Moja pierwsza strona internetowa w HTML5 i CSS3

Damian Wielgosik

Od Autora

Żyjemy w czasach, kiedy strony internetowe stały się częścią naszej codzienności, często udanie zastępując prasę i książki, przy okazji oferując ich użytkownikom całą gamę zupełnie nowych możliwości. Prawdopodobnie codziennie odwiedzasz przynajmniej kilka ulubionych miejsc w internecie. Robisz zakupy, wysyłasz wiadomości, grasz, sprawdzasz newsy i oglądasz zdjęcia z wakacji Twoich znajomych. Przy pomocy stron internetowych bawisz się, zarabiasz i poznajesz innych ludzi. Dlatego dziś umiejętność ich tworzenia wydaje się niezwykle cenna. Internet daje wiele możliwości rozwoju, a wiedza o tym, jak zbudowany jest internet pozwala zrozumieć zmiany, jakie zachodzą we współczesnym społeczeństwie i gospodarce.

Czy kiedykolwiek zastanawiałeś się, jak stworzyć własną stronę?

Zapraszam Cię do wspólnej podróży po Internecie. Przyjrzymy się stronom internetowym, które znamy i odwiedzamy na co dzień. Analizując je użyjemy porównań oraz analogii, które pozwolą Ci lepiej zrozumieć jak są zbudowane. Zajmuję się tym zawodowo od kilkunastu lat i jestem przekonany, że na koniec naszej przygody wspólnie stworzymy Twoją pierwszą stronę internetową.

Podziękowania dla Kuby Uczciwka za wsparcie graficzne; Piotra Mierzejewskiego, Pawła Czerskiego i Grzegorza Kaliciaka za wsparcie merytoryczne. Specjalne podziękowania dla Weroniki i Arnolda.

ROZDZIAŁ 1

Co klocki mają wspólnego ze stronami internetowymi?

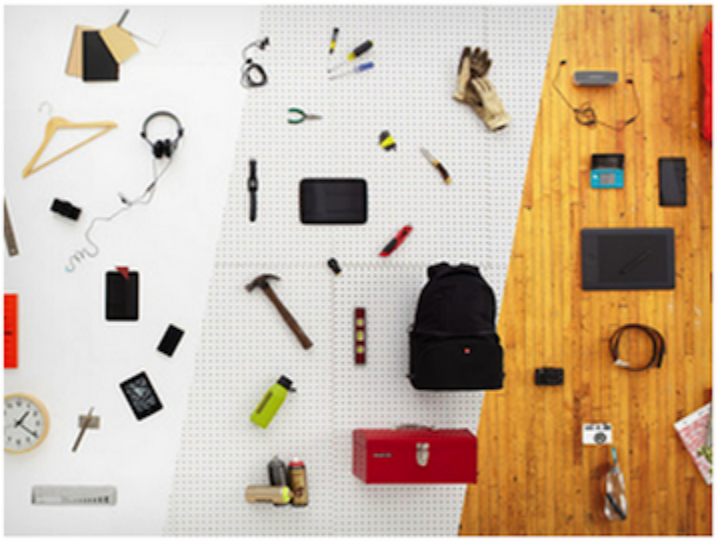
Bardzo lubię The Verge. Jest to serwis, na którym znajdziesz ciekawe artykuły o nowych technologiach, nauce i kulturze. Na stronie głównej jest kilkanaście wyróżnionych materiałów, które z reguły wyglądają podobnie. Tytuł, kategoria, data, obrazek, wstęp. Prezentują się one tak, jakby były zbudowane z klocków.

Przyjrzyjmy się jednemu z tekstów:

FEATURE

Back to School 2013: The Verge buying guide

Verge Staff | August 22, 2013 11:02 am



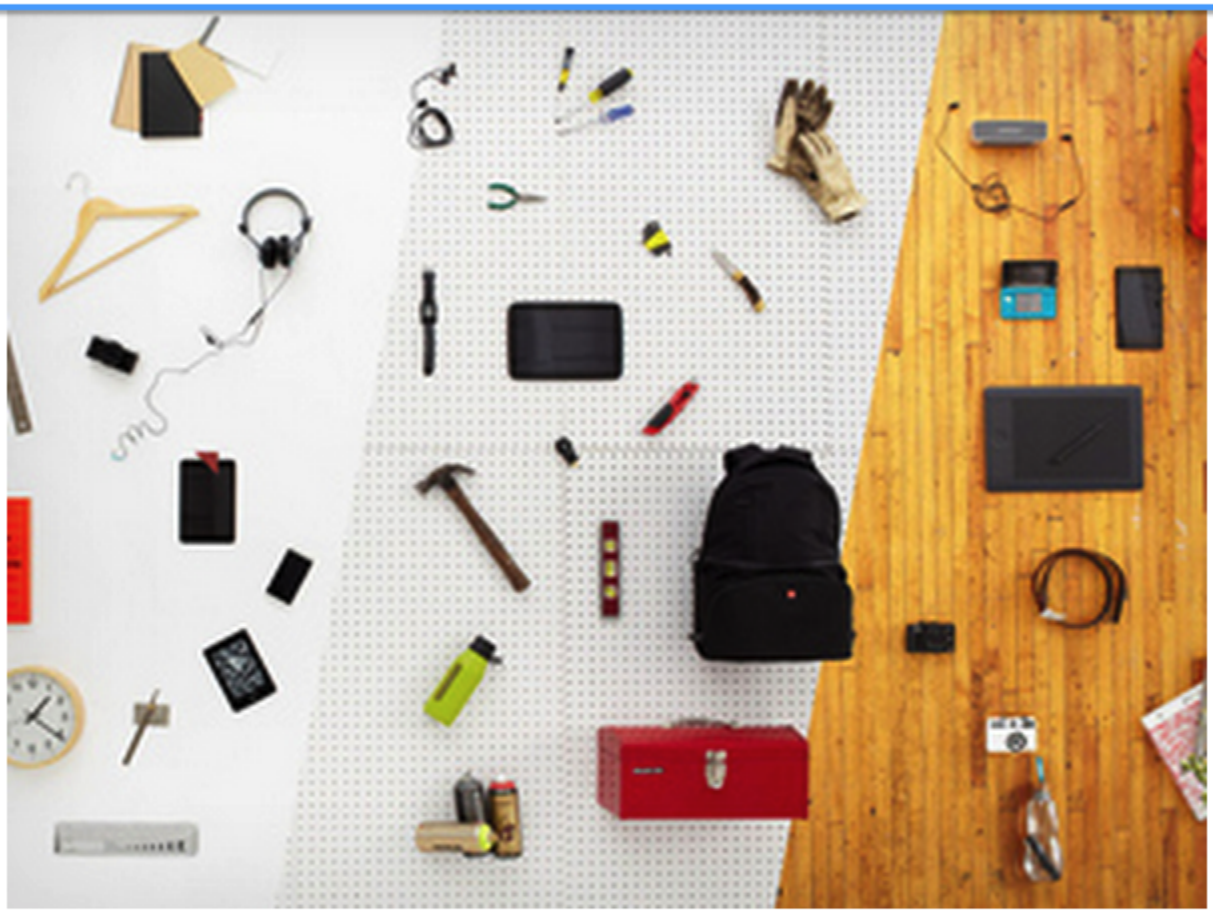
First, the bad news: summer's over. It's chillier out there every day, and soon you won't have beaches, road trips, or *kaiju* to entertain you. It's time to pull your jacket out of the closet and get to work.

Now the good news: You can make your own fun, you just need the right gear. So...

Na pierwszy rzut oka nie ma tu nic skomplikowanego. Spróbujmy zabawić się i wyróżnić każdy element, jakbyśmy mieli do czynienia z klockami.

Back to School 2013: The Verge buying guide

Verge Staff | August 22, 2013 11:02 am



First, the bad news: summer's over. It's chillier out there every day, and soon you won't have beaches, road trips, or *kaiju* to entertain you. It's time to pull your jacket out of the closet and get to work.

Now the good news: You can make your own fun, you just need the right gear. So...

W sumie uzyskaliśmy pięć klocków, które ułożone są jeden po drugim.

Jak zapewne pamiętasz z dzieciństwa, aby zbudować coś konkretnego, potrzebujemy wiele klocków. Każdy z nich pełni jakąś funkcję - budując domek, jeden przyda się do ścian, a drugi do podłogi. Nie ma uniwersalnego elementu, z którego zrobimy cokolwiek nam tylko przyjdzie na myśl. Spójrz na rysunek poniżej – duży wybór, prawda?

Podobnie jest ze stronami internetowymi. Budując je, używasz elementów według ich przeznaczenia. W naszym przykładzie już na pierwszy rzut oka widać, że mamy do czynienia z różnymi obiektami, więc zaznaczenie ich tym samym kolorem wydaje się

nieco mylące. Przecież tytuł to nie to samo, co akapit z treścią. Pełni on zupełnie inną funkcję.

Wyróżnijmy więc elementy naszego artykułu według funkcji, jakie pełnią na stronie.

Jest już o wiele ciekawiej. Mamy teraz kilka typów „klocków". Tylko dwa są tego samego rodzaju, a mianowicie akapity pod obrazkiem. Jako że niczym się nie różnią, przechowują tylko kolejne fragmenty tekstu, należą do tej samej grupy. Dla porządku, nazwijmy każdą z tych części, kierując się funkcją, jaką pełni w kontekście całego artykułu.

W ten sposób podzieliliśmy go na elementy według ich semantycznego znaczenia. Jest to dokładnie to, czego spodziewa się od nas przeglądarka internetowa - otóż musimy jej „powiedzieć", jaką funkcję w tekście pełni każdy element, o ile da się go logicznie wyodrębnić. Gdybyśmy tego nie zrobili, nasza strona zostanie wyświetlona jako jednolity tekst. Już tłumaczę, dlaczego.

Prawdopodobnie w przeszłości stworzyłeś kilka takich artykułów, używając któregoś z programów typu Word czy Pages. W aplikacjach do edycji tekstu uzyskanie takiego efektu dla nagłówka to kwestia kliknięcia kilku przycisków - “niech tytuł będzie nagłówkiem pierwszego stopnia, a wstęp zwykłym tekstem i voilà".

Gdybyśmy tworzyli i uruchamiali strony w Wordzie, mógłbyś zamknąć tę książkę na tym etapie i zająć się czymś innym (Breaking Bad wydaje się rozsądną alternatywą). Problem pojawia się, kiedy uświadomimy sobie, że właściwie powyższy artykuł jest częścią strony internetowej, którą wyświetliła przeglądarka, a nie program do edycji tekstu.

ROZDZIAŁ 2

Budujemy naszą pierwszą stronę internetową

Przypuśćmy, że chcemy zbudować stronę z ogłoszeniem o pracę. Powinno ono wyglądać tak, jak poniżej:

Przed tworzeniem jakiejkolwiek strony, bardzo dobrze jest podzielić jej treść na mniejsze elementy pod względem ich znaczenia. Spróbujmy wyróżnić każdą z części tego ogłoszenia, podobnie, jak zrobiliśmy to w przypadku newsa z The Verge.

