jest równoznaczne z ustawieniem jej na `100px`, `50%` oznacza 50 pikseli itd.

W naszym projekcie procentów będziemy używać do określania wymiarów pól elementów, co będzie przydatne przy wyświetlaniu strony na mniejszych ekranach.

# Ustawianie rodziny fontów

W Skeletonie domyślne kroje pisma dokumentu to Helvetica Neue i Helvetica. Jeśli fonty te są niedostępne, zostanie zastosowany krój Arial albo domyślny krój bezszeryfowy, jaki będzie dostępny w komputerze użytkownika.

Definicje rodzin fontów znajdują się w pliku *base.css*:

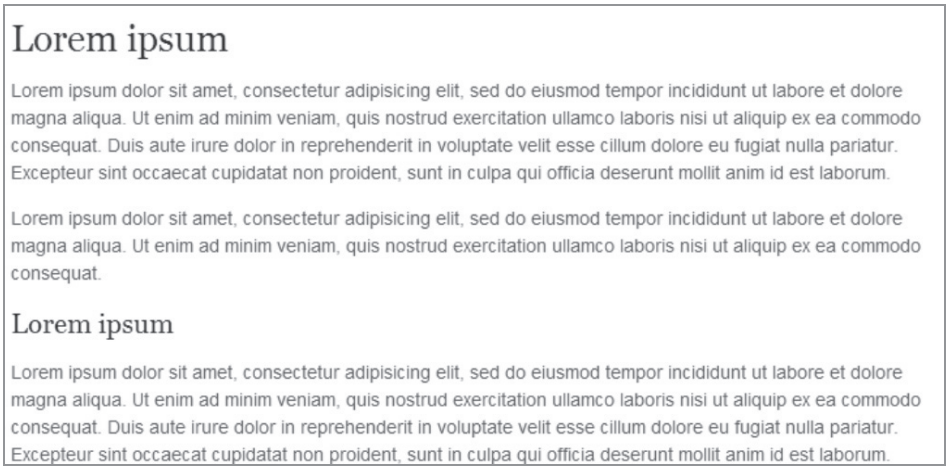
```
body {
    background: #fff; font: 14px/21px "HelveticaNeue",
    "Helvetica Neue", Helvetica, Arial, sans-serif;
    color: #444;
    -webkit-font-smoothing: antialiased;
    -webkit-text-size-adjust: 100%;
}
```

W nagłówkach (`h1`, `h2`, `h3` itd.) ustawione są kroje szeryfowe Georgia i Times New Roman:

```
h1, h2, h3, h4, h5, h6 {
    color: #181818; font-family: "Georgia", "Times New Roman", serif;
    font-weight: normal;
}
```

Kroje te bardzo dobrze nadają się do prezentowania akapitów (pierwszy rysunek na następnej stronie).

Na naszej stronie jednak się nie sprawdzą, bo my mamy bardzo mało tekstu do wyświetlenia (drugi rysunek na następnej stronie).





Dlatego w nagłówkach ustawimy taki sam krój pisma jak w pozostałej części strony, aby zapewnić jednolitość stylu.

## Czas na działanie — ustawianie rodziny fontów nagłówków

Otwórz plik *styles.css* i wpisz poniższą regułę pod deklaracją własności `box-sizing` opisaną w części „Czas na działanie — definiowanie własności box-sizing":

```
h1, h2, h3, h4, h5, h6 {
    font-family: "HelveticaNeue", "Helvetica Neue", Helvetica, Arial,
    sans-serif;
    font-weight: bold;
}
```

### Co uzyskaliśmy?

Ustawiliśmy krój pisma nagłówków na taki sam jak w pozostałej części dokumentu oraz dodatkowo ustawiliśmy własność `font-weight` na `bold`, aby pogrubić te nagłówki.

> Nie istnieje żadna żelazna reguła pozwalająca określić, jakie kroje pisma do siebie pasują. Ich dobór jest sztuką. Ale można postępować według kilku prostych wskazówek, które umożliwią osiągnięcie przynajmniej znośnego efektu. Parę takich porad opisał Ian Yates w serwisie Webdesigntuts+ (*http://webdesign.tutsplus.com/articles/a-beginners-guide-to-pairing-fonts--webdesign-5706*).

# Formatowanie nagłówka

Teraz sformatujemy styl poszczególnych sekcji strony. Jej nagłówek znajduje się w elemencie HTML5 `header` przypisanym do klasy `header`. W elemencie tym znajduje się dodatkowo element `div` przypisany do klasy `logo` zawierający logo strony.

## Czas na działanie — definiowanie stylów nagłówka

Aby zdefiniować styl nagłówka, wykonaj następujące czynności:

1. W pliku *styles.css* wpisz poniższą regułę CSS. Zawiera ona deklaracje ustawiające kolor tła nagłówka oraz jego dopełnienie, obramowanie, cień (definiowany za pomocą własności CSS3 `box-shadow`) i dolny margines odsuwający nagłówek od znajdującej się pod nim sekcji:

```
.header {
    padding: 22px 0;
    background-color: #3a3f43;
    margin-bottom: 14px;
    box-shadow: 0 1px 3px 0 rgba(0,0,0,0.3);
    border-bottom: 1px solid #181f25;
}
```

2. Następnie zdefiniujemy styl kontenera logo. W poniższej regule została zastosowana własność CSS3 `border-radius`, która służy do zaokrąglania rogów elementów:

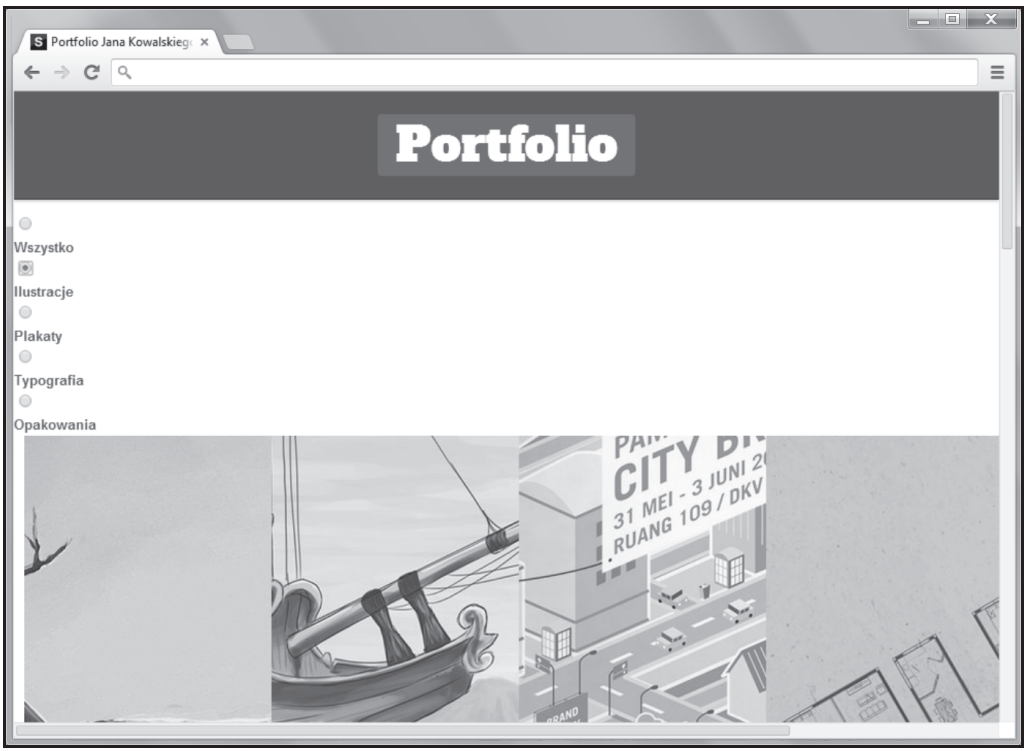
```
.logo {
    text-align: center;
    border-radius: 3px;
    background-color: #515558;
    width: 250px;
    padding: 5px 0;
    margin: 0 auto;
}
```

3. Teraz kolej na styl samego logo, które w przypadku naszej strony jest po prostu tekstem. Do jego prezentacji zastosujemy font Alfa Slab One, który dodaliśmy z serwisu Google Fonts w rozdziale 2.:

```
.logo h1 {
    color: #fff;
    font-weight: normal;
    font-family: "Alfa Slab One", Arial, sans-serif;
    margin-bottom: 0;
}
```

### Co uzyskaliśmy?

Zdefiniowaliśmy style dotyczące nagłówka. Ustawiliśmy mu kolor tła, cień oraz style pola (dopełnienie, obramowanie i margines). Ponadto w logo zastosowaliśmy krój pisma Alfa Slab One z serwisu Google Fonts. Na poniższym zrzucie ekranu widać, jak aktualnie wygląda nasza strona:




# Selektory CSS

W tym projekcie będziemy używać selektorów CSS do wybierania elementów z określonej struktury elementów. Użyjemy więc selektorów dziecka, przylegającego elementu siostrzanego oraz ogólnie elementu siostrzanego. Przyjrzyjmy się im wszystkim po kolei.

## Selektor dziecka

W CSS można wybierać elementy, które są zagnieżdżone w określonych innych elementach (rodzicach). Zapewne wiesz, jak się wybiera element potomny w CSS. Wystarczy określić element nadrzędny (wpisując jego klasę, identyfikator albo nazwę), a następnie wyznaczyć element potomny:

```
.parent p {
    background-color: tomato;
}
```

Powyższa reguła CSS dotyczy wszystkich elementów `p` zagnieżdżonych w elemencie należącym do klasy `parent`.