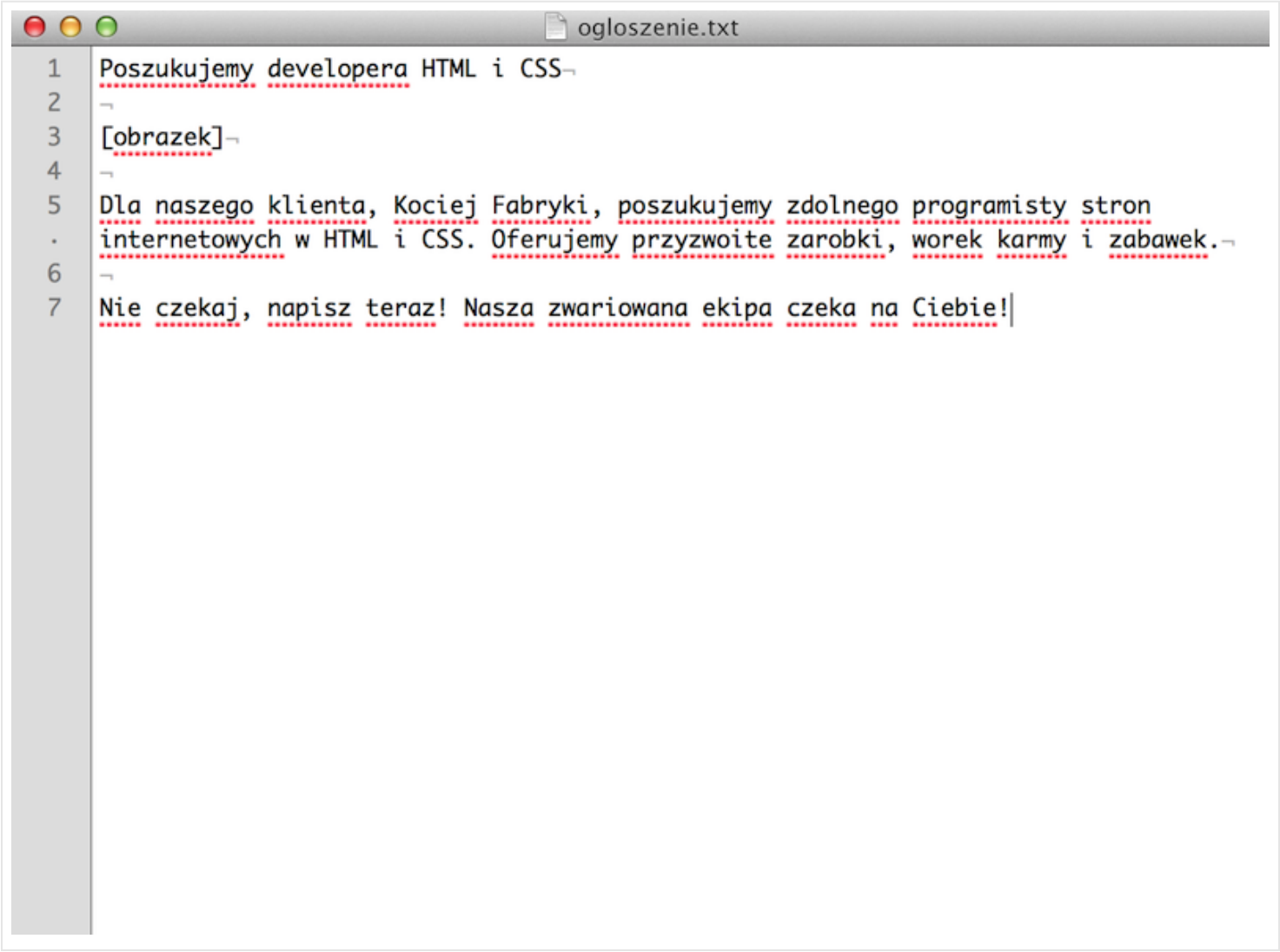
Poszukujemy developera HTML i CSS

Dla naszego klienta, Kociej Fabryki, **poszukujemy zdolnego programisty** stron internetowych w HTML i CSS. Oferujemy przyzwoite zarobki, worek karmy i zabawek.

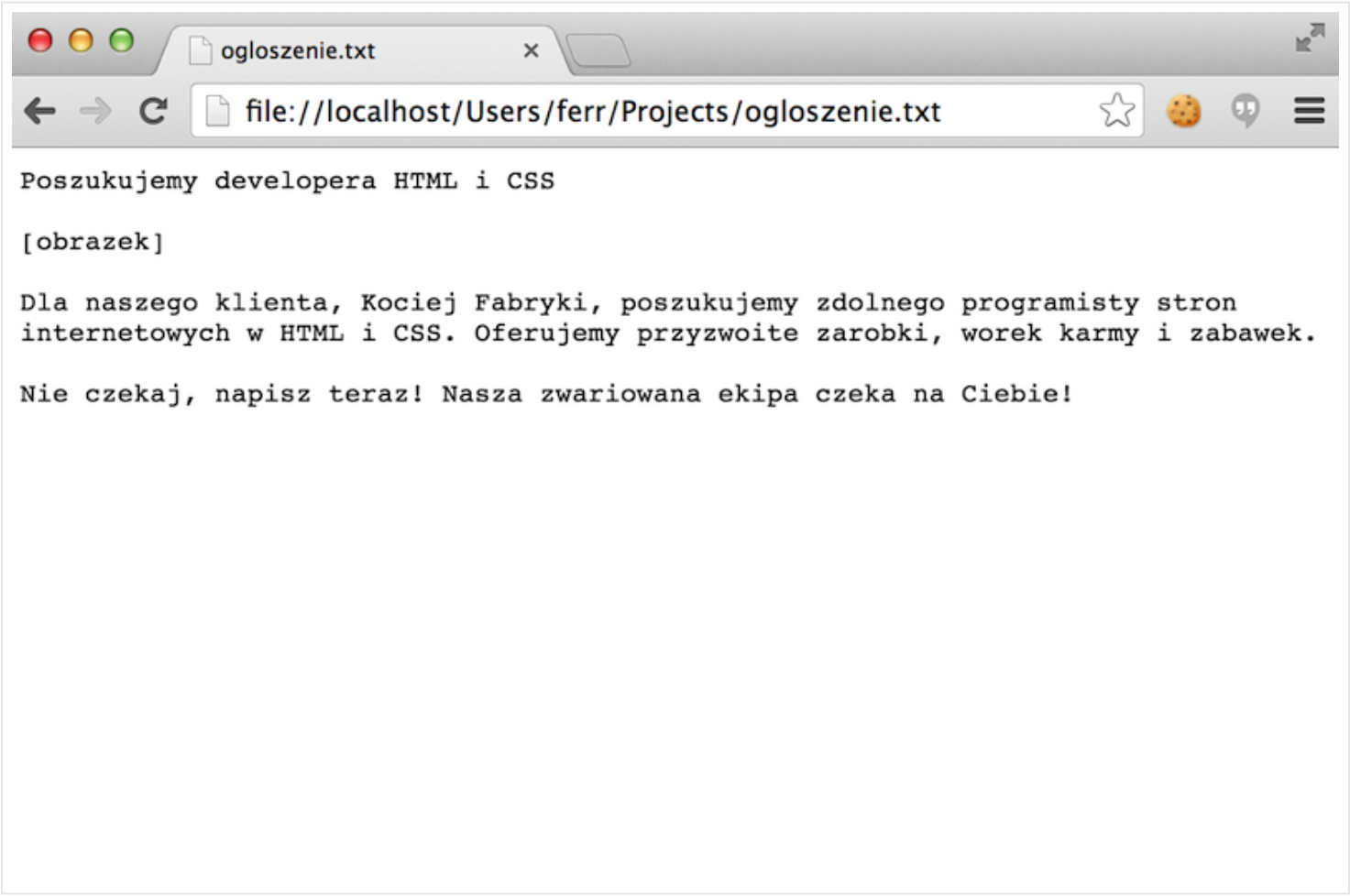
Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na Ciebie!

Szybka analiza pomoże nam lepiej zrozumieć, jakie fragmenty tekstu powinniśmy zaznaczyć. Na czerwono – nagłówek artykułu, na zielono – obrazek, a fioletowym kolorem oznaczyliśmy dwa akapity z treścią ogłoszenia.

Wracając na chwilkę do analogii z edytorami tekstu, gdyby strony internetowe tworzono, jak zwykłe dokumenty tekstowe, prawdopodobnie otworzyłbyś teraz notatnik lub podobny program i napisał w nim treść ogłoszenia, a całość zapisał jako plik tekstowy:



Po chwili oddechu sprawdziłbyś swoje dzieło w przeglądarce internetowej:



Nie wygląda to jednak tak, jak sobie zamierzyliśmy. Mamy przed sobą jedną masę tekstu, nie wyświetla się też obrazek. Co teraz? Może powinniśmy spróbować stworzyć to ogłoszenie w Wordzie? Albo w Photoshopie? Nie do końca. Popełniliśmy kilka błędów, które trzeba naprawić. Największy z nich polega na tym, iż uznaliśmy, że strony internetowe to tylko zwykły tekst.

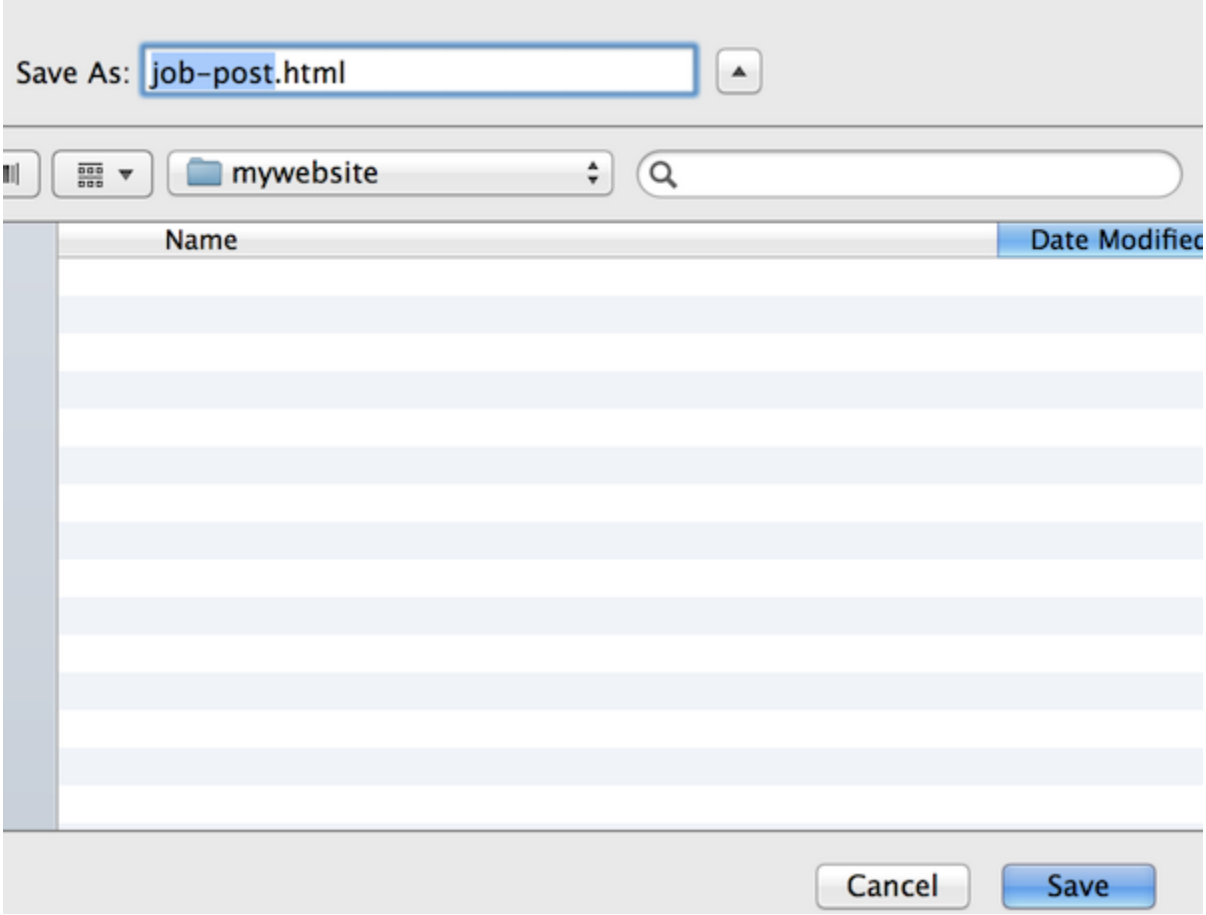
Przede wszystkim przeglądarka nie zrozumiała, który fragment tekstu jest tytułem, a który powinien być obrazkiem. Musimy nazwać dla niej każdy element według funkcji, jaką pełni w tekście. Dokonajmy więc małego tłumaczenia.

Jak zapewne zauważyłeś, pojawiły się specjalne oznaczenia, które na razie jednak nic Ci nie mówią. Zaufaj mi, lada moment dowiesz się, dlaczego są tutaj. Na razie spróbuj tylko skopiować je po kolei od góry do dołu, jakbyś brał do ręki klocki i układał je jeden pod drugim.

Powinieneś uzyskać coś podobnego:

```
<h1>Poszukujemy developera HTML i CSS</h1>
<img src="images/white-cat.jpg">
<p>Dla naszego klienta, Kociej Fabryki, poszukujemy
zdolnego programisty stron internetowych w HTML i
CSS. Oferujemy przyzwoite zarobki, worek karmy i
zabawek.</p>
<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa
czeka na Ciebie!</p>
```

Następnie zapisz całość do pliku html (przy okazji – do kodowania dobrze jest ściągnąć specjalny edytor kodu, taki jak Sublime Text Editor – zapewnie on o wiele wyższą wygodę w pisaniu kodu HTML niż w normalnym edytorze tekstu):



Spróbujmy teraz wyświetlić nowy plik w przeglądarce:

Poszukujemy developera HTML i CSS

Dla naszego klienta, Kociej Fabryki, **poszukujemy zdolnego programisty** stron internetowych w HTML i CSS. Oferujemy przyzwoite zarobki, worek karmy i zabawek.

Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na Ciebie!

Jest nieco lepiej. Każdy z fragmentów tekstu wygląda teraz nieco inaczej, dodatkowo wyświetlił się obrazek. Nie jest to jeszcze finalna wersja, bo brakuje kolorów, pogrubienia i krój tekstu jest nieco inny. Nim się tym zajmiemy, skupmy się jeszcze na samej strukturze strony. Porównajmy, jak wyglądał zwykły dokument tekstowy, a jak prezentuje się kod HTML:

job-post.html

```
1  <h1>Poszukujemy developera HTML i CSS</h1>
2
3  <img src="images/white-cat.jpg">
4
5  <p>Dla naszego klienta, Kociej Fabryki,
   poszukujemy zdolnego programisty stron
   internetowych w HTML i CSS. Oferujemy
   przyzwoite zarobki, worek karmy i zabawek.
   </p>
6
7  <p>Nie czekaj, napisz teraz! Nasza
   zwariowana ekipa czeka na Ciebie!</p>
```

Poszukujemy developera HTML i CSS

```
1  Poszukujemy developera HTML i CSS
2
3  [obrazek]
4
5  Dla naszego klienta, Kociej
   Fabryki, poszukujemy zdolnego
   programisty stron internetowych w
   HTML i CSS. Oferujemy przyzwoite
   zarobki, worek karmy i zabawek.
6
7  Nie czekaj, napisz teraz! Nasza
   zwariowana ekipa czeka na Ciebie!
```

Dzięki temu, że otoczyliśmy fragmenty tekstu specjalnym znacznikami, powstał zupełnie inny dokument, który zrozumiała przeglądarka. Brawo! To właśnie jest HTML, czyli **Hypertext Markup Language**, a prościej - język znaczników (zwanych też tagami), dzięki którym możesz opisać tekst tak, by dogadać się z przeglądarką (a także z wyszukiwarkami i kilkoma innymi programami, o których dowiesz się na kolejnych stronach). Znaczniki HTML są jak słowa w mowie. Bez nich nie można zbudować strony internetowej.

W naszym przypadku użyliśmy tylko kilku tagów. Oto ich lista wraz z funkcją, jaką pełnią:

- `<h1>` - nagłówek pierwszego stopnia
- `<img>` - obrazek
- `<p>` - akapit tekstu

Jeśli więc na swojej stronie masz kilka akapitów z tekstem - otaczasz je znacznikiem `<p>`. Jeśli najważniejszy nagłówek w tekście - `<h1>`. I tak analogicznie.

Budowa każdego z tagów jest bardzo prosta:

```
<nazwa>treść</nazwa>
```

Zwróć uwagę, że każdy znacznik trzeba, jak to się mówi w żargonie HTMLowców, "zamknąć". Jeśli więc uznasz, że w danym miejscu kończy się akapit, piszesz </p>. Jeśli to już koniec nagłówka pierwszego stopnia, dodajesz </h1>. Zwróć uwagę na ukośnik przed nazwą znacznika, to on daje znać przeglądarce, że to już koniec danego elementu.

Musisz też wiedzieć, że są też takie, których nie musisz zamykać. Na przykład <img>, służący do wstawiania obrazków. W naszym przykładzie użyliśmy go następująco:

```
<img src="images/white-cat.jpg">
```

Jest ich jeszcze kilka - dowiesz się o wszystkich w dalszej części tej książki.

Najważniejsze, byś pamiętał, że tak jak słowniki językowe definiują zasób słów danego języka, tak i w HTML występuje swego rodzaju słownik, gdzie znajdziesz wszystkie znaczniki wraz z opisem, kiedy należy ich użyć. Na razie poznaliśmy tylko kilka. Klasyfikacją wszystkich zajmuje się W3C, specjalna organizacja, która na bieżąco reguluje, jak ma wyglądać język HTML. Nie zasiada w niej co prawda Bralczyk, ale możesz sobie wyobrazić, że podobnie do języka polskiego, pracują tam specjaliści. Listę wszystkich możliwych znaczników znajdziesz na ich stronie: http://www.w3.org/TR/html-markup/elements.html. Zapamiętaj również, że podobnie jak języki naturalne, HTML pod wpływem czasu co jakiś czas się zmienia. Obecnie mamy do czynienia z jego piątą wersją. Poprzednie wersje nosiły nazwę XHTML i HTML4.

Wracając do naszego przykładu, uzyskaliśmy rozwiązanie, którym jest poniższy kod:

```
<h1>Poszukujemy developera HTML i CSS</h1>

<img src="images/white-cat.jpg">

<p>Dla naszego klienta, Kociej Fabryki, poszukujemy zdolnego
programisty stron internetowych w HTML i CSS. Oferujemy
przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
```

Zawiera on dwa akapity tekstu:

```
<p>Dla naszego klienta, Kociej Fabryki, poszukujemy zdolnego
programisty stron internetowych w HTML i CSS. Oferujemy
przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
```

Kontynuując dalszą analizę treści naszego ogłoszenia warto zwrócić uwagę na fragment pierwszego akapitu: “poszukujemy zdolnego programisty". Wydaje mi się, że jest to najistotniejszy element całego tekstu. Zaznaczmy w HTML ten fakt, powiedzmy przeglądarce “hej, to jest nieco ważniejszy fragment" z całego kontekstu. Posłuży nam do tego element `<strong>`, którym oznaczamy ważniejsze znaczeniowo fragmenty tekstu. Dodajmy go do akapitu:

```
<p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
```

Zauważ, że znalazł się on wewnątrz elementu `<p>` - to znaczy najpierw otworzyliśmy tag `<p>` i pomiędzy nim umieściliśmy kolejny - `<strong>`. W slangu programistycznym

mówimy, że jest “dzieckiem" elementu <p>. Zapytasz zapewne, czy w ten sposób można zagnieżdżać różne tagi. Oczywiście, jednak każdy tag ma określoną listę możliwych elementów HTML, które można w nim umieścić, a więc zagnieżdżając dwa tagi HTML musimy najpierw sprawdzić, czy jest to dozwolone. To tak, jakby jeden klocek zbudowany byłby z kilku – w tym wypadku <p> stworzony został z tekstu i tagu .

Efekt wyróżnienia fragmentu tekstu prezentuje się następująco:

Dla naszego klienta, Kociej Fabryki, **poszukujemy zdolnego programisty** stron internetowych w

Zapamiętaj też, że przeglądarka internetowa wyświetla taki fragment jako pogrubiony, jednak nie jest to reguła, a jedynie miły efekt uboczny. Utarło się, że powoduje pogrubienie tekstu, jednak nie należy na tym polegać. Powodem użycia tego tagu HTML powinno być zaznaczenie ważnego fragmentu, a nie uzyskanie pogrubienia tekstu. Wszelkie efekty wizualne uzyskujemy dzięki językowi CSS, o którym dowiesz się więcej na dalszych stronach.

ROZDZIAŁ 3

Cechy kompletnego kodu HTML

Parę minut temu powstała nasza pierwsza strona internetowa. Muszę Cię jednak nieco zmartwić – nie jest jeszcze kompletna według obowiązujących standardów pisania stron. Tak jak inżynierowie muszą trzymać się wytycznych projektowych, a restauratorzy regulaminu Sanepidu, tak i web developerzy muszą stosować się do pewnych z góry wyznaczonych zasad, które definiują dobrze wykonaną robotę. W przypadku tworzenia stron internetowych powstały standardy sieciowe, które stoją na straży dobrych praktyk. Mówią nam między innymi, jakie tagi możemy zagnieżdżać w jakich (a więc to, co przed chwilą omówiliśmy), czy też jak powinien wyglądać pełny dokument HTML, do którego nikt nie może się przyczepić. No właśnie, zapytasz pewnie: jak? Już tłumaczę!

Zacznijmy od kodu ogłoszenia o pracę:

```
<h1>Poszukujemy developera HTML i CSS</h1>

<img src="images/white-cat.jpg">
<p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
```

Czego tu brakuje, by było "po bożemu"? Zaczynając tradycyjnie od góry, w każdym szanującym się dokumencie HTML należy umieścić najpierw tzw. doctype:

```
<!DOCTYPE html>
<h1>Poszukujemy developera HTML i CSS</h1>

<img src="images/white-cat.jpg">
<p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
```

"Powie" to przeglądarce o tym, że ma **na pewno** do czynienia z dokumentem HTML i podczas ładowania strony powinna potraktować powyższy kod według ustalonych dla języka standardów. Zostanie więc poinformowana o kilku regułach, jakimi powinna się kierować, wyświetlając naszą stronę – będzie między innymi wiedzieć, jakie tagi mogą być zawarte w jakich, czy też które są dozwolone, a które nie. Pomoże jej to lepiej zinterpretować nasz kod.

Pod doctype'em powinniśmy z kolei umieścić znacznik `<html>`:

```
<!DOCTYPE html>
<html>
<h1>Poszukujemy developera HTML i CSS</h1>

<img src="images/white-cat.jpg">
<p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
</html>
```

Zauważ, że cały dotychczasowy kod naszej strony umieściliśmy pomiędzy `<html></html>`:

```
<!DOCTYPE html>
<html>
nasz kod
</html>
```

A więc wszystko, co należy do naszej strony zawiera się w tagu `<html>`. Ma to sens - kod HTML powinien zawierać się w znaczniku `<html>`.

Idąc dalej, znacznik `<html>`, który dodaliśmy, nie jest jeszcze gotowy. Musimy umieścić jeszcze informację, w jakim języku jest nasza strona. Jako że jest to język polski, użyjmy stosownego oznaczenia:

```
<!DOCTYPE html>
<html lang="pl">
<h1>Poszukujemy developera HTML i CSS</h1>

<img src="images/white-cat.jpg">
<p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

<p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
</html>
```

Niniejszym dodaliśmy do naszego tagu tzw. "atrybut". Atrybutem w HTML jest wszystko, co napiszemy obok nazwy znacznika HTML, ale pomiędzy `<>`. Atrybuty definiujemy według następującego szablonu:

```
nazwaAtrybutu="wartość atrybutu"
```

Pozwól mi podać kilka przykładów z życia. Na swojej drodze jako developer HTML możesz spotkać ich wiele, ponieważ oprócz "lang" jest ich całe mnóstwo:

- `<div id="sidebar"></div>` - nazwą atrybutu jest "id", a jego wartością "sidebar"
- `<p class="landscape"></body>` - nazwą atrybutu jest "class", a jego wartością "landscape"

- <input type="text"> - nazwą atrybutu jest "type", a jego wartością "text"

Dlaczego używamy atrybutów? Zauważ, że sam znacznik HTML – w naszym wypadku <html> – może być niewystarczający, jeśli chodzi o informacje, jakie zawiera. Dzięki atrybutom możemy opisywać tagi, których używamy i dodawać więcej informacji o nich. W tym wypadku daliśmy znać, że nasz dokument HTML został napisany w języku polskim, a więc dodaliśmy do tagu <html> atrybut lang (skrót od language po angielsku). Takich przypadków jest więcej i dowiesz się o nich w trakcie lektury tej książki.

Atrybutów dla jednego tagu HTML może być wiele. Tutaj kilka przykładów z życia wziętych:

```
<input type="text" value="wpisz tutaj tekst">
<a href="http://functionite.pl" title="Strona szkoleniowa firmy
Functionite">szkolenie z HTML</a>
```

I tak dalej.

Ważną informacją jest fakt, że standard języka HTML, oprócz zasad zagnieżdżania tagów, zdefiniował też dla każdego z nich listę dostępnych atrybutów. A więc mając do dyspozycji na przykład tag <p> łatwo dowiemy się, jakich atrybutów możemy użyć z nim w parze.