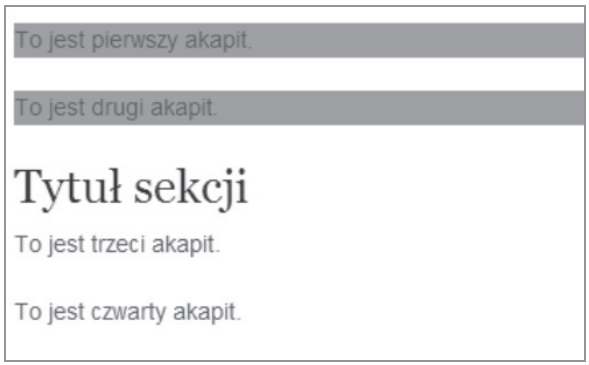
Ale czasami trzeba odwołać się tylko do potomka pierwszego stopnia, tzn. sprawić, aby elementy będące wnukami danego elementu nie zostały już wybrane. W takim przypadku należy użyć kombinatora `>`, który ogranicza wybór elementów do bezpośrednich potomków rodzica:

```
.parent > p {
    background-color: tomato;
}
```

W poniższej strukturze HTML powyższa reguła zmieni kolor tła tylko pierwszego i drugiego akapitu:

```
<div class="parent">
    <p> To jest pierwszy akapit. </p>
    <p> To jest drugi akapit. </p>
    <section>
        <h3>Tytuł sekcji</h3>
        <p> To jest trzeci akapit. </p>
        <p> To jest czwarty akapit. </p>
    </section>
</div>
```

Wynik jest następujący:




## Selektor przylegającego elementu siostrzanego

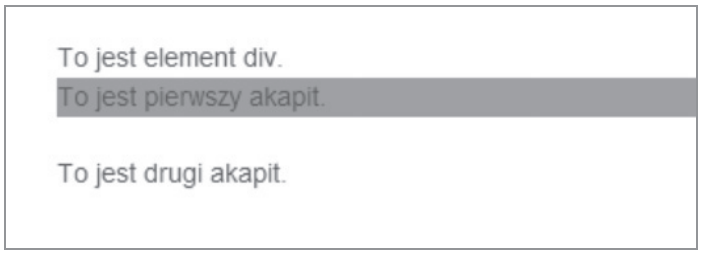
Selektor przylegającego elementu siostrzanego definiuje się przy użyciu kombinatora `+`. Służy on do wybierania elementu, który znajduje się bezpośrednio za określonym elementem. Dotyczy to na przykład sytuacji, gdy w kodzie strony znajduje się element `div`, a bezpośrednio za nim jest `p`:

```
<div>To jest element div.</div>
<p>To jest pierwszy akapit.</p>
<p>To jest drugi akapit.</p>
```

Naszym celem jest element `p` znajdujący się bezpośrednio za elementem `div`. Chcemy mu zdefiniować pomidorowy kolor tła:

```
div + p {
    background-color: tomato;
}
```

Efekt zastosowania tej reguły do powyższego kodu HTML jest następujący:

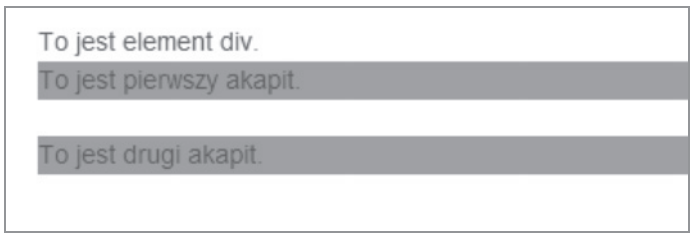



## Ogólny selektor elementu siostrzanego

Ogólny selektor elementu siostrzanego jest nowym typem selektora wprowadzonym w CSS3. Do jego deklarowania używa się tyldy (~):

```
div ~ p {
    background-color: tomato;
}
```

Jego działanie jest podobne do działania selektora przylegającego elementu siostrzanego. Różnica polega na tym, że wybierany jest nie tylko pierwszy element, ale wybierane są wszystkie elementy siostrzane znajdujące się za wyznaczonym. Jeśli więc będziemy mieli kod HTML o strukturze takiej samej jak w przykładzie z selektorem przylegającego elementu siostrzanego, kolor tła zostanie zastosowany do wszystkich akapitów, jak widać na poniższym zrzucie ekranu:



# Pseudoklasy CSS3

Użyjemy też kilku pseudoklas CSS3. Pseudoklasa to klasa służąca do wybierania elementu znajdującego się w określonym stanie. Na przykład pseudoklasa `:hover` stosuje reguły CSS do elementu, gdy znajdzie się nad nim kursor.

W tym projekcie będziemy używać następujących pseudoklas:

- `:checked`
- `:nth-child`

Zobaczmy.

## Pseudoklasa CSS3 :checked

W CSS3 wprowadzono pseudoklasę o nazwie `:checked`. Umożliwia ona wybieranie elementów HTML, pól wyboru lub przycisków radiowych, które są zaznaczone przez użytkownika. Poniższy selektor wybiera przycisk radiowy o identyfikatorze `posters`, gdy ten zostanie zaznaczony:

```
#posters:checked {
    /* deklaracje stylów */
}
```

Pseudoklasy `:checked` można używać do wybierania zaznaczonego przycisku radiowego, który jest używany jako element nawigacji po stronie. Podobnie jak w tradycyjnej nawigacji zbudowanej przy użyciu elementów `a`, używamy też pseudoklasy `:hover` w celu zdefiniowania stylów dla elementu, gdy znajduje się nad nim kursor.

## Pseudoklasa CSS3 :nth-child

W CSS wprowadzono także nową pseudoklasę o nazwie `:nth-child`. Za jej pomocą można wybierać elementy w takiej kolejności, w jakiej znajdują się w kodzie źródłowym. Pseudoklasa ta wymaga podania argumentu, który może być liczbą lub słowem kluczowym (`odd` albo `even`).

Na przykład poniższa reguła zmieni kolor tła trzeciego elementu `li`:

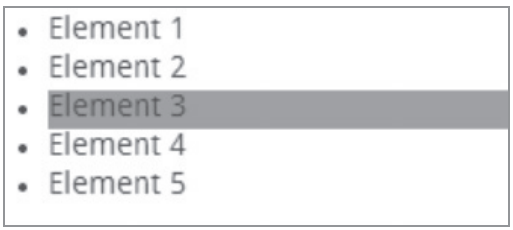
```
li:nth-child(3) {
    background-color: tomato;
}
```

W poniższej strukturze HTML powyższa reguła zmieni kolor tła w środkowym elemencie `li`:

```
<ul>
    <li>Element 1</li>
    <li>Element 2</li>
    <li>Element 3</li>
```

```
        <li>Element 4</li>
        <li>Element 5</li>
    </ul>
```

Efekt użycia tego kodu widać na poniższym zrzucie ekranu:




Można także zamiast liczby wpisać słowo kluczowe `even` albo `odd`. Poniższa reguła zmieni kolor tła w elementach `li` o nieparzystych numerach (pierwszym, trzecim, piątym itd.):

```
li:nth-child(odd) {
    background-color: tomato;
}
```

Pseudoklasie `:nth-child` można też przekazywać wzory wyboru elementów w bardziej wymyślnych sekwencjach:

```
li:nth-child(2n+2) {
    background-color: tomato;
}
```

Litera `n` jest w tym przypadku zmienną, która przyjmuje wartości 0, 1, 2, 3 itd. W związku z tym wzór `2n+2` oznacza wybór drugiego, czwartego, ósmego, dziesiątego itd. elementu `li`.

> Więcej informacji na temat zasady działania pseudoklasy `:nth-child` można znaleźć w artykule Chrisa Coyiera w serwisie CSS Tricks (*http://css-tricks.com/how-nth-child-works/*). Coyer utworzył też poręczne narzędzie do testowania wzoru pseudoklasy `:nth-child` (*http://css-tricks.com/examples/nth-child-tester/*).

# Style miniatur i podpisów

Po sformatowaniu stylu nagłówka można przejść do stylizowania i rozmieszczenia obrazów portfolio. Mamy 12 miniatur, z których każda znajduje się w elemencie HTML5 `figure` i ma podpis zawierający opis miniatury umieszczony w elemencie `figcaption`.

## Czas na działanie — definiowanie stylów miniatur i podpisów

Aby dodać style dla miniatur i podpisów, wykonaj następujące czynności:

1. Otwórz plik *styles.css*. Zaczniemy od przesunięcia portfolio nieco w dół za pomocą marginesu:

```
.portfolio {
    margin-top: 20px;
}
```

2. Obrazy będą wyświetlane w czterech kolumnach. Szerokość każdej z nich będzie wynosić 240 pikseli, bo taką wartość otrzymaliśmy z działania 960 / 4 = 240. Aby obrazy mieściły się na stronie, musimy usunąć lewy i prawy margines, które zostały odziedziczone z klasy `columns` systemu Skeleton:

```
.portfolio .four.columns {
    width: 240px;
    margin-right: 0;
    margin-left: 0;
}
```

3. Następnie ustawimy pozycjonowanie elementu `figure` na względne (`relative`), aby położenie znajdujących się w nim elementów, np. `img` i `figcaption`, było wyznaczane względem niego. Ponadto ustawimy własność `overflow` elementu `figure` na `hidden`:

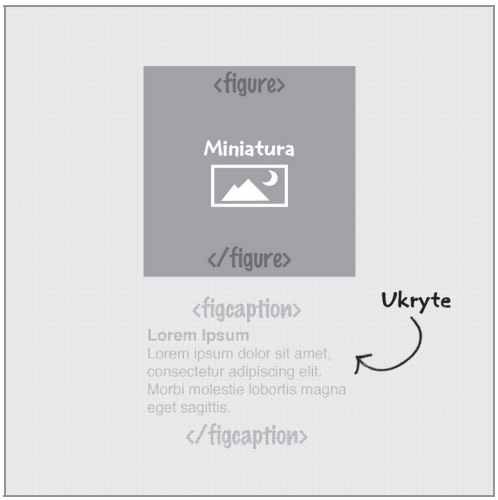
```
.portfolio > figure {
    position: relative;
    overflow: hidden;
}
```

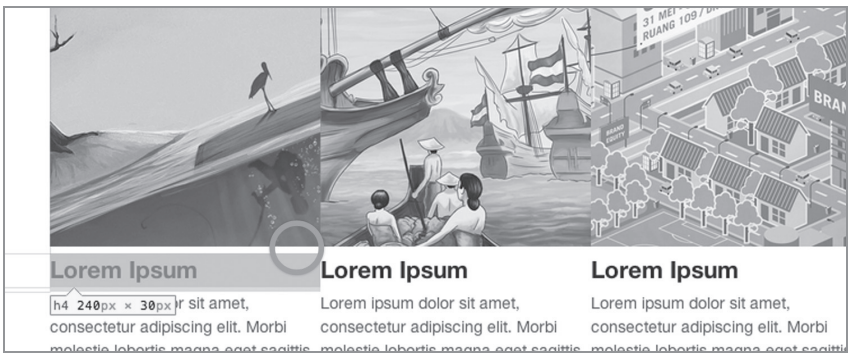
   Ustawieniem własności `overflow` elementu `figure` na `hidden` ukrywamy element wychodzący poza granice elementu `figure`. W naszym przykładzie wykorzystamy to do ukrywania elementu `figcaption`, jak pokazałem na pierwszym zrzucie ekranu przedstawionym na następnej stronie.