Kończąc rozprawę o atrybutach warto dodać, że już wcześniej użyłeś jednego z nich. Pamiętasz obrazek? Wstawiłeś go do dokumentu HTML poprzez umieszczenie następującej linijki:

```
<img src="images/white-cat.jpg">
```

W tym przypadku został użyty atrybut src z wartością images/white-cat.jpg. Bez niego mielibyśmy jedynie sam tag HTML:

```
<img>
```

Jeślibyśmy tak to zostawili, przeglądarka nie miałaby żadnej informacji z jakiego źródła wyświetlić obrazek. Dzięki atrybutowi "src" mówimy przeglądarce: "hej, załaduj obrazek ze źródła, który Ci podałem i wyświetl go". W tym wypadku z "images/white-cat.jpg", a więc przeglądarka w folderze, w którym znajduje się nasza strona poszuka katalogu "images", a potem obrazka "white-cat.jpg". Można również podać adres do pliku znajdującego się na innym serwerze, na przykład http://4.bp.blogspot.com/-z4sMggeD4dg/UO2TB9INuFI/AAAAAAAAdwc/Kg5dqlKKHrQ/s1600/funny-cat-pictures-032-025.jpg.

Tak właśnie działają atrybuty. Dzięki nim podajemy przeglądarce dodatkowe informacje, które są pomocne w wyświetlaniu danych tagów HTML lub czasem też niezbędne - jak przy obrazkach.

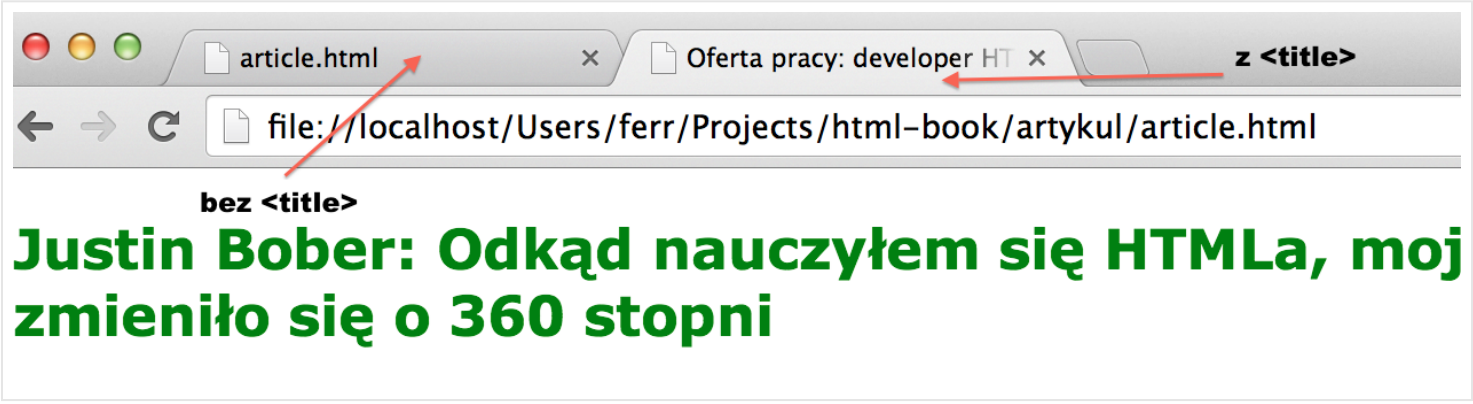
Przechodząc jednak z powrotem do ulepszania naszego dokumentu, dodajmy do niego kolejne, niezbędne fragmenty. Tym razem będzie to tag `<head>`, w którym umieszcza się różne elementy, nieco mniej istotne niż sama treść dokumentu. Na przykład jego tytuł, informacje pomocne dla wyszukiwarek internetowych i tak dalej. `<head>` umieszczamy w `<html>`:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
  </head>
  <h1>Poszukujemy developera HTML i CSS</h1>

  <img src="images/white-cat.jpg">
  <p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

  <p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
</html>
```

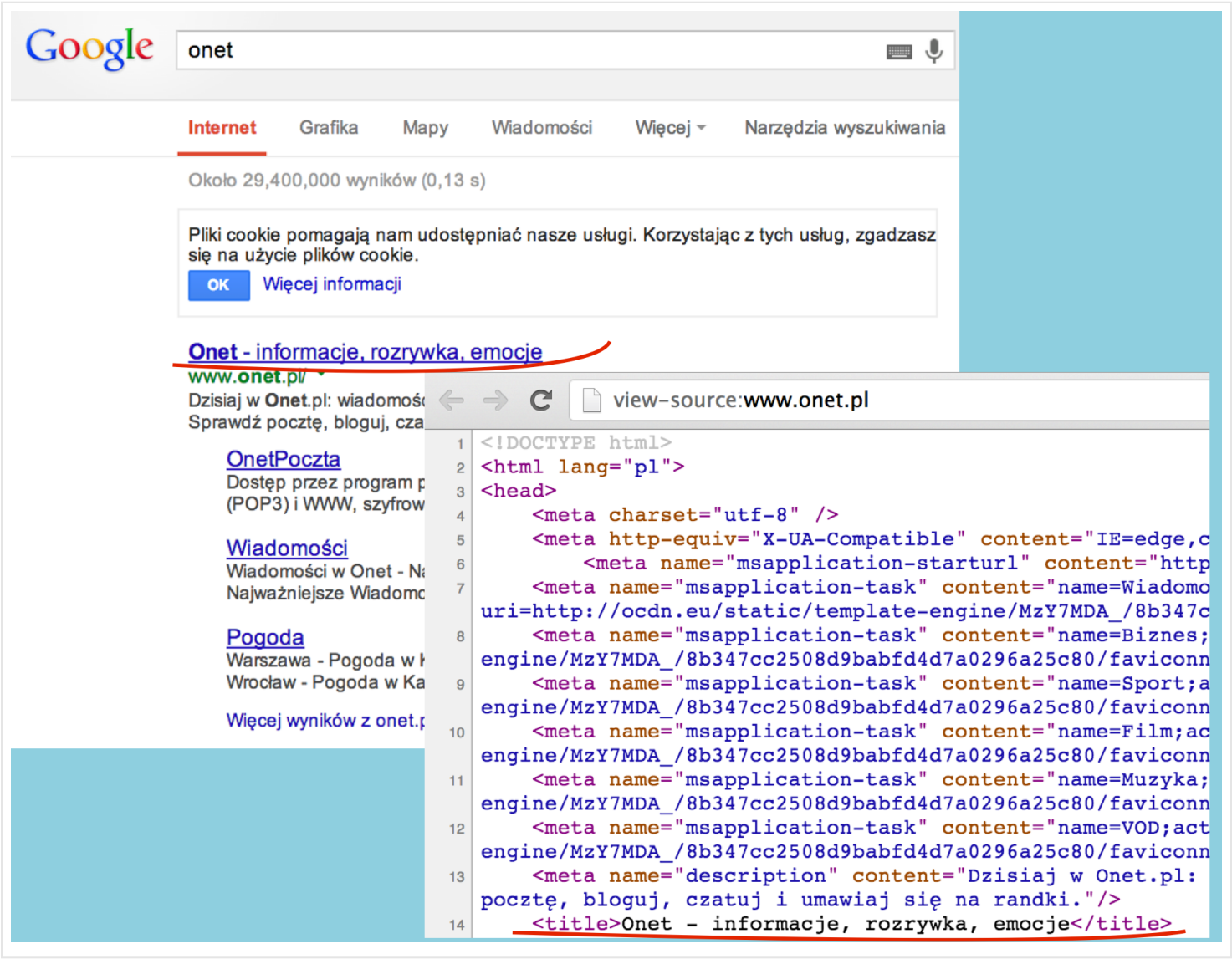
Na razie jest on pusty, jednak jak powiedziałem, można umieścić w nim m.in. tytuł całego dokumentu. Jako że tworzymy stronę z ofertą pracy, niech jej tytuł brzmi "Oferta pracy: developer HTML i CSS". Brzmi w porządku, prawda? Do dzieła, dodajmy go!

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Oferta pracy: developer HTML i CSS</title>
  </head>
  <h1>Poszukujemy developera HTML i CSS</h1>

  <img src="images/white-cat.jpg">
  <p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

  <p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
</html>
```

Jak widzisz, dodaliśmy ten tytuł pomiędzy znacznikiem <title>. Zapewne zapytasz, jaka jest różnica i po co w ogóle to zrobiliśmy? Niech wyjaśnią Ci to dwa poniższe zrzuty ekranu, gdzie pokazane jest wyświetlanie strony z <title> i bez <title>.



Gdyby z kolei Onet nie miał znacznika <title>, ciężko byłoby się połapać choćby w wynikach wyszukiwania:

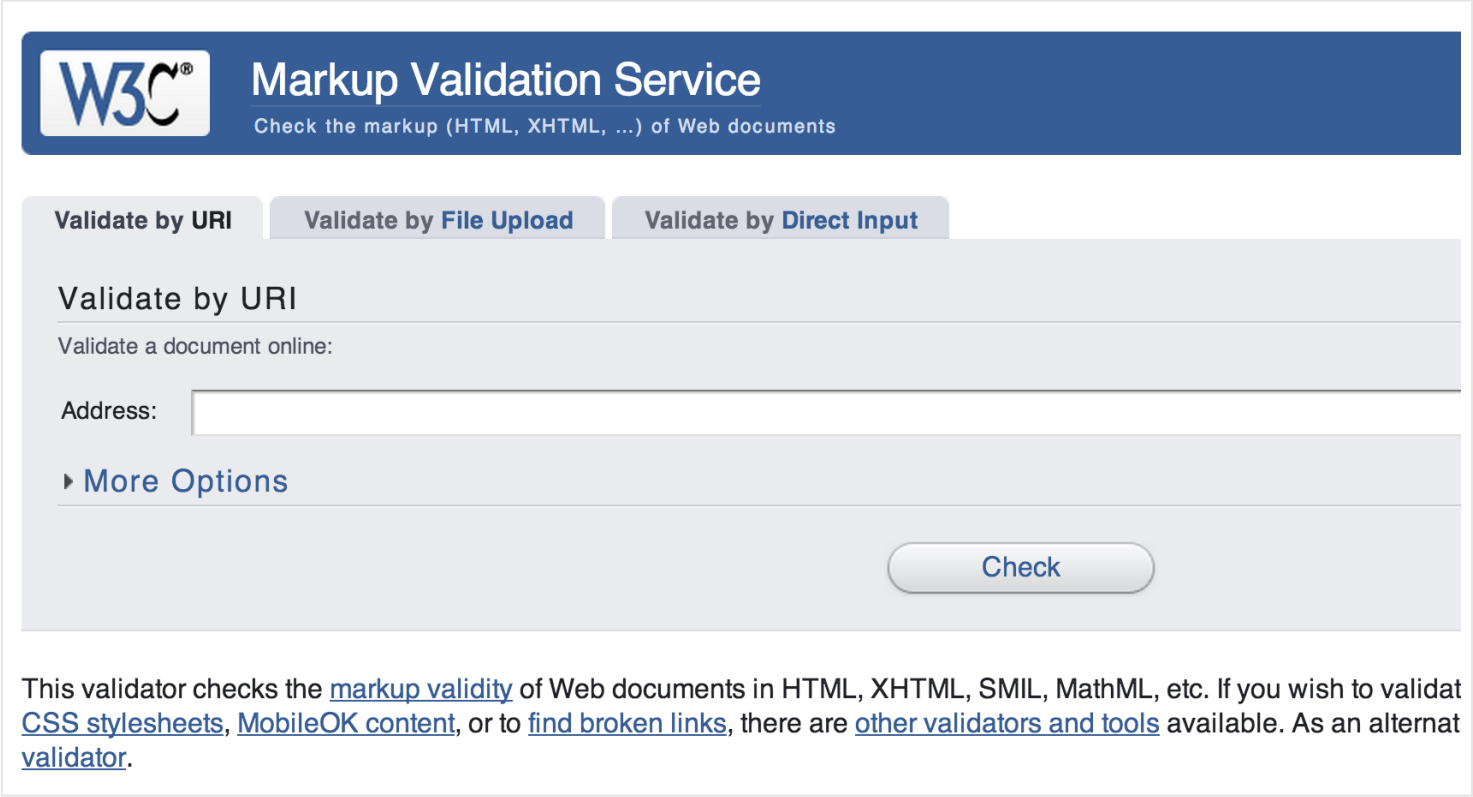


Jak widzisz, jest to bardzo ważny element kodu Twojej strony. Bez niego byłaby ona bez nazwy, co zdecydowanie odbiłoby się na odwiedzinach internautów – osobiście nie zaufałbym stronie, która nie ma nawet tytułu.