4. Własność `max-width` obrazu ustawimy na `100%`, dzięki czemu obraz będzie rozmiarem pasował do zawierającego go elementu, niezależnie od jego rozmiaru:

```
.portoflio > figure img {
    max-width: 100%;
}
```

5. Ponadto jeśli uważnie przyjrzysz się miniaturze, dostrzeżesz niewielki odstęp znajdujący się za elementem `img`, który jest typowy dla elementów śródliniowych i jest związany z domyślnym wyrównaniem elementu w pionie (*http://www.impressivewebs.com/difference-block-inline-css/*) (patrz drugi rysunek na następnej stronie).




Jednym ze sposobów na usunięcie tego odstępu jest ustawienie własności `display` na `block`. Dlatego też dodajemy deklarację `display: block` do reguły dla elementów `img`:

```
.portfolio > figure img {
    max-width: 100%;
    display: block;
}
```

Można też pozbyć się tej pustej przestrzeni poprzez ustawienie własności `vertical-align` na `top`.

6. Następnie zdefiniujemy style dla podpisów miniatur, które są reprezentowane przez element HTML5 `figcaption`. Najpierw ustawimy własność `position` na `absolute`:

```
.portfolio figcaption {
    position: absolute;
}
```

Spowoduje to zmianę wysokości elementu nadrzędnego i zniknięcie podpisów wywołane ustawieniem ich pozycji na miejsce poza obszarem elementu `figure`.

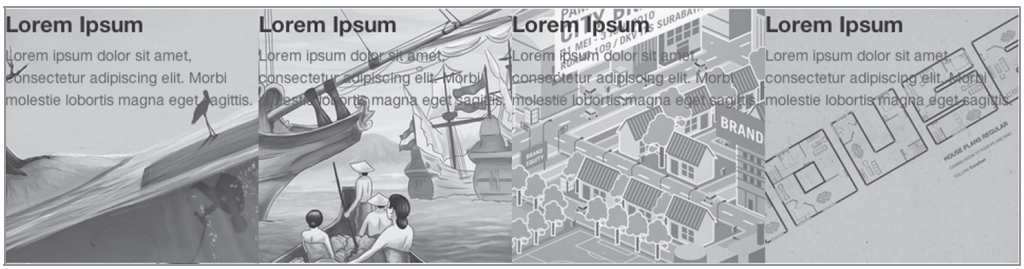
Następnie ustawiamy wysokość i szerokość podpisu na `100%`, aby jego wymiary zawsze pokrywały się z wymiarami rodzica, którym w tym przypadku jest element `figure`:

```
.portfolio figcaption {
    position: absolute;
    width: 100%;
    height: 100%;
}
```

7. Ustawiając własność `position` elementu `figure` na `absolute`, możemy go przesunąć w dowolnym kierunku, nie przejmując się sąsiednimi elementami. W tym przypadku ustawimy własności `left` i `top` na `0`, jak widać na poniższym listingu:

```
.portfolio figcaption {
    position: absolute;
    width: 100%;
    height: 100%;

    left: 0;
    top: 0;
}
```

Dzięki temu, że elementowi `figure` własność `position` została ustawiona na `relative`, położenie znajdującego się w nim podpisu jest ustalane względem właśnie tego elementu `figure` (będącego rodzicem podpisu). To sprawia, że podpis nakłada się na miniaturę, jak na poniższym zrzucie ekranu:




8. Następnie ustawimy kolor tła podpisu przy użyciu techniki RGBA (ang. *red*, *green*, *blue*, *alpha* — czerwony, zielony, niebieski, alfa). Wartość kanału każdego koloru oznacza się liczbą z przedziału od `0` do `255`.

Jeśli na przykład wpisze się `0` dla każdego kanału (`rgba(0,0,0,1)`), to otrzyma się kolor czarny, który w zapisie szesnastkowym ma postać `#000000`. Analogicznie same wartości `255` oznaczają kolor biały (`#ffffff` w notacji szesnastkowej).

Technika RGBA umożliwia także dostosowanie poziomu przezroczystości koloru za pomocą kanału alfa. Zakres wartości tego kanału wynosi od `0` do `1`. Wartość `0` oznacza `0%`, a `1` to `100%`. W takim razie `0.5` jest równoznaczne z `50%`.

W poniższej regule ustawiamy czarny kolor tła i przezroczystość na poziomie 80%:

```
.portfolio figcaption {
    position: absolute;

    width: 100%;
    height: 100%;

    left: 0;
    top: 0;

    background-color: rgba(58,63,67,.8);
}
```

9. Dodamy też dopełnienie, aby odsunąć nieco tekst podpisu od krawędzi elementu kontenera. W poniższej regule dopełnienie zostało ustawione na 10%:

```
.portfolio figcaption {
    position: absolute;

    width: 100%;
    height: 100%;

    left: 0;
    top: 0;

    background-color: rgba(58,63,67,.8);

    padding: 10%;
}
```

Dzięki temu, że wcześniej ustawiliśmy własność `box-sizing` na `border-box`, rozmiar podpisu (wysokość i szerokość) pozostanie na poziomie 100%, niezależnie od dopełnienia. Inaczej mówiąc: podpis nadal będzie miał wymiary swojego rodzica, które wynoszą 240 × 240 pikseli.

10. Jako że tło jest ciemne, tekstowi musimy nadać jasny kolor. W związku z tym zmienimy kolor tekstu podpisu na biały (`#fff`):

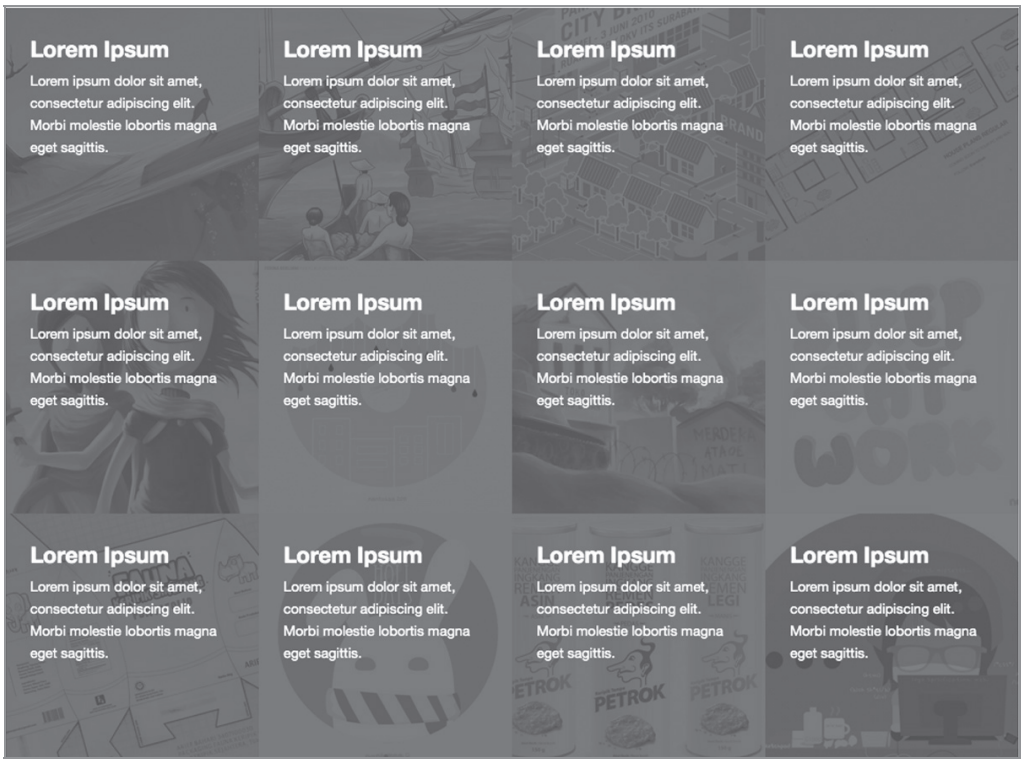
```
.portfolio figcaption h4 {n
    color: #fff;
}
.portfolio figcaption p {
    color: #fff;
}
```

11. Może jest to kwestia gustu, ale mnie się wydaje, że tekst w podpisie jest za duży. Dlatego zmniejszę go o 1 piksel w stosunku do rozmiaru bazowego. Podstawowy rozmiar pisma wynosi 14 pikseli, więc zmienię go na 13, czyli `0.929em`:

```
.portfolio figcaption p {
    color: #fff;
    font-size: 0.929em;
}
```

### Co uzyskaliśmy?

Właśnie dodaliśmy reguły stylistyczne dotyczące miniatury i jej podpisu. Na tym etapie nasze portfolio wygląda jak na poniższym zrzucie ekranu:




# Przekształcenia dwuwymiarowe CSS

Ostanie lata przyniosły sporo nowości w CSS3, wśród których znajdują się m.in. przekształcenia CSS3 (*http://www.w3.org/TR/css3-transforms/*). Przy użyciu przekształceń można przesuwać, obracać, zniekształcać oraz zwiększać i zmniejszać elementy HTML.

## Funkcja translate()

Funkcja `translate()` w przekształceniach CSS3 służy do przesuwania elementów względem ich pierwotnego położenia. Można jej używać na trzy sposoby:

- Aby przesunąć element w poziomie:

```
transform: translateX(value);
```

- Aby przesunąć element w pionie:

```
transform: translateY(value);
```

- Można też użyć skróconej składni umożliwiającej dokonanie przesunięcia zarówno w poziomie, jak i w pionie:

```
transform: translate(x-value,y-value);
```

Położenie elementu określa się za pomocą wartości *x* i *y*, które reprezentują odpowiednio współrzędną na osi *x* i współrzędną na osi *y* w układzie współrzędnych podobnym do kartezjańskiego (*http://pl.wikipedia.org/wiki/Układ_współrzędnych_kartezjańskich*).

Ponieważ jednak strony internetowe czyta się od góry, współrzędna *y* ma odwrotne znaczenie, tzn. ujemna wartość oznacza przesunięcie w górę, a dodatnia — w dół.

Powiedzmy, że chcemy przesunąć element o 100 pikseli w prawo. Takie przesunięcie można zapisać w następujący sposób:

```
transform: translateX(100px)
```

Można też użyć skróconej notacji następująco:

```
transform: translate(100px,0)
```

Podobnie, aby przesunąć element o 100 pikseli w górę, można to zapisać tak:

```
transform: translateY(-100px)
```

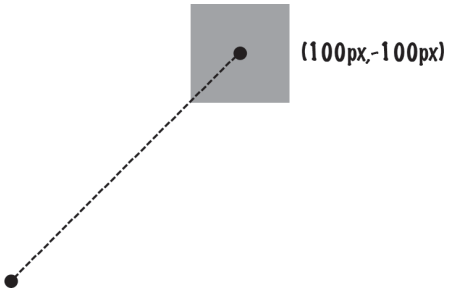
Ale można też użyć notacji skróconej:

```
transform: translate(0,-100px)
```

Jeśli natomiast zechcemy przesunąć element na ukos, zdefiniujemy zarówno przesunięcie poziome, jak i pionowe:

```
transform: translate(100px,-100px)
```

Ta deklaracja przesunie element do góry i w prawo, jak na poniższym rysunku:




## Przedrostki firmowe

Funkcję `translate()` obsługują przeglądarki Chrome 4, Safari 3.1, Firefox 3.5, Internet Explorer 10 oraz Opera 10.5, chociaż żeby zadziałała, trzeba dodać odpowiednie przedrostki firmowe. W związku z tym kompletna reguła przekształcenie wygląda tak:

```
-webkit-transform: translate(x,y); /* Webkit (np. Chrome i Safari) */
-moz-transform: translate(x,y);    /* Mozilla Firefox */
-ms-transform: translate(x,y);     /* Internet Explorer */
-o-transform: translate(x,y);      /* Opera */
transform: translate(x,y);         /* standardowa składnia zalecana przez W3C */
```

Przeglądarki rozpoznają deklaracje z własnymi przedrostkami, a pozostałe ignorują. Na przykład przeglądarki oparte na mechanizmie Webkit, np. Chrome i Safari, używają przedrostka `-webkit-`. Gdy specyfikacja przekształceń zostanie ukończona i w pełni zaimplementowana w przeglądarkach, używana będzie wymieniona na końcu standardowa deklaracja zalecana przez W3C (*http://w3.org/*).

Dlatego właśnie powinno się stosować wersje deklaracji z przedrostkami firmowymi: zwiększają one zakres obsługiwanych przeglądarek.

W powyższej regule użyto aż pięciu deklaracji, z których każda zasadniczo służy do tego samego celu. Jak widać, wpisywanie przedrostków firmowych to bardzo żmudne zadanie, które lepiej wykonać przy użyciu specjalnego narzędzia.

Jednym z takich generatorów przedrostków firmowych jest Prefixr (*http://prefixmycss.com/*). Jeśli używasz programu Sublime Text 2, to możesz pobrać specjalny pakiet umożliwiający jego uruchomienie bezpośrednio w edytorze.

Istnieje też biblioteka JavaScript o nazwie Prefix Free (*http://leaverou.github.com/prefixfree/*), która dodaje przedrostki w locie, dzięki czemu nie trzeba ich w ogóle wpisywać.

# Przejścia CSS3

Kolejnym znakomitym nowym dodatkiem w CSS3 są przejścia umożliwiające stopniowe — w odróżnieniu od błyskawicznego — zmienianie jednej reguły CSS na inną w określonym czasie. Do definiowania przejść CSS3 służy następująca składnia (pokazane są też wersje z przedrostkami):

```
-webkit-transition: property duration timing-function delay; /* Webkit (np. Chrome / & Safari) */
-moz-transition: property duration timing-function delay;    /* Mozilla Firefox */
-o-transition: property duration timing-function delay;      /* Opera */
transition: property duration timing-function delay;         /* standardowa / składnia zalecana przez W3C */
```

Aktualnie przejścia CSS3 są obsługiwane przez przeglądarki Chrome 4, Firefox 4, Safari 3.1, Opera 10.5 oraz Internet Explorer 10.0 (*http://caniuse.com/#feat=css-transitions*).

> Przeglądarka Internet Explorer nie obsługuje przejść CSS3 i dlatego na powyższym listingu nie ma wersji deklaracji z przedrostkiem `-ms-`, bo Internet Explorer 10 obsługuje je bez przedrostka (*http://msdn.microsoft.com/en-us/library/ie/hh673535(v=vs.85).aspx*).

## Wartości przejść CSS3

W przedstawionych przykładach użyte zostały po cztery wartości: `property`, `transition-duration`, `timing-function` oraz `delay`. Zobaczmy, do czego służą:

| Wartość | Opis |
|---|---|
| `property` | Ta wartość wskazuje własność CSS, do której ma zostać zastosowany efekt przejścia. Może to być `width`, `height`, `color`, `background` itd. |
| | Jeśli nie zostanie wprost zdefiniowana, to zostanie zastosowana wartość domyślna `all`, stosująca przejście do wszystkich własności. |
| `transition-duration` | Wartość określająca czas trwania przejścia. Można ją wyrażać w milisekundach lub sekundach, np. `200ms` albo `0.2s`. |
| `timing-function` | Wartość określająca sposób wykonywania przejścia w czasie. Dostępnych jest pięć gotowych ustawień: `ease`, `easein`, `ease-out`, `ease-in-out` oraz `linear`. Ich działanie można obejrzeć na stronie *http://www.css3.info/preview/css3-transitions/*. |
| `delay` | Wartość określająca, po upływie jakiego czasu ma się rozpocząć wykonywanie przejścia. |

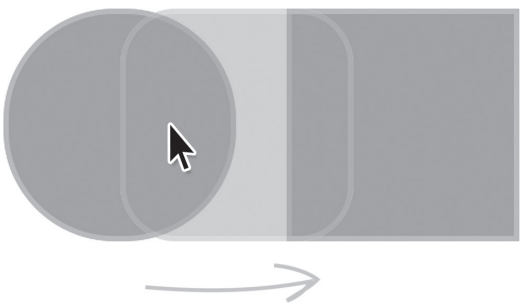
Poniższa przykładowa reguła tworzy koło z elementu `div`, które potem dzięki przejściom zmienimy stopniowo w prostokąt, gdy zostanie nad nim ustawiony kursor:

```
div {
    width: 200px;
    height: 200px;
    border-radius: 100px;
    border: 5px solid orange;
    background-color: tomato;
}
div:hover {
    border-radius: 0;
}
```

Jak wspominałem wcześniej, teraz przejście zostanie wykonane natychmiast, bo nie użyliśmy żadnej własności przejść CSS3. Zrobimy to teraz. Ustawimy efekt przejścia dla własności `border-radius` o czasie trwania `200ms`:

```
div {
    /* pozostałe deklaracje są takie same jak powyżej */
    -webkit-transition: border-radius 200ms;
    -moz-transition: border-radius 200ms;
    -o-transition: border-radius 200ms;
    transition: border-radius 200ms;
}
```

Teraz zmiana będzie się dokonywała stopniowo, jak pokazuje poniższy rysunek, i będzie bardziej atrakcyjna:



## Czas na działanie — zmiana obrazu po najechaniu kursorem

Aby zmienić obraz na coś innego po najechaniu na niego kursorem, wykonaj następujące czynności:

1. Otwórz plik *style.css*. Przypomnę, że podpisy obrazów są reprezentowane przez elementy HTML5 `figcaption`. Wcześniej elementom tym zdefiniowaliśmy już trochę ustawień, np. pozycjonowanie, szerokość i wysokość, kolor tła, kolor pisma oraz dopełnienie.

Teraz dodamy reguły dotyczące efektu hover. Ogólnie chodzi o to, że po najechaniu kursorem na miniaturę podpis stopniowo wysunie się z wybranej strony i zakryje obraz.

W poniższej regule najpierw przesuwamy podpis w prawo za pomocą przekształcenia CSS3:

```
.portoflio figcaption {
    position: absolute;
    left: 0;
    top: 0;

    width: 100%;
    height: 100%;
    padding: 10%;

    background-color: rgba(58,63,67,.8);

    -webkit-transform: translateX(100%);
    -moz-transform: translateX(100%);
    -ms-transform: translateX(100%);
    -o-transform: translateX(100%);
    transform: translateX(100%);
}
```

2. Teraz dodamy efekt przejścia CSS3. W poniższej regule ustawiamy przejście dla wszystkich własności na czas 350ms:

```
.portoflio figcaption {
    position: absolute;
    left: 0;
    top: 0;

    width: 100%;
    height: 100%;
    padding: 10%;

    background-color: rgba(58,63,67,.8);

    -webkit-transform: translateX(100%);
    -moz-transform: translateX(100%);
    -ms-transform: translateX(100%);
    -o-transform: translateX(100%);
    transform: translateX(100%);

    -webkit-transition: all 350ms;
    -moz-transition: all 350ms;
    -o-transition: all 350ms;
    transition: all 350ms;
}
```

3. Na koniec dodamy definicję stanu `hover`. W stanie tym podpis wraca do swojego pierwotnego położenia dzięki ustawieniu funkcji `translateX` na `0`:

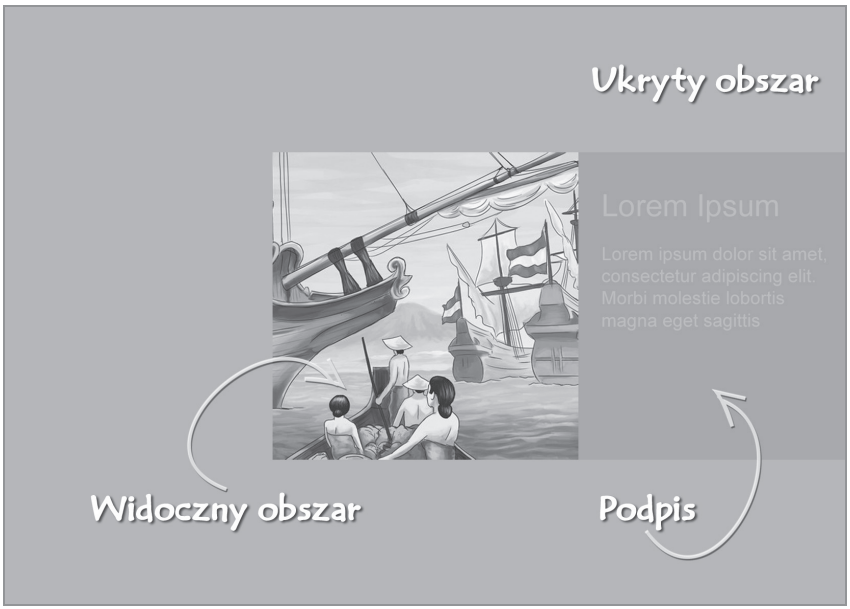
```
.container figure:hover figcaption {
    -webkit-transform: translateX(0);
    -moz-transform: translateX(0);
    -ms-transform: translateX(0);
    -o-transform: translateX(0);
    transform: translateX(0);
}
```

## Co uzyskaliśmy?

Kod ten wykonuje kilka działań. Po pierwsze przesuwa podpis za pomocą własności `transform` o `100%` szerokości rodzica w prawo:

```
-webkit-transform: translateX(100%);
-moz-transform: translateX(100%);
-ms-transform: translateX(100%);
-o-transform: translateX(100%);
transform: translateX(100%);
```

Teraz podpis jest niewidoczny, bo ustawiliśmy, że elementy wychodzące poza granice elementu `figure` mają być niewidoczne. Ilustruje to poniższy zrzut ekranu:

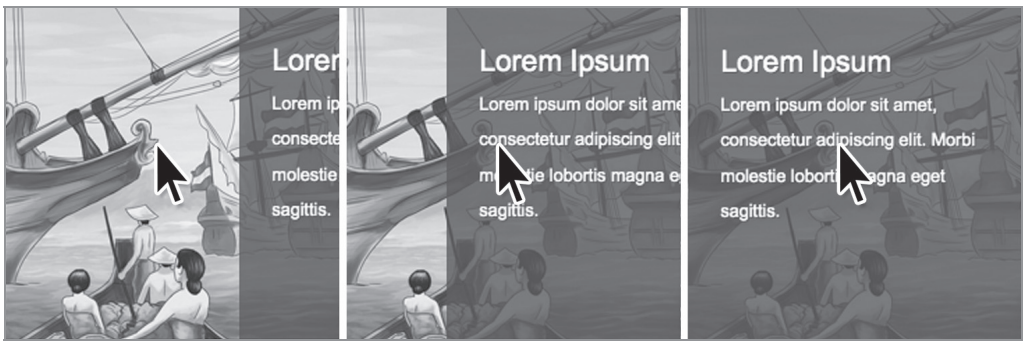


Następnie stosujemy efekt przejścia do wszystkich własności podpisu:

```
-webkit-transition: all 350ms;
-moz-transition: all 350ms;
-o-transition: all 350ms;
transition: all 350ms;
```

W końcu zdefiniowaliśmy style elementu w stanie `hover`. Chodzi nam o to, by podpis obrazu po najechaniu na niego kursorem wysuwał się z prawej strony i wracał do swojego pierwotnego położenia. Dlatego właśnie ustawiliśmy współrzędną przesunięcia na `0%`.

Po wpisaniu do arkusza stylów wszystkich opisanych w tej części reguł efekt hover powinien działać:




# Filtrująca nawigacja po stronie

Jak wspominałem, nawigacja na tej stronie będzie sortować portfolio według kategorii. W związku z tym do jej utworzenia użyłem przycisków radiowych z etykietami:

```
<input class="nav-menu" id="all" type="radio" name="filter" checked="checked"/>
<label for="all">Wszystkie</label>
<input class="nav-menu" id="illustrations" type="radio" name="filter"/>
<label for="illustrations">Ilustracje</label>
<input class="nav-menu" id="posters" type="radio" name="filter"/>
<label for="posters">Plakaty</label>
<input class="nav-menu" id="typography" type="radio" name="filter"/>
<label for="typography">Typografia</label>
<input class="nav-menu" id="packaging" type="radio" name="filter"/>
<label for="packaging">Opakowania</label>
```

## Czas na działanie — tworzenie filtra portfolio

Aby utworzyć filtr portfolio, wykonaj następujące czynności:

1. Otwórz plik *style.css*. Nawigacja jest zbudowana na bazie przycisków radiowych przypisanych do klasy `nav-menu`. Najpierw ukryjemy przyciski radiowe:

```
.nav-menu {
    display: none;
}
```

2. Następnie dodamy style formatujące element `label`. W poniższej regule ustawiamy własność `display` na `inline-block`, aby etykiety były wyświetlane w jednym rzędzie:

```
label {
    padding: 5px 10px;
    color: #3a3f43;
    cursor: pointer;
    display: inline-block;
}
```

3. Gdy zostanie wybrany przycisk radiowy, zmieniają się style elementu `label`. W tym przypadku ustawimy ciemniejszy kolor tła, aby pokazać, że menu zostało zaznaczone. To pociąga za sobą konieczność zmiany koloru tekstu na biały.

   Aby odnieść się do elementu `label` znajdującego się bezpośrednio obok zaznaczonego elementu `input`, należy użyć selektora przylegającego elementu siostrzanego:

```
.nav-menu:checked + label {
    color: #fff;
    background-color: #3a3f43;
    border-radius: 3px;
}
```

4. Następnie zdefiniujemy funkcję filtrującą. Zaczniemy od ukrycia miniatury portfolio. W tym celu zdefiniujemy następującą regułę z selektorem `.nav-menu`:

```
.nav-menu,
.portfolio > figure.columns {
    display: none;
}
```

5. Aby uzyskać funkcję filtrującą, trzeba połączyć kilka selektorów CSS. Ale najpierw trzeba zbudować nawigację przy użyciu przycisków radiowych i każdemu z nich przypisać inny identyfikator. Z przyciskami tymi związane są etykiety za pomocą atrybutu `for="id"`. Dzięki temu kliknięcie etykiety powoduje zaznaczenie odpowiadającego jej przycisku radiowego. Do zaznaczonego przycisku można się odnieść dzięki pseudoklasie CSS3 `:checked`. W poniższym kodzie wybieramy zaznaczony element `input`, którego identyfikator to `all`:

```
#all:checked
```

Dołączając selektor przylegającego elementu, odnosimy się do sekcji `portfolio`:

```
#all:checked ~ .portfolio
```

Po wybraniu sekcji `portfolio` możemy wybierać znajdujące się w niej elementy potomne, aby zastosować do nich reguły stylistyczne. W tym przypadku naszym celem jest zdefiniowanie własności `display` o wartości `block` elementom przypisanym do klasy `all`:

```
#all:checked ~ .portfolio .all {
    display: block;
}
```

Dzięki temu kliknięcie elementu `input` o identyfikatorze `all` będzie powodowało wyświetlenie wszystkich miniatur portfolio.

6. Teraz zrobimy to samo dla pozostałych kategorii — `poster`, `illustration`, `typography` oraz `package`:

```
#all:checked ~ .portfolio .all,
#posters:checked ~ .portfolio .poster,
#illustrations:checked ~ .portfolio .illustration,
#typography:checked ~ .portfolio .typography,
#packaging:checked ~ .portfolio .package {
    display: block;
}
```

## Co uzyskaliśmy?

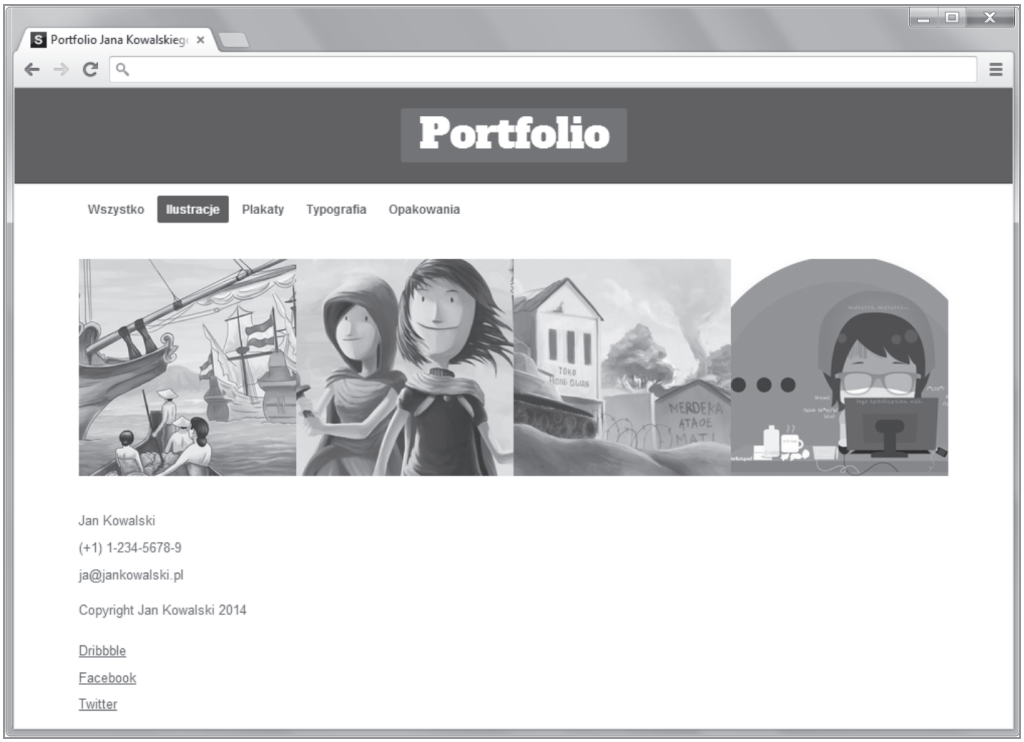
Napisaliśmy style dotyczące nawigacji i zdefiniowaliśmy funkcję filtrowania przy użyciu selektorów CSS.