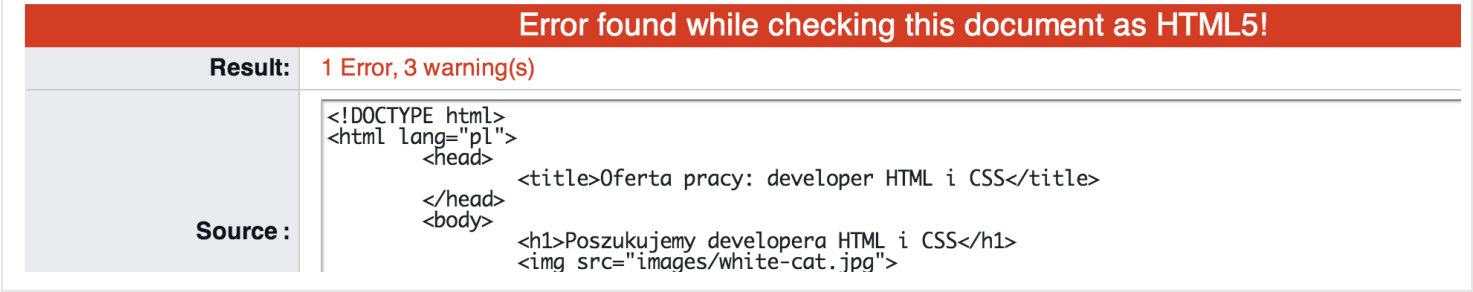
Pozostała nam teoretycznie ostatnia rzecz do naprawienia. Musimy dodać znacznik `<body>`, w którym zawsze umieszczamy treść całej strony. Wszystko, co wyświetla się w przeglądarce pod paskiem adresu jest umieszczone w `<body>`.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Oferta pracy: developer HTML i CSS</title>
  </head>
  <body>
    <h1>Poszukujemy developera HTML i CSS</h1>

    <img src="images/white-cat.jpg">
    <p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

    <p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
  </body>
</html>
```

Hurra! Wygląda to na koniec naszych starań i wydaje się, że wszystko się zgadza. Jako początkujący twórca stron internetowych zapytasz zapewne, jak sprawdzić, czy wszystko jest ok, czy zagnieździłeś odpowiednio tagi i użyłeś dobrych atrybutów. Przecież nie każdy ma wśród znajomych zaawansowanych developerów HTML, których może poprosić o konsultacje. Intuicja Cię nie zawiodła. Istnieje specjalne narzędzie, którym sprawdzamy poprawność naszego kodu. Zrobiła go organizacja W3C, ta sama, która definiuje standard i stoi na straży poprawności dokumentów HTML. To narzędzie nazywa się po prostu walidatorem HTML i jest dostępne na stronie http://validator.w3.org. Gdy wejdziesz na tę stronę powinieneś zobaczyć taki obrazek:



Wykorzystajmy go i sprawdźmy, czy nasz kod jest poprawny.



Ups. Nie udało się. Mamy jeden błąd i 3 ostrzeżenia. Przyjrzyjmy się im.

Jedyny błąd jaki zgłosił walidator dotyczy naszego obrazka. Oto treść komunikatu:

> An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

Okazuje się, że tagi `<img>` w naszym dokumencie muszą mieć zawsze atrybut `alt`, który zostanie użyty, gdyby przeglądarka nie mogła załadować obrazka. Wtedy wyświetli się treść tego atrybutu - lepsze to, niż pusta przestrzeń, prawda? Przy okazji użytkownik

czytający naszą stronę nie straci być może cennych informacji, które zawarte były w postaci graficznej.

Nim wybierzemy wartość atrybutu "alt", zajrzyjmy do specyfikacji języka HTML, by upewnić się, że dokonamy właściwego wyboru. Jak czytamy na stronie W3C:

> "Gives the fallback content for the image. The requirements on the alt attribute's value are described in the next section."

Przechodząc dalej, wiedz, że wyróżniono kilka scenariuszy, kiedy i jak mamy używać atrybutu alt. W naszym wypadku mamy do czynienia z obrazkiem, który został umieszczony dla dekoracji posta i sam w sobie nie zawiera żadnej treści, wykresu czy innej informacji dla czytelnika. W takim razie powinniśmy zostawić atrybut "alt" pusty, a mówi o tym poniższy fragment specyfikacji:

> "However, a decorative image that isn't discussed by the surrounding text but still has some relevance can be included in a page using the img element. Such images are decorative, but still form part of the content. In these cases, the alt attribute must be present but its value must be the empty string."

W innych przypadkach powinniśmy wybrać taką treść, która słownie zastępowałaby to, co znajduje się na danym obrazku. Bardzo ważne jest, aby zapamiętać, że nie chodzi o sam opis grafiki, a o tekst, który byłby reprezentacją obrazka, gdybyśmy nie mogli go z różnych przyczyn załadować (np. brak internetu, usunięcie materiału z serwera i tak dalej).

Szczegółowe wytyczne dla kilku wyróżnionych przypadków znajdziesz na stronie ze specyfikacją, jak powinniśmy stosować tag <img>: http://www.w3.org/wiki/HTML/Elements/img

Przechodząc jednak do sedna, ustaliliśmy, że należy dodać pusty element alt, więc zróbmy to:

```
<img src="images/white-cat.jpg" alt="">
```

Sprawdźmy ponownie, czy teraz nasz dokument jest poprawny...

Udało się, zero błędów!

Zostały jednak ostrzeżenia. Najważniejszy z nich brzmi następująco:

> "No character encoding information was found within the document, either in an HTML meta element or an XML declaration. It is often recommended to declare the character encoding in the document itself, especially if there is a chance that the document will be read from or saved to disk, CD, etc."

Bez obaw, to prosty komunikat. Nim jednak przejdę do jego wytłumaczenia, spróbuj przez chwilę przeanalizować następujący przykład, który pozwoli Ci jeszcze lepiej zrozumieć problem.

Otóż chciałbym, byś zmodyfikował nieco treść ogłoszenia. Być może zauważyłeś, ale celowo nie umieściłem w nim polskich znaków diakrytycznych. Proszę, dodaj kilka. Ja po prostu umieszczę dodatkowy akapit na końcu:

```
<p>PS Czyż polskie znaki nie są fajne i w ogóle?</p>
```

Całość wyświetla się teraz następująco:

Poszukujemy developera HTML i CSS

Dla naszego klienta, Kociej Fabryki, **poszukujemy zdolnego programisty** stron internetowych w HTML i CSS. Oferujemy przyzwoite zarobki, worek karmy i zabawek.

Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na Ciebie!

PS CzyÅ¼ polskie znaki nie sÄ… fajne i w ogÃ³le?

Spójrz na ostatnią linijkę. Zamiast polskich znaków wyświetliły się jakieś dziwne krzaczki. Dlaczego tak się stało? Ma to związek z komunikatem, który dostaliśmy od walidatora. Okazuje się, że nie podaliśmy jakiego kodowania znaków ma użyć przeglądarka, więc ta wybrała sobie taki, który nie zawiera polskich znaków diakrytycznych. Kodowanie to nic innego jak zestaw znaków, jakich może użyć przeglądarka do wyświetlenia tekstu. Jak widać, domyślnie jest on bardzo ubogi i nie

obsługuje polskich "ogonków" (podobnie byłoby z chińskimi napisami i tak dalej). Zmieńmy to, dodając jedną linijkę w części <head> naszego dokumentu. Tak, to tutaj, gdzie dodajemy różne tzw. metadane, którymi opisujemy nasz dokument. Umieściliśmy tam już tytuł strony, pora teraz na skonfigurowanie kodowania znaków.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Oferta pracy: developer HTML i CSS</title>
  </head>
  <body>
    <h1>Poszukujemy developera HTML i CSS</h1>

    <img src="images/white-cat.jpg">
    <p>Dla naszego klienta, Kociej Fabryki, <strong>poszukujemy
zdolnego programisty</strong> stron internetowych w HTML i CSS.
Oferujemy przyzwoite zarobki, worek karmy i zabawek.</p>

    <p>Nie czekaj, napisz teraz! Nasza zwariowana ekipa czeka na
Ciebie!</p>
  </body>
</html>
```

Tym razem użyliśmy tagu <meta> z atrybutem charset (z ang. char - znak, set - zbiór) równym "utf-8". Oznacza to wybór systemu kodowania znaków o nazwie utf-8, który wspiera używanie lokalnych znaków diakrytycznych. Stosuj go zawsze na swoich stronach, ponieważ działa on również z innymi językami świata.

Uff! Wygląda na to, że to już koniec pracy nad tym przykładem. Właśnie zbudowałeś swoją pierwszą stronę internetową od podstaw. Brawo!

ROZDZIAŁ 4

Kodujemy artykuł na bloga

Pora na kolejne ćwiczenie, które powinno pokazać Ci filozofię pisania bardziej zaawansowanych stron internetowych w HTML5.

Spróbujmy uzyskać poniższy efekt. Uznajmy, że jest to artykuł na bloga.

Justin Bober: Odkąd nauczyłem się HTML-a, moje życie zmieniło się o 360 stopni

Napisał: Damian Wielgosik

Justin Bober wyznał coś, czego nie spodziewaliby się nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik title, jego życie stało się łatwiejsze. Ponoć prywatnymi mentorami Bobera stali się, jak donoszą osoby z otoczenia Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem jest walidator HTML.

Zadowolony kot Justina Bobera

Bober stworzył już kilka stron i nie zamierza na tym poprzestać. „Prawdopodobnie zaśpiewam o HTML-u na mojej następnej płycie" - dodaje artysta.

Zauważ, że mamy tutaj jeden nagłówek (tytułowy). Poza tym są informacje o autorze, a także zdjęcie z opisem oraz akapit z cytatem (ostatni). Warto dodać, że w przyszłości podczas swojej pracy jako front-end developer z reguły będziesz dostawał od grafika projekt strony w pliku graficznym i na jego podstawie będziesz musiał przenieść wszystko do HTML-a.

Zacznijmy od elementów, które zawsze są stałe i nie zmieniają się. Jak powiedzieliśmy sobie wcześniej, każda strona ma doctype, tag `<html>`, a w nim `<head>` i `<body>`. W `<head>` powinien z kolei znaleźć się kod, który ustawi odpowiednie kodowanie znaków diakrytycznych. A więc nasz startowy szablon wyglądał będzie tak:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
  </head>
  <body>
  </body>
</html>
```

Na pierwszy rzut oka brakuje jeszcze tytułu całej strony. Dodajmy go!

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
  </body>
</html>
```

W tym momencie mamy gotowy szablon, który możemy uzupełniać tekstem artykułu.

Jeśli zapiszemy taki plik jako article.html i potem otworzymy go w przeglądarce – nic nie zobaczymy, ponieważ pomiędzy `<body></body>` nie ma żadnej treści. Czas więc na jej analizę. Zaczynamy standardowo od najbardziej ogólnej części, wyróżniając stopniowo bardziej szczegółowe.

Do czynienia mamy z artykułem/postem na bloga. To pierwsza, bardzo istotna informacja. Następnie mamy część informacji nagłówkowych, jak to w artykułach bywa. Składa się ona z tytułu “Justin Bober: Odkąd nauczyłem się HTML-a, moje życie zmieniło się o 360 stopni" i informacji o autorze - “Napisał: Damian Wielgosik". Potem jest pierwszy akapit tekstu, miejsce na zdjęcie i jego opis oraz ostatni akapit, w którym zaznaczony jest cytat: “Prawdopodobnie zaśpiewam o HTML-u na mojej następnej płycie". To wszystko. Tego typu przeprowadzona analiza bardzo pomaga uzmysłowić sobie w głowie, jak będzie wyglądał kod HTML. Zaczynamy od najbardziej ogólnego elementu (główny rodzic), a następnie wyróżniamy jego dzieci, czyli wszystko, co się w nim zawiera (np. tytuł i informacje o autorze w nagłówku).