> Mimo że pseudoklasa `:checked` oficjalnie została dodana dopiero do specyfikacji CSS3, była obsługiwana już przez przeglądarkę Firefox 1. Aby zobaczyć, jak działa, można obejrzeć demo na stronie W3C pod adresem *http://www.w3.org/Style/CSS/Test/CSS3/Selectors/current/html/full/flat/css3-modsel-25.html*.

Teraz powinno być możliwe sortowanie portfolio według kategorii, jak widać na zrzucie ekranu pokazanym na następnej stronie.

# Stopka

W tej części zdefiniujemy style stopki. W stopce znajdują się odnośniki do naszych profili w serwisach społecznościowych oraz informacje kontaktowe: numer telefonu, adres e-mail, a także imię i nazwisko.




## Czas na działanie — formatowanie stopki

Aby sformatować stopkę, wykonaj następujące czynności:

1. Cały czas pracujemy w pliku *style.css*. Stopka naszej strony jest zdefiniowana w elemencie HTML5 `footer` i jest przypisana do klasy `footer`. Zaczniemy od zdefiniowania stylów dekoracyjnych, takich jak marginesu, dopełnienia i obramowania:

```
.footer {
    border-top: 1px solid #ccc;
    margin-top: 28px;
    padding: 28px 0;
}
```

2. Następnie przeniesiemy odnośniki do mediów społecznościowych na lewą stronę:

```
.social {
    float: left;
}
```

3. W sekcji odnośników społecznościowych wygospodarowaliśmy trochę miejsca na formułkę o prawach autorskich. Zmienimy kolor tekstu na szary i odsuniemy ją nieco od dolnej krawędzi strony za pomocą własności `margin-bottom`:

```
.social .copyright {
    color: #ccc;
    margin-bottom: 10px;
    font-size: 1em;
}
```

4. Odnośniki do mediów społecznościowych znajdują się w elementach `li`. Sprawimy, że będą wyświetlane w jednym rzędzie:

```
.social ul li {
    display: inline;
}
```

5. Teraz zdefiniujemy style samych odnośników. Najpierw, aby móc ustawiać ich szerokość i wysokość, należy ustawić własność `display` elementu kotwicy na `inline-block`, a następnie ustawimy szerokość na 42 piksele i wysokość na 36 pikseli:

```
.social ul li a {n
    display: inline-block;
    width: 36px;
    height: 42px;
}
```

6. Zastosujemy też technikę podmiany tekstu odnośnika na obraz, który później dodamy za pomocą własności `background-image`:

```
.social ul li a {
    display: inline-block;
    width: 48px;
    height: 48px;

    /* poniżej znajdują się style dotyczące zastępowania tekstu odnośnika obrazem */
    text-indent: 100%;
    white-space: nowrap;
    overflow: hidden;
}
```

7. W następnej kolejności dodamy do elementu `a` za pomocą własności `background-image` ikony mediów społecznościowych, które połączyliśmy w jeden plik sprite'ów w rozdziale 2.:

```
.social-dribbble a,
.social-facebook a,
.social-twitter a {
    background-image: url('../images/social.png');
    background-repeat: no-repeat;
}
```

8. Kolejną czynnością będzie zmodyfikowanie reguł CSS wygenerowanych podczas łączenia ikon w jeden plik. Reguły te określają położenie obrazu z ikonami:

```
.social-dribbble-hover{
    background-position: 0 0;
    width: 48px;
    height: 48px;
}
.social-dribbble{
    background-position: 0 -58px;
    width: 48px;
    height: 48px;
}
.social-facebook-hover{
    background-position: 0 -116px;
    width: 48px;
    height: 48px;
}
.social-facebook{
    background-position: 0 -174px;
    width: 48px;
    height: 48px;
}
.social-twitter-hover{
    background-position: 0 -232px;
    width: 48px;
    height: 48px;
}
.social-twitter{
    background-position: 0 -290px;
    width: 48px;
    height: 48px;
}
```

Najpierw zmienimy klasę definiującą stan `hover` (`.social-dribbble-hover`) i przypiszemy ją do łączy w postaci ikon:

```
.social-dribbble a:hover {
    background-position: 0 0;
    width: 48px;
    height: 48px;
}
.social-dribbble{
    background-position: 0 -58px;
    width: 48px;
    height: 48px;
}
.social-facebook a:hover{
    background-position: 0 -116px;
    width: 48px;
```

```
    height: 48px;
}
.social-facebook{
    background-position: 0 -174px;
    width: 48px;
    height: 48px;
}
.social-twitter a:hover{
    background-position: 0 -232px;
    width: 48px;
    height: 48px;
}
.social-twitter{
    background-position: 0 -290px;
    width: 48px;
    height: 48px;
}
```

9. Jako że szerokość i wysokość ustawiliśmy wcześniej, teraz możemy usunąć te definicje:

```
.social-dribbble a:hover {
    background-position: 0 0;
}
.social-dribbble{
    background-position: 0 -58px;
}
.social-facebook a:hover{
    background-position: 0 -116px;
}
.social-facebook{
    background-position: 0 -174px;
}
.social-twitter a:hover{
    background-position: 0 -232px;
}
.social-twitter{
    background-position: 0 -290px;
}
```

10. Sekcję kontaktową przesuniemy na prawą stronę:

```
.contact {
    float: right;
}
```

11. Zmienimy kolor tekstu i łączy tekstowych na szary:

```
.contact, .contact a {
    color: #ccc;
}
```

**12.** Następnie za pomocą własności `background-image` dodamy ikony danych kontaktowych, które połączyliśmy w jeden plik w rozdziale 2. Ale te ikony wstawimy do pseudoelementu `:before`.

Pseudoelement `:before` dodaje element przed treścią elementu, do którego został zastosowany. Istnieje też pseudoelement `:after`, który dodaje element za wyznaczonym elementem.

W kodzie HTML można to przedstawić następująco:

```
<div>
    <span> </span> <!-- :before -->
    Treść
    <span> </span> <!-- :after -->
</div>
```

Pseudoelementy nie dodają prawdziwych elementów HTML i dlatego właśnie nazywa się je ***pseudo***elementami. Są one traktowane tak jak elementy, ale w rzeczywistości nimi nie są.

W CSS3 zmieniono trochę składnię pseudoelementów w porównaniu z wcześniejszymi wersjami tej technologii. Obecnie elementy te definiuje się przy użyciu dwóch dwukropków (`::before` i `::after`), aby odróżnić je od pseudoklas, które definiuje się za pomocą jednego dwukropka (`:hover` i `:checked`).

W poniższej regule dodajemy pseudoelement `:before` do elementu `li`, zmieniamy też własność `display` na `inline-block`, aby móc ustawiać szerokość i wysokość:

```
.contact ul li:before {
    content: '';
    display: inline-block;
    width: 24px;
    height: 24px;
    background-image: url('../images/contact.png');
    margin-right: 0.1em;
}
```

**13.** Na koniec dostosowujemy położenie tła zawierającego ikony danych kontaktowych:

```
.contact-name:before {
    background-position: 0 -29px;
}
.contact-phone:before {
    background-position: 0 -63px;
}
.contact-email:before{
    background-position: 0 5px;
}
```

### Co uzyskaliśmy?

Zdefiniowaliśmy style stopki i dodaliśmy sekcje z odnośnikami do profilów społecznościowych i danymi kontaktowymi. Teraz stopka powinna wyglądać tak jak na poniższym zrzucie ekranu:


# Definiowanie stylów dla mniejszych ekranów

Teraz zdefiniujemy style umożliwiające wygodne przeglądanie strony w urządzeniach o mniejszych ekranach. W tym celu użyjemy zapytań medialnych CSS3.

Zanim zaczniemy definiować własne style, skopiujemy wszystkie zapytania medialne z pliku *layout.css* systemu Skeleton do naszego pliku *style.css*. Reguły do skopiowania są pokazane poniżej:

```
/* Urządzenia i przeglądarki o rozmiarze okna mniejszym niż */
@media only screen and (max-width: 959px) {

}
/* Tablet w poziomie do 960 */
@media only screen and (min-width: 768px) and (max-width: 959px) {

}
/* Wszystkie urządzenia przenośne */
@media only screen and (max-width: 767px) {

}
/* Urządzenia przenośne w poziomie do tabletu w pionie */
@media only screen and (min-width: 480px) and (max-width: 767px) {

}
/* Urządzenia przenośne w pionie do urządzenia przenośne w poziomie */
@media only screen and (max-width: 479px) {

}
```

## Czas na działanie — rozmiar obszaru widoku poniżej 960 pikseli

Zdefiniujemy style przeznaczone dla przeglądarek o szerokości okna mniejszej niż 960 pikseli:

1. Definiowane style zostaną umieszczone w poniższym zapytaniu medialnym, które serwuje style przeglądarkom o szerokości mniejszej niż 960 pikseli:

```
@media only screen and (max-width: 959px) {
}
```

2. Jako że definiujemy style dla mniejszego urządzenia, musimy zmienić szerokość kolumn i kontenera na względne jednostki. W tym przypadku szerokość kontenera będzie wynosić `100%`, a szerokość kolumn wyniesie po `25%`, bo kontener jest podzielony na cztery kolumny:

```
.container {
    width: 100%;
}
.portfolio .four.columns {
    width: 25%;
}
```

3. Następnie ukryjemy menu nawigacyjne, aby użytkownicy mogli poruszać się po stronie za pomocą palców:

```
label {
    display: none;
}
```

4. Pod każdym wierszem ustawimy niewielki odstęp za pomocą własności `margin-bottom` dodanej do elementu `figure`:

```
.portfolio .all {
    margin-bottom: 15px;
}
```

5. Jako że ukryliśmy nawigację, musimy gdzieś indziej przesunąć informację o kategorii.

   Przeniesiemy ją na górę obrazu. Możemy do tego celu wykorzystać pseudoelement `:before` oraz pobierać informacje z atrybutu HTML5 `data-*`:

```
.portofolio > figure:before {
    content: attr(data-category);
    font-size: 1em;
    padding: 8px;
    width: 100%;
    color: #fff;
    display: block;
    font-weight: bold;
    text-transform: capitalize;
```

```
    background-color: rgba(42,47,51,0.8);
    position: absolute;
}
```

6. W tym pomniejszonym obszarze widoku nie będziemy ukrywać podpisów obrazów, tylko je wyświetlimy. W związku z tym ustawiamy własność `position` na `relative`, a `translateX` na `0%`, ponadto ustawimy domyślny kolor tła podpisom:

```
.portfolio figcaption {
    position: relative;

    -webkit-transform: translateX(0%);
    -moz-transform: translateX(0%);
    -ms-transform: translateX(0%);
    -o-transform: translateX(0%);
    transform: translateX(0%);

    background-color: #3a3f43;
}
```

7. Za pomocą selektora `nth-child` wybierzemy elementy `figure` o nieparzystym numerze oraz ustawimy im tło na ciemniejsze niż domyślne ustawione w punkcie 6, dzięki czemu podpisy będą się od siebie różnić:

```
.portoflio figure:nth-child(odd) figcaption {
    background-color: #2a2f33;
}
```

8. Na koniec dostosujemy margines w stopce:

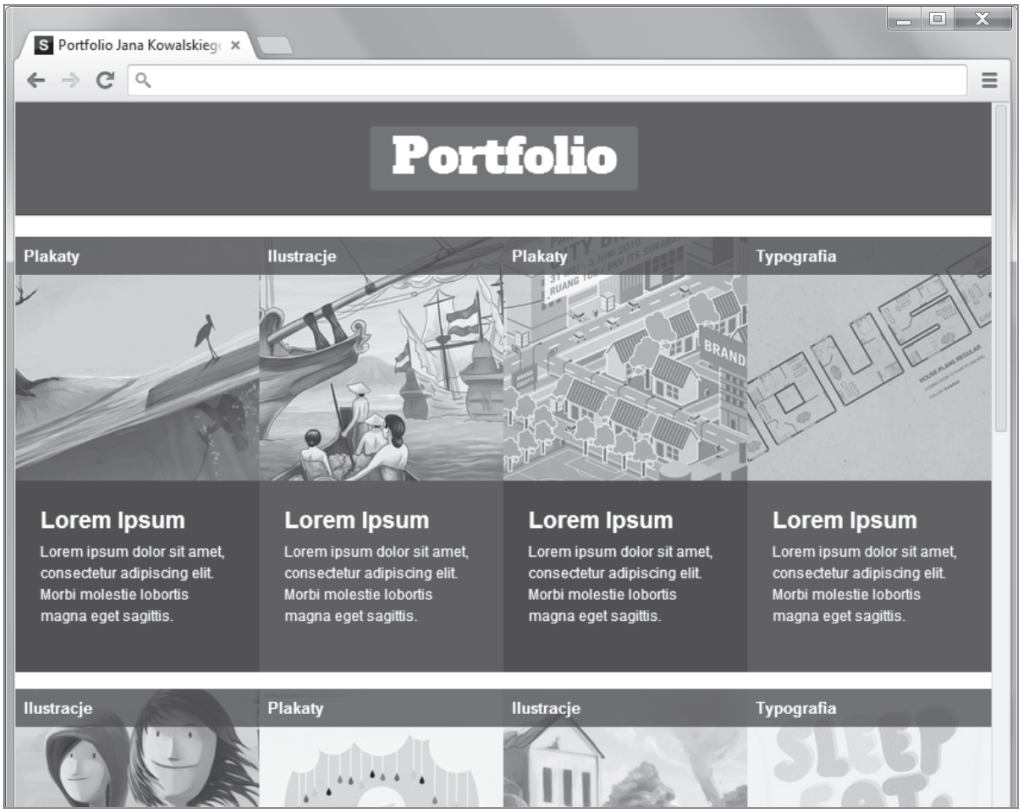
```
.footer {
    border-top: 1px solid #ccc;
    margin-top: 42px;
    padding: 28px;
}
```

### Co uzyskaliśmy?

Dodaliśmy style dla okien o szerokości mniejszej niż 960 pikseli. Efekt przedstawiono na następnej stronie.

## Czas na działanie — rozmiar obszaru widoku od 767 do 480 pikseli

Teraz zdefiniujemy style dla okien o szerokości od 767 do 480 pikseli. Rozmiary takie najczęściej mają urządzenia przenośne i tablety.




1. Reguły stylistyczne będziemy wpisywać w poniższym zapytaniu medialnym:

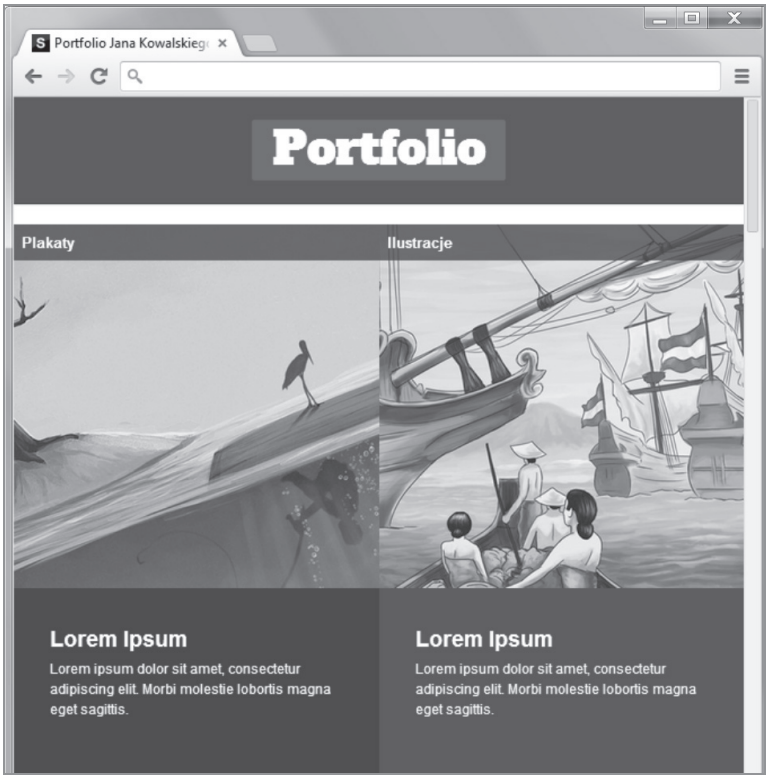
```
@media only screen and (min-width: 480px) and (max-width: 767px) {

}
```

2. Jako że obszar widoku został jeszcze bardziej zwężony, musimy nadać kolumnom nieco większe wymiary procentowe. Podzielimy szerokość kontenera na pół, aby otrzymać dwie kolumny o szerokości 50% szerokości obszaru widoku:

```
.portfolio .four.columns {
    width: 50%;
}
```

## Co uzyskaliśmy?

Zdefiniowaliśmy styl dla obszarów widoku o szerokości od 767 do 480 pikseli. Efekt tego zabiegu przedstawia poniższy zrzut ekranu:




## Czas na działanie — rozmiar obszaru widoku poniżej 480 pikseli

Na koniec dodamy style dla obszarów widoku o szerokości mniejszej niż 480 pikseli:

1. Style przeznaczone dla okien przeglądarek o szerokości poniżej 480 pikseli umieścimy w następującym zapytaniu medialnym:

```
@media only screen and (max-width: 479px) {
}
```

2. Ponieważ tym razem miejsca jest bardzo mało, zostawimy tylko jedną kolumnę o szerokości 100% szerokości obszaru widoku, aby zapewnić dobrą widoczność obrazu:

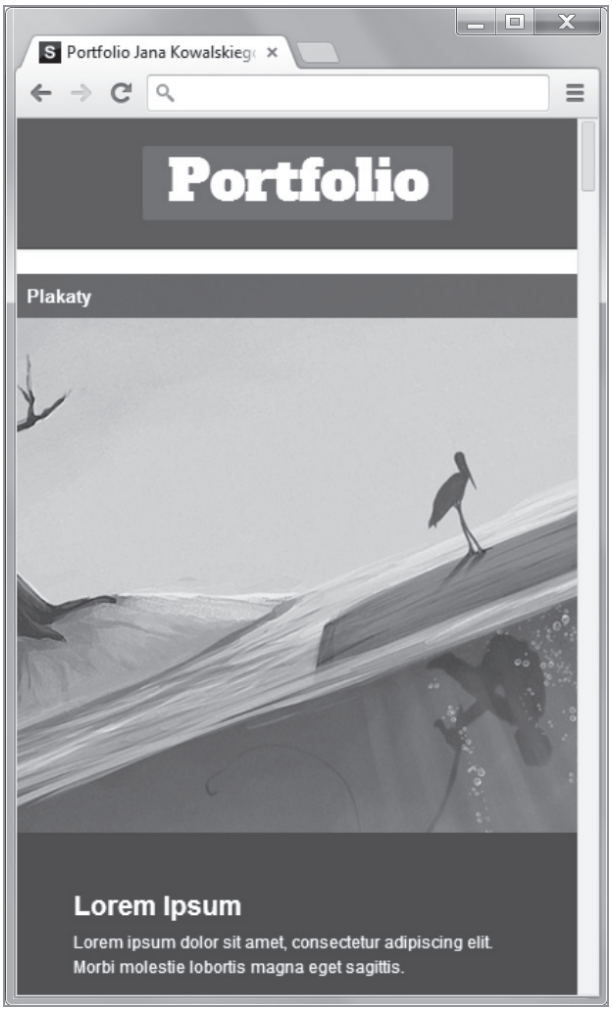
```
.portfolio .four.columns {
    width: 100%;
}
```

3. W stopce usuniemy deklarację własności `float` i ustawimy własność `text-align` na `center`:

```
.footer {
    text-align: center;
}
.contact, .social {
    float: none;
    display: block;
}
```

## Co uzyskaliśmy?

Zdefiniowaliśmy style przeznaczone dla okien przeglądarek o szerokości mniejszej niż 480 pikseli. Na poniższym zrzucie ekranu widać efekt tych działań:



# Testowanie strony w różnych rozmiarach obszaru widoku

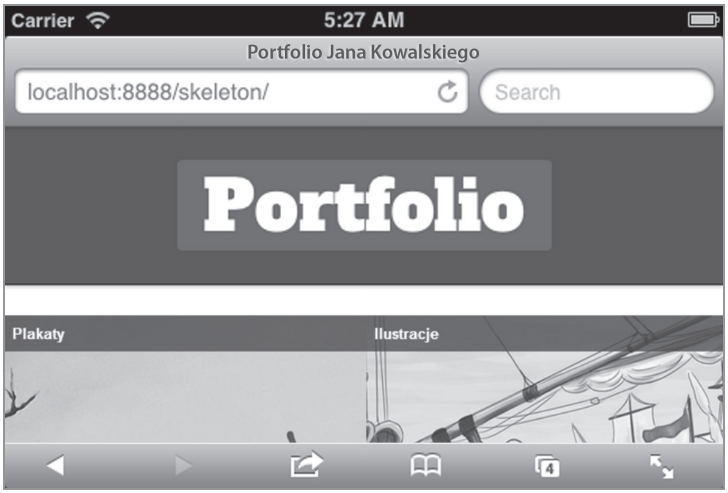
Strona jest gotowa i można zacząć jej testowanie. Sprawdzimy, jak wygląda w przeglądarce komputera stacjonarnego i w mniejszych przeglądarkach poprzez zmniejszenie okna tej samej przeglądarki. Można też użyć jednego z poniższych specjalnych narzędzi:

- Narzędzie *Widok responsywny* Firefoksa: *https://developer.mozilla.org/en-US/docs/Tools/Responsive_Design_View*.
- Responsinator: *http://www.responsinator.com/*.
- Screenqueries: *http://screenqueri.es/*.