Ustaliliśmy, że mamy do czynienia z artykułem (a więc nazwanym materiałem tekstowym, posiadającym między innymi tytuł i kilku akapitów tekstu). Dodajmy tę informację do naszego kodu HTML:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
    </article>
  </body>
</html>
```

Jak zapewne zauważyłeś, użyliśmy elementu `<article>`, który oznacza właśnie taki materiał tekstowy w HTML. Artykuł składa się z nagłówka i akapitów stanowiących jego treść, a więc wszystkie tagi reprezentujące nagłówek i akapity znajdą się naturalnie w obrębie `<article>`.

Idąc dalej według listy elementów, które wyróżniliśmy, natrafiamy na sekcję złożoną z informacji nagłówkowych. A więc tytułu i danych autora. Tego typu sekcję w HTML5 oznaczamy poprzez tag `<header>` (ang. “header" to nagłówek). Grupuje on różne dane nagłówkowe danej sekcji (w tym przypadku jest to artykuł).

Dodajmy więc `<header>` do kodu. Przy okazji zauważ, że poruszamy się od góry do dołu, zagłębiając się (zauważ wcięcia) do poszczególnych elementów, tworząc swego rodzaju hierarchię.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
      </header>
    </article>
  </body>
</html>
```

Sekcja nagłówkowa gotowa, jednak nic w niej nie ma. Ustaliliśmy, że powinny znaleźć się tutaj tytuł i dane autora.

Tytuł oznaczymy tzw. nagłówkiem pierwszego stopnia, czyli najważniejszym. Będzie to tag `<h1>`. Jako że nie ma w HTML-u bardziej stosownego tagu do opisania informacji o autorze, używamy z kolei zwykłego `<p>`.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje
życie          zmieniło się o 360 stopni</h1>
        <p>Napisał: Damian Wielgosik</p>
      </header>
    </article>
  </body>
</html>
```

Przy okazji wiedz, że oprócz <h1> są jeszcze w HTML nagłówki niższych stopni:

- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

Elementy te odwzorowują logiczną strukturę tytułów i podtytułów. A więc jeśli mamy na przykład do czynienia z książką, to jej tytuł będzie zawarty pomiędzy elementem <h1>, czyli najważniejszym, głównym nagłówkiem. Następnie nazwy rozdziałów będą oznaczone tagiem <h2>, a na podrozdziałów - <h3>. Nie można na to spoglądać w ten sposób, że jeśli treść na stronie jest nieco mniej ważna, to ni stąd ni zowąd oznaczymy ją <h4> według naszego “widzimisię" - musimy zachować kolejność, a więc jeśli <h4> to wcześniej musi być gdzieś <h3>, dla którego <h4> to podtytuł. Wcześniej muszą wystąpić kolejno <h2> i <h1>. Na przykładzie tej książki mogłoby to wyglądać tak:

```
<h1>Podręcznik do matematyki</h1>
<h2>Liczby</h2>
<h3>Parzyste</h3>
<h2>Funkcje</h2>
<h3>Funkcja logarytmiczna</h3>
```

Idąc dalej, dodajmy pierwszy akapit tekstu. Aby to zrobić omijamy <header>, ponieważ logika nakazuje, aby treść samego artykułu była bezpośrednio częścią <article>, a nie nagłówka.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje życie
zmieniło się o 360 stopni</h1>
        <p>Napisał: Damian Wielgosik</p>
      </header>
      <p>Justin Bober wyznał coś, czego nie spodziewaliby się
nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody
rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik
title, jego życie stało się łatwiejsze. Ponoć prywatnymi
mentorami Bobera stali się, jak donoszą osoby z otoczenia
Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po
Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem
jest walidator HTML.</p>
    </article>
  </body>
</html>
```

Następnie czas na dodanie zdjęcia i jego opisu. Dla tego typu treści – a więc powiązanych tematycznie z całym dokumentem np. zdjęć, wykresów czy map, stworzono tag <figure>. Warto dodać, że warunkiem użycia jest to, że nie powinien mieć on wpływu

na znaczenie całego dokumentu, a więc równie dobrze zamieszczona w nim treść mogłaby być na oddzielnej stronie. Jako dziecko <figure> można użyć dodatkowo elementu <figcaption>, w którym umieścimy opis zdjęcia.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje życie
zmieniło się o 360 stopni</h1>
        <p>Napisał: Damian Wielgosik</p>
      </header>
      <p>Justin Bober wyznał coś, czego nie spodziewaliby się
nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody
rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik
title, jego życie stało się łatwiejsze. Ponoć prywatnymi
mentorami Bobera stali się, jak donoszą osoby z otoczenia
Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po
Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem
jest walidator HTML.</p>
      <figure>
        <img src="cat.jpg" alt="Kot Justina Bobera jest
zadowolony">
        <figcaption>Zadowolony kot Justina Bobera</figcaption>
      </figure>

    </article>
  </body>
</html>
```

Po uporaniu się z dodaniem obrazka, pozostało już nam tylko umieścić ostatni akapit na stronie z artykułem. Zauważ, że jego fragment – “Prawdopodobnie zaśpiewam o HTML-u na mojej następnej płycie" – został wyróżniony jako cytat. Należy więc stosownie

zaznaczyć to w naszym kodzie, aby całość nabrała sensu semantycznego. Być może w przyszłości ktoś będzie szukał cytatów z Justina Bobera i takie oznaczenie pomoże w szybszym znalezieniu tego cytatu. Inaczej wyszukiwarka miałaby do czynienia z jedną, wielką masą tekstu, z której ciężko byłoby wyłapać cytat.

Do oznaczania cytatów stworzono tag `<q>`

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje życie
zmieniło się o 360 stopni</h1>
        <p>Napisał: Damian Wielgosik</p>
      </header>
      <p>Justin Bober wyznał coś, czego nie spodziewaliby się
nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody
rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik
title, jego życie stało się łatwiejsze. Ponoć prywatnymi
mentorami Bobera stali się, jak donoszą osoby z otoczenia
Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po
Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem
jest walidator HTML.</p>
      <figure>
        <img src="cat.jpg" alt="Kot Justina Bobera jest
zadowolony">
        <figcaption>Zadowolony kot Justina Bobera</figcaption>
      </figure>
      <p>Bober stworzył już kilka  stron i nie zamierza na tym
poprzestać. <q>Prawdopodobnie zaśpiewam o HTML-u na mojej
następnej płycie</q> - dodaje artysta.
      </p>
    </article>
  </body>
</html>
```

Zapisz teraz cały kod do pliku z rozszerzeniem .html i wyświetl go w przeglądarce. Właśnie stworzyłeś swoją drugą stronę. Świetnie! To kolejny krok do zostania profesjonalnym twórcą stron internetowych.

ROZDZIAŁ 5

Zmień wygląd strony dzięki CSS3

Zapewne zauważyłeś, że oprócz zdjęć nasza strona nie wygląda zbyt interesująco. Czarny tekst na białym tle nie będzie zachęcający dla odwiedzających. Popracujmy więc nad wyglądem dzięki CSS (Cascading Style Sheets). Używając tego języka możemy zmienić na naszych stronach między innymi kolory i wielkości fontów. A to tylko początek długiej listy możliwości.

O ile HTML służy do opisania treści na stronie i podzieleniu jej na fragmenty według ich znaczenia, tak CSS odpowiada za wygląd naszych stron internetowych. Kod CSS można umieścić w oddzielnym pliku z rozszerzeniem .css i "wstawić" go poprzez specjalny tag HTML. Można też umieścić go bezpośrednio w dokumencie HTML. Nim pomówimy o różnych strategiach dołączenia "CSSów" do naszych stron, wyjaśnijmy najpierw na czym polega filozofia tego języka.

Załóżmy na chwilę, że składamy normalny dom, do którego musimy wybrać elementy, takie jak okna, drzwi, dach, ściany, rynny i tak dalej. Należy kupić w sklepie okna o różnej wielkości, ustalić, gdzie konkretnie będą zainstalowane drzwi, a gdzie rynny, pomalować niezbędne części. Gdybyś robił to w CSS, jedno z wielu rozwiązań tego zadania mogłoby wyglądać tak:

```
dach {
  background-color: green;
}

drzwi {
  background-color: yellow;
  width: 100px;
  height: 300px;
}

okno {
  border: 5px solid brown;
  width: 150px;
}
```

Analizując ten kod od góry do dołu (zauważ, że to już pewna reguła w czytaniu kodu, która nota bene nie dotyczy tylko HTML i CSS) napotykamy na poniższy fragment:

```
dach {
  background-color: green;
}
```

Przekładając ten przykład na ludzki język, wybraliśmy wszystkie elementy o nazwie "dach" i ustawiliśmy dla nich kolor tła ("background-color") na zielony ("green"). Reszta kodu wygląda podobnie. Na przykład okna:

```
okno {
  border: 5px solid brown;
  width: 150px;
}
```

Taki kod mówi: "dla wszystkich okien ustaw, co następuję: ramkę (border) o szerokości 5 pikseli (5px), zaznaczona ciągłą linią (solid) o kolorze brązowym (brown). Ponadto niech znalezione okna będą szerokości (width) 150 pikseli (150px). Zauważ, że zamiast pisać "tło: czerwone;", masz tutaj do czynienia z poleceniami w języku angielskim. Jak już

pewnie wywnioskowałeś, jest to w tworzeniu stron i w ogóle w programowaniu kolejna reguła, której bezwzględnie należy się trzymać - używaj wszędzie języka angielskiego. HTML i CSS na szczęście to wymuszają ze względu na swoją składnię.

Jak zapewne zauważyłeś, jest w tym wszystkim pewien powtarzający się schemat. Najpierw podajemy nazwę elementu, a potem definiujemy dla niego pewne cechy wyglądu, które umieszczamy pomiędzy nawiasami klamrowymi ("{" oraz "}"). A więc używając języka opisowego, wygląda to mniej więcej tak:

nazwa elementu { nazwa_wlasciwosci: wartosc wlasciwosci wyglądu; }

Tego typu konstrukcja nazywa się fachowo regułą CSS. Reguła składa się z kolei z selektora (wszystko, co jest przed pierwszą klamrą) i listy właściwości, które piszesz pomiędzy nawiasami klamrowymi.

Istnieją też różne sposoby, by nasze wyszukiwanie było jeszcze bardziej konkretne. Powiedzmy, że interesują nas okna tylko na parterze. Co wtedy? Moglibyśmy napisać coś podobnego:

```
parter okno {
  border: 5px solid brown;
  width: 150px;
}
```

W zasadzie zmienił się tylko selektor - zamiast:

```
okno {
```

Mamy:

```
parter okno {
```

Taki kod czytamy od lewej następująco: "znajdź parter, a następnie poszukaj w nim okien i ustaw tym oknom następujące własności". Gdybyśmy mieli do czynienia z takim selektorem:

```
parter sciana okno {
```

Wtedy przeczytalibyśmy to tak: "znajdź parter, a w nim ściany, a w nich okna i oknom ustaw, co następuje". I tak dalej. Jeśli pamiętasz analogię, w której mówiliśmy o zagnieżdżonych tagach HTML jako o dzieciach i rodzicach, tutaj jest podobnie. Szukamy elementów, które znajdują się w innych elementach, a więc szukamy "dzieci rodziców".

Używając przeglądarki nie można niestety wybudować domu, a więc nasz powyższy, abstrakcyjny przykład niewiele nam powie, bo nie będziemy mogli tego zobaczyć na własne oczy. Gdyby jednak poszczególne części domu zastąpić tagami HTML? I tak, zamiast okien możemy szukać akapitów (<p>), a zamiast drzwi nagłówków pierwszego stopnia <h1>? Wyglądałoby to tak:

```
p {
}

h1 {
}
```

Tego typu kod nie zmieni jednak nic w wyświetlaniu strony, ponieważ pomiędzy nawiasami klamrowymi nie umieściliśmy żadnych właściwości - tła, wielkości i tak dalej. Możemy je dodać, jednak bardzo miło byłoby pracować z żywym organizmem, który wymaga naszej pomocy. Co Ty na to, by użyć strony z artykułem o Justinie Boberze i dodać tam nieco kolorów i życia?