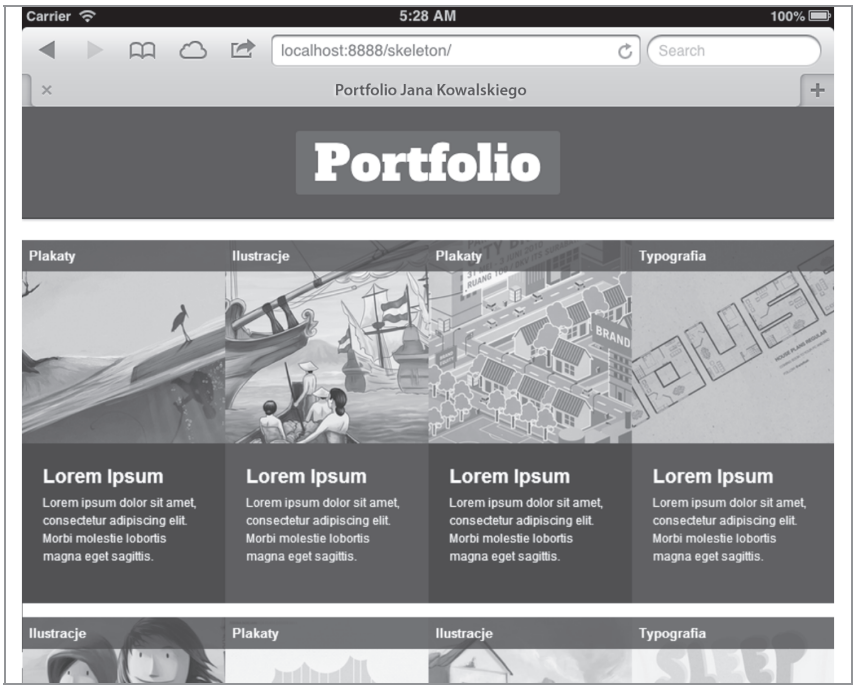
Najlepiej oczywiście testować stronę w prawdziwych urządzeniach — telefonach, tabletach, czytnikach — bo można sprawdzić, jak działa w realnych warunkach. Poniższe zrzuty ekranu przedstawiają wygląd naszej strony w urządzeniach iPhone i iPad.

Poniższy zrzut ekranu przedstawia stronę w telefonie iPhone w orientacji pionowej. W tak małym obszarze widoku (320 × 480 pikseli) nawigacja jest ukryta i zastąpiona nazwą kategorii wyświetloną nad każdą z miniatur. Także podpis każdego obrazu można zobaczyć pod obrazem.


Poniższy zrzut ekranu przedstawia naszą stronę w telefonie iPhone w orientacji poziomej. Podobnie jak w orientacji pionowej, nawigacja jest ukryta. Ale jako że szerokość obszaru widoku jest większa — 480 × 320 pikseli — wyświetlone zostały dwie miniatury w jednym rzędzie.

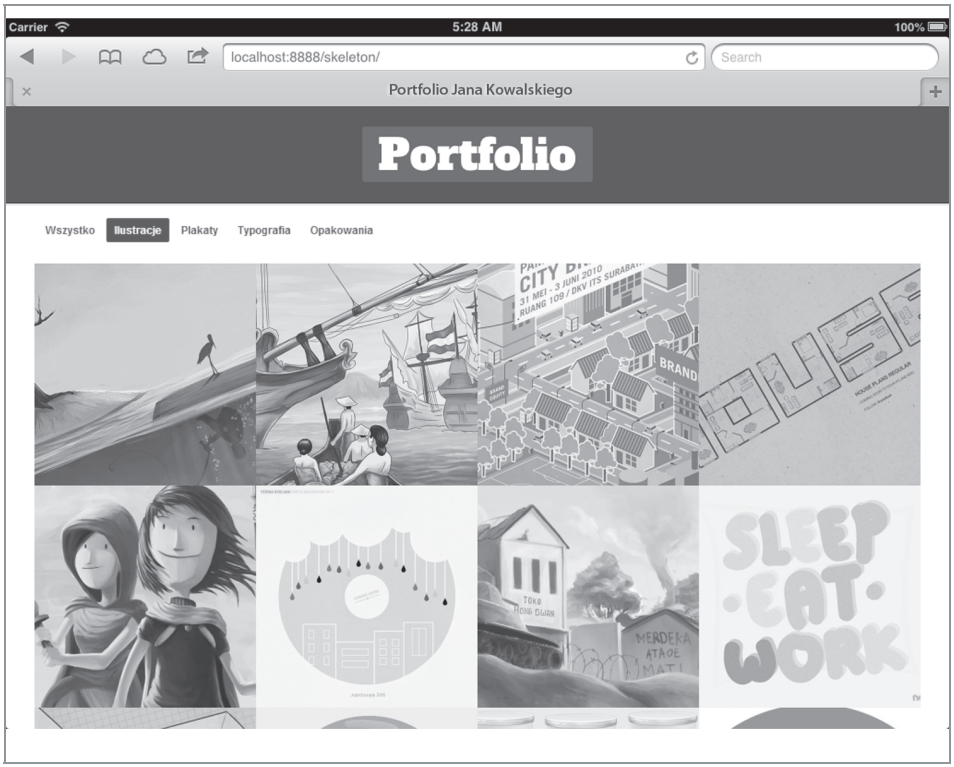



Poniższy zrzut ekranu przedstawia naszą stronę w iPadzie w orientacji pionowej. W tak szerokim obszarze widoku (768 × 1024 pikseli) zmieszczą się cztery miniatury w rzędzie, a opisy są cały czas widoczne i umieszczone pod miniaturami.




Na koniec zobaczymy jeszcze stronę w iPadzie w orientacji poziomej. Wymiary obszaru widoku wynoszą 1024 × 768 pikseli, więc jest wystarczająco miejsca, aby wyświetlić po cztery miniatury

w rzędzie. W tym rozmiarze ekranu widoczna jest też nawigacja. Można jej używać do sortowania obrazów.




# Podsumowanie

Właśnie ukończyliśmy prace nad naszą pierwszą elastyczną stroną zbudowaną przy użyciu technologii CSS3. W rozdziale tym wykonaliśmy następujące czynności:

- Dopracowaliśmy detale za pomocą nowych własności wprowadzonych w CSS3, takich jak `box-sizing`, `border-radius` oraz `box-shadow`.
- Utworzyliśmy atrakcyjny efekt zamiany obrazu i tekstu za pomocą przekształceń i przejść CSS3.
- Utworzyliśmy funkcję filtrowania portfolio za pomocą kombinacji selektorów CSS.
- Dostosowaliśmy styl stron do różnych rozmiarów obszaru widoku za pomocą zapytań medialnych CSS3.

W tym rozdziale poznaliśmy system szkieletowy Skeleton, w następnym zaś poznamy inny system służący do tworzenia elastycznych stron internetowych.

# Skorowidz

## F



## G



## H



## I



## J



## K



## L



## M

## N



## O



## P



## R



## S

## T



## U

## W



## Z

# Responsive Web Design. Nowoczesne strony WWW na przykładach

Responsywne strony WWW to dziś standard, do którego dążą wszyscy projektanci stron. W świecie smartfonów, tabletów, laptopów, telewizorów z dostępem do sieci oraz komputerów stacjonarnych stworzenie strony, która będzie się dobrze prezentowała i zachowa swoją funkcjonalność na każdym z urządzeń, to klucz do sukcesu! Ta książka to doskonałe źródło informacji na temat responsywnych stron internetowych — zacznij je projektować już teraz!

W trakcie lektury poznasz zalety i wady trzech najlepszych szkieletów do błyskawicznego tworzenia stron WWW. To Bootstrap, Skeleton oraz Foundation Zurb. Dodatkowo zapoznasz się z preprocesorami CSS oraz zaznajomisz się z nowościami wprowadzonymi w CSS3. Po opanowaniu wiedzy zawartej w kolejnych rozdziałach z łatwością zbudujesz przykładową witrynę produktu na podstawie szkieletu Bootstrap, portfolio na podstawie Skeletona oraz responsywną stronę firmową na podstawie Foundation Zurb. Dzięki praktycznemu podejściu do tematu błyskawicznie przyswoisz cenną wiedzę na temat tworzenia responsywnych stron WWW. Książka ta jest doskonałą lekturą dla wszystkich projektantów oraz pasjonatów tworzenia stron WWW!

**Buduj nowoczesne, responsywne strony WWW!**

**Sięgnij po książkę i dowiedz się, jak:**

- tworzyć elastyczne arkusze stylów CSS za pomocą preprocesorów
- zbudować responsywną stronę produktu za pomocą szkieletu Bootstrap
- wykorzystać możliwości Foundation Zurb
- stworzyć nowoczesną stronę, prezentującą się idealnie na różnych urządzeniach




ISBN 978-83-246-9190-6

9 788324 691906

Cena: 49,00 zł