Przypomnijmy, że kod wyglądał tak:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Justin Bober zafascynowany HTML-em</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje życie
zmieniło się o 360 stopni</h1>
        <p>Napisał: Damian Wielgosik</p>
      </header>
      <p>Justin Bober wyznał coś, czego nie spodziewaliby się
nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody
rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik
title, jego życie stało się łatwiejsze. Ponoć prywatnymi
mentorami Bobera stali się, jak donoszą osoby z otoczenia
Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po
Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem
jest walidator HTML.</p>
      <figure>
        <img src="cat.jpg" alt="Kot Justina Bobera jest
zadowolony">
        <figcaption>Zadowolony kot Justina Bobera</figcaption>
      </figure>
      <p>Bober stworzył już kilka  stron i nie zamierza na tym
poprzestać. <q>Prawdopodobnie zaśpiewam o HTML-u na mojej
następnej płycie</q> - dodaje artysta.
      </p>
    </article>
  </body>
</html>
```

Przywołując na chwilę analogię budowy domu, tutaj – zamiast drzwi, okien i podobnych – mamy z kolei do czynienia z elementami takimi jak <article>, <p>, <header>, <body>, <figcaption> i tak dalej. To z tych tagów zbudowaliśmy tę stronę. To bardzo ważne porównanie, gdy będziemy próbować uzmysłowić sobie, jak działa CSS.

Przy okazji przygotowałem dla Ciebie zrzut z ekranu, ze wskazówką, jak powinna wyglądać strona po naszych modyfikacjach:

Justin Bober: Odkąd nauczyłem się HTML-a, moje życie zmieniło się o 360 stopni

Napisał: Damian Wielgosik

Justin Bober wyznał coś, czego nie spodziewaliby się nawet najwięksi fani tego zdolnego muzyka i eseisty. Młody rockendrolowiec przyznał, że odkąd postawił pierwszy znacznik title, jego życie stało się łatwiejsze. Ponoć prywatnymi mentorami Bobera stali się, jak donoszą osoby z otoczenia Kanadyjczyka, Ryan Losling i Nicolas Klatka. Często spacerują po Los Angeles i rozprawiają godzinami o tym, jak fajnym narzędziem jest walidator HTML.

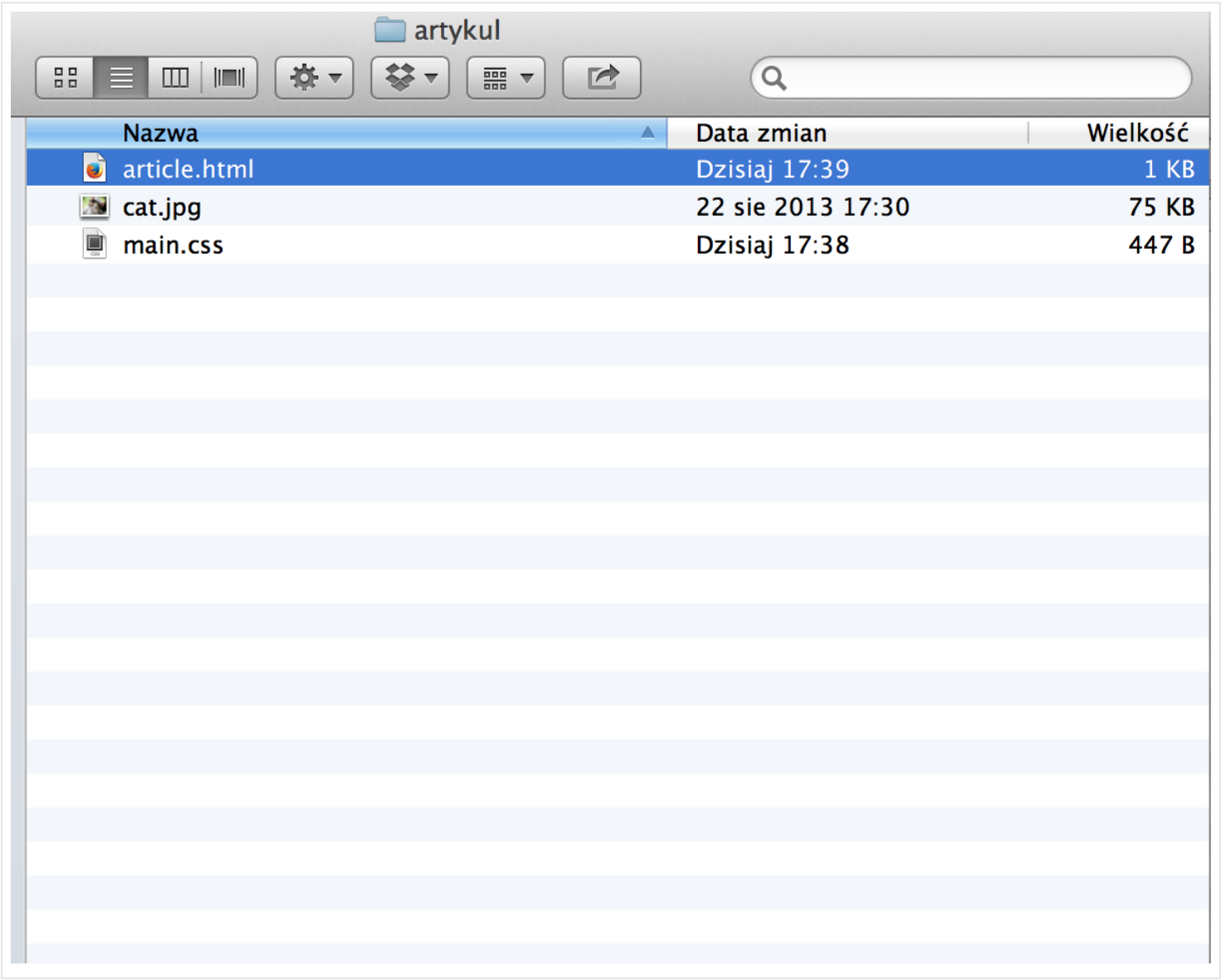
Zadowolony kot Justina Bobera

Bober stworzył już kilka stron i nie zamierza na tym poprzestać. „Prawdopodobnie zaśpiewam o HTML-u na mojej następnej płycie" - dodaje artysta.

Jak widać, trzeba będzie nanieść trochę zmian - dodać proste kolory, tła, krój fonta i tak dalej. Odtwórzmy więc powyższą stronę krok po kroku!

Pierwsza czynność to oczywiście zapisanie całego kodu HTML do oddzielnego pliku. U mnie nazywa się on article.html. Następnie należy stworzyć plik, w którym przechowywać będziemy nasze reguły CSS. Niech to będzie main.css.

Tak prezentuje się to na moim komputerze:



Otwieram więc article.html w przeglądarce, a plik main.css za pomocą edytora tekstu - polecam tutaj Sublime Text Editor 2 lub TextMate. Po każdej zmianie w main.css będę odświeżał stronę, aby monitorować postępy, jeśli chodzi o jej wygląd. Muszę jedynie dodać do article.html polecenie, aby przeglądarka wczytywała również kod z main.css i

stosowała się do tamtejszych reguł CSS. Robi się to w <head> dzięki tagowi <link>. Wystarczy dodać coś takiego:

```
<link rel="stylesheet" href="main.css">
```

Atrybut "href" wskazuje, w którym pliku mieści się kod CSS. "stylesheet" mówi nam z kolei, że chodzi o arkusz stylów CSS.

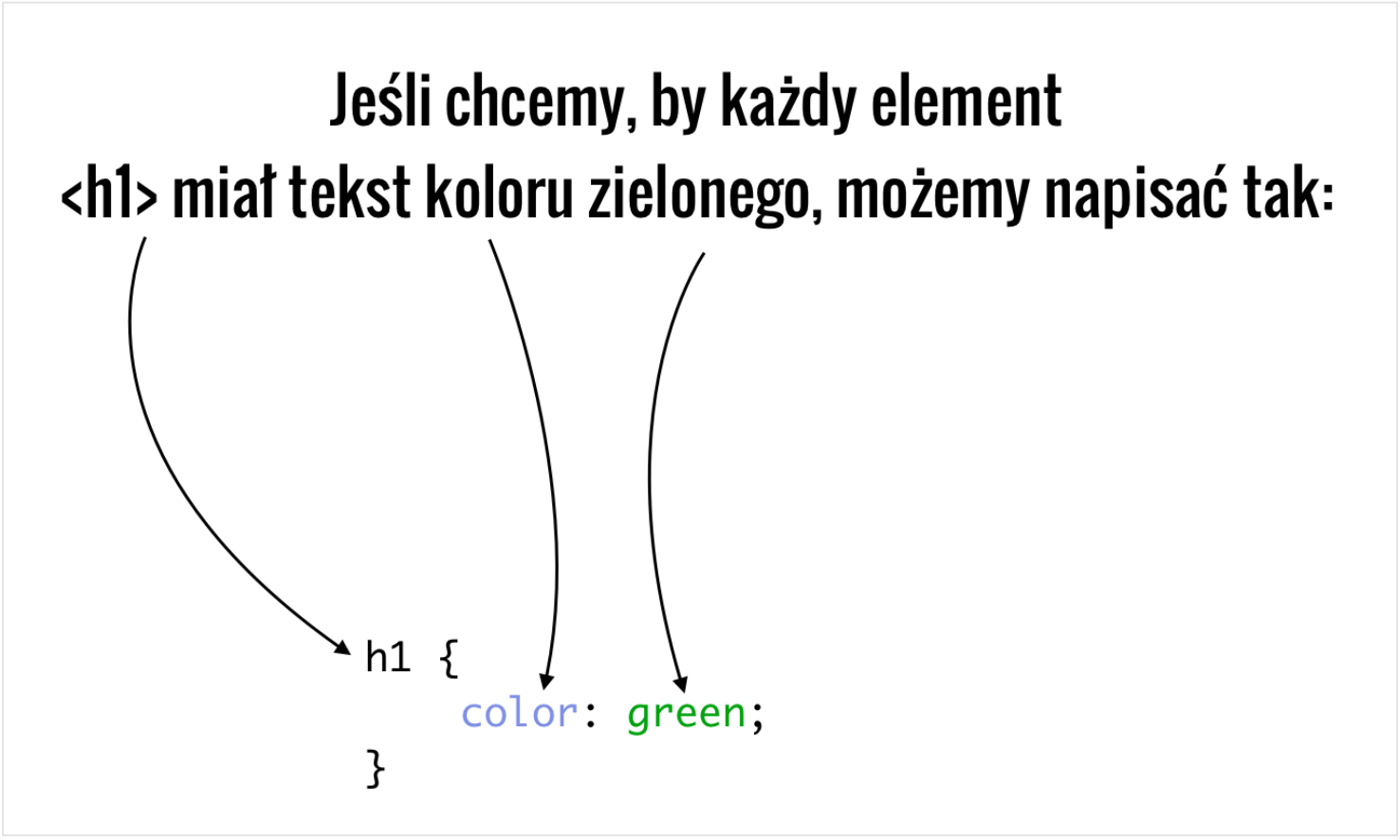
Na początek zajmijmy się nagłówkiem pierwszego stopnia, czyli tytułem artykułu. Analogicznie do okien i ścian, szukamy takiego elementu w CSS:

```
h1 {
}
```

Brawo! Właśnie poprzez kod CSS powiedzieliśmy przeglądarce, by dla wszystkich elementów <h1> w HTML, zaaplikowała zdefiniowany przez nas wygląd. Problem w tym, że nie ustaliliśmy jaki, ponieważ pomiędzy nawiasami klamrowymi nie ma żadnego kodu. Jako że chcemy, by tekst tytułowy był koloru zielonego, zastosujmy specjalnie do tego stworzoną właściwość "color" ustawioną na "green" (czyli zielony po polsku):

```
h1 {
  color: green;
}
```

Działanie tej reguły wytłumaczyłem jeszcze w postaci poniższego schematu:



Sprawdźmy teraz, jak wygląda nasza strona po zapisaniu zmian.

Tak jest! Tytuł został oznaczony kolorem zielonym! Pora teraz zająć się danymi o autorze. Tekst ma tutaj kolor biały, a tło jest czerwone. Znajdźmy ten fragment w kodzie HTML. Jak widać, informacja o autorze zawarta jest między tagiem `<p>`:

```
<article>
  <header>
    <h1>Justin Bober: Odkąd nauczyłem się HTML-a, moje życie
zmieniło się o 360 stopni</h1>
    <p>Napisał: Damian Wielgosik</p>
  </header>
```

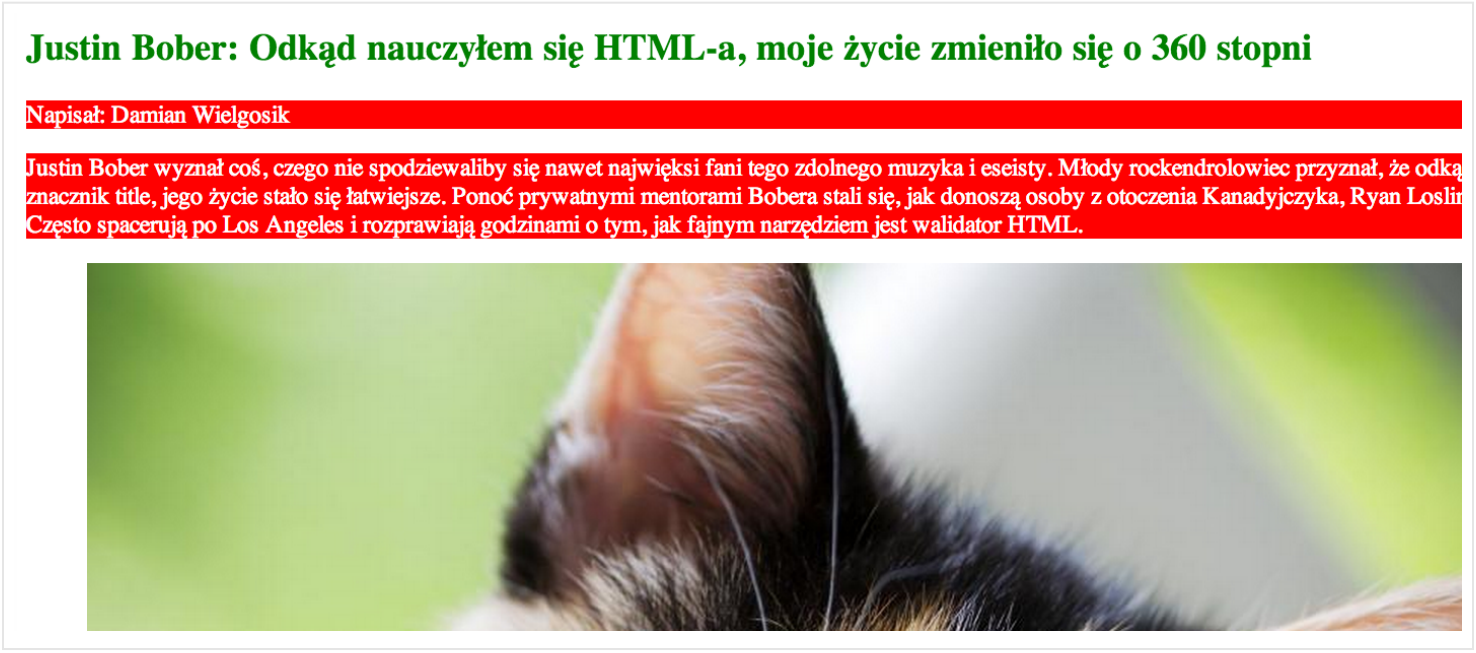
A więc przekładając to na język CSS, powinniśmy znaleźć wszystkie tagi p i nadać im biały kolor tekstu oraz czerwony kolor tła. Dodajmy więc stosowną linijkę do pliku main.css:

```
p {
    background-color: red;
    color: white;
}
```

Cała zawartość main.css wygląda w tej chwili tak:

```
h1 {
  color: green;
}

p {
    background-color: red;
    color: white;
}
```

Jak widzisz, dodajemy reguły jedna pod drugą. Czas sprawdzić, jak wygląda teraz nasza strona:



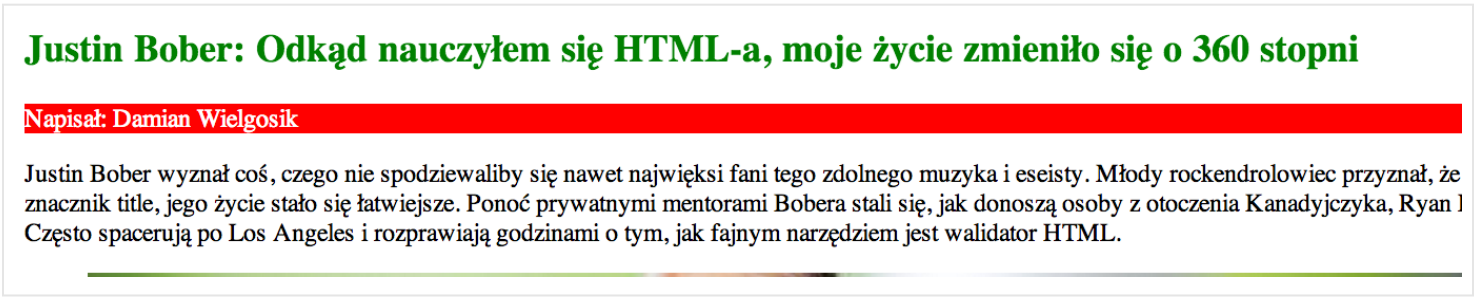
Nie jest do końca tak, jak to sobie wyobrażaliśmy. Okazało się, że wszystkie inne akapity również zostały uwzględnione i teraz są wyświetlone na czerwonym tle. To nasza wina, ponieważ używając tego kodu:

```
p {
    background-color: red;
    color: white;
}
```

"powiedzieliśmy" przeglądarce mniej więcej tak: znajdź na stronie **wszystkie** elementy `<p>` i ustaw im czerwone tło, a kolor fontu biały. Nam z kolei zależy, by **tylko** jeden akapit miał takie właściwości, konkretnie ten, który znajduje się w nagłówku artykułu. Musimy więc zmodyfikować tak powyższy selektor, by znaleźć paragraf, który jest "dzieckiem" `<header>`, a ten z kolei niech będzie "dzieckiem" `<article>`:

```
article header p {
    background-color: red;
    color: white;
}
```

Sprawdźmy efekt tych zmian:



Znacznie lepiej! Wybrano właściwy akapit. Jak to się stało? Otóż przeglądarka przeczytała powyższy selektor CSS, aby wiedzieć, do których tagów zastosować zmianę wyglądu. Potem przeanalizowała kod HTML i znalazła wszystkie tagi pasujące do selektora, a więc znalazła `<article>` (u nas akurat jest tylko jeden), potem poszukała zagnieżdżonych w nich tagów <header>, a na końcu znalazła umieszczone w elementach `<header>` tagi `<p>` i zaaplikowała im wygląd, który zdefiniowaliśmy. CSS właśnie działa w ten sposób – w selektorach definiujesz jakich tagów chcesz "poszukać", odnosząc się do nich m.in. według hierarchii, jakiej użyłeś z kolei w kodzie HTML.

Zajmijmy się teraz zdjęciem do artykułu. Uznajmy, że załączniki do artykułu powinny mieć 600 pikseli szerokości. Jako że odpowiednikiem załącznika w naszym kodzie HTML jest tag `<figure>`, nadajmy mu szerokość 600px:

```
article figure  {
    width: 600px;
}
```

A więc każdy element `<figure>` w `<article>` będzie miał szerokość 600 pikseli. U nas jest tylko jeden, ale równie dobrze moglibyśmy mieć kilka zdjęć do artykułu. Następnie musimy ustawić obramowanie:

```
article figure  {
    width: 600px;
    border: 3px solid black;
}
```

Użyliśmy tutaj właściwości o nazwie "border". Po dwukropku podajemy szerokość obramowania, następnie styl obramowania, czyli "solid" (linia ciągła) i jego kolor - "black" (czarny). Zobaczmy, jak wygląda teraz nasz artykuł:

Zadowolony kot Justina Bobera

Niestety mamy problem. O ile obramowanie wyświetla się dobrze i kontener <figure> jest szeroki na 600 pikseli, tak obrazek w <img> wystaje poza obramowanie, a więc i poza <figure>. Dzieje się tak ponieważ ustaliliśmy szerokość dla elementu <figure>, jednak obrazek (w tagu <img>) nie ma ustalonej żadnej szerokości, a więc wyświetla się w oryginalnych rozmiarach. Fajnie byłoby, gdyby przyjął 100% szerokości swojego rodzica, czyli tagu <figure>. Robimy to w prosty sposób:

```
article figure img {
    width: 100%;
}
```

Wygląda to teraz tak:

Zadowolony kot Justina Bobera

Dla lepszego efektu wizualnego, dobrze byłoby dodać odstęp od krawędzi dla wszystkich elementów w <figure>. Robimy to dzięki właściwości "padding". Zmodyfikujmy więc selektor "article figure":

```
article figure  {
    width: 600px;
    border: 3px solid black;
    padding: 5px;
}
```

Efekt naszych zmian widać na obrazku poniżej:

Zadowolony kot Justina Bobera

Spróbuj zmodyfikować wartość "paddingu". Zobaczysz, jak zmienia się biała przerwa między obrazkiem na ramką.

Wygląda na to, że prawie udało nam się osiągnąć zamierzony cel. Dlaczego prawie? Zauważ, że tekst akapitów rozciąga się na całą szerokość okna przeglądarki. Może wypadałoby go jakoś ograniczyć z góry zdefiniowaną szerokością? Na przykład 800 pikseli. Wybierzmy więc specjalnym selektorem CSS sam artykuł i nadajmy mu szerokość 800px:

```
article {
    width: 800px;
}
```

Od razu lepiej!

Jeśli przyjrzysz się jeszcze początkowemu obrazkowi to zobaczysz, że mamy do dyspozycji tam nieco inny krój fonta. Tak jak w programach do edycji tekstu typu Microsoft Word możesz zmienić krój fonta, tak i w CSS da się to zrobić. Najlepiej ustawić globalny typ fonta dla wszystkich elementów. W CSS robi się to podając właściwość fonta dla tagu `<body>`. Wtedy wszystkie elementy "dzieci" zawarte w `<body>` zostaną również objęte tymi ustawieniami. W naszym przykładzie zastosowałem font o nazwie Verdana. Użyjmy go:

```
body {
    font-family: Verdana;
}
```

Możesz usunąć tę linijkę, by zobaczyć różnicę. Rzeczywiście. Dla nagłówka, jak i dla normalnych akapitów zmienił się krój fonta.

Ostatecznie nasz kod w main.css wygląda tak:

```
body {
    font-family: Verdana;
}

article {
    width: 800px;
}

article header h1 {
    color: green;
}

article header p {
    background-color: red;
    color: white;
}

article figure  {
    width: 600px;
    border: 3px solid black;
    padding: 5px;
}

article figure img {
    width: 100%;
}
```

Dobrym zwyczajem jest, by rozpoczynać od najogólniejszych selektorów, dlatego zacząłem od body, potem article i dalej, idąc od góry do dołu, umieściłem coraz bardziej szczegółowe i zagnieżdżone.

ROZDZIAŁ 6

Projektujemy menu

Jak zapewne zauważyłeś, w naszym kodzie CSS użyliśmy samych nazw tagów HTML. W <article> na przykład szukaliśmy <figure>:

```
article figure  { ... }
```

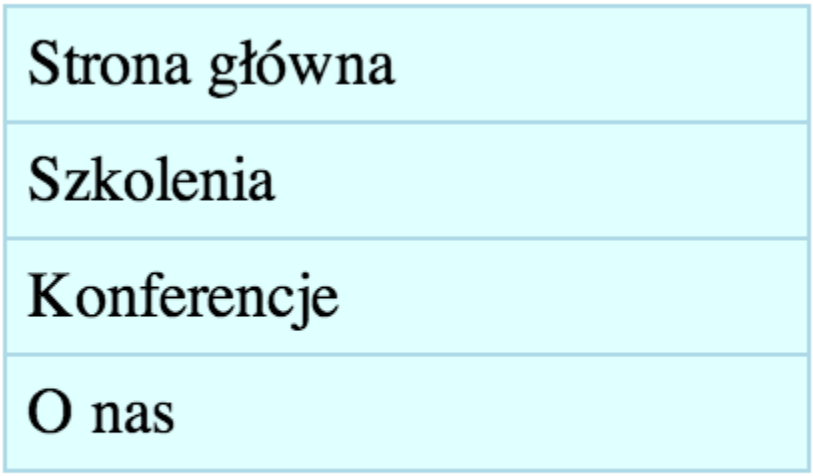
W CSS istnieją jednak inne metody, by odnieść się do poszczególnych elementów. Jedną z nich poznamy na przykładzie menu na stronie. Uznajmy, że dysponujemy takim oto kodem HTML, który należy "ostylować" używając kodu CSS w pliku main.css.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
      <meta charset="utf-8">
      <title>Menu</title>
      <link rel="stylesheet" href="main.css" media="screen">
  </head>
  <body>
    <nav>
      <ul>
        <li>
          <a href="index.html">Strona główna</a>
        </li>
        <li>
          <a href="szkolenia.html">Szkolenia</a>
        </li>
        <li>
          <a href="konferencje.html">Konferencje</a>
        </li>
        <li>
          <a href="onas.html">O nas</a>
        </li>
      </ul>
    </nav>
  </body>
</html>
```

Będzie to strona tylko i wyłącznie z samym menu, które będzie składać się z czterech pozycji:

- Strona główna
- Szkolenia
- Konferencje
- O nas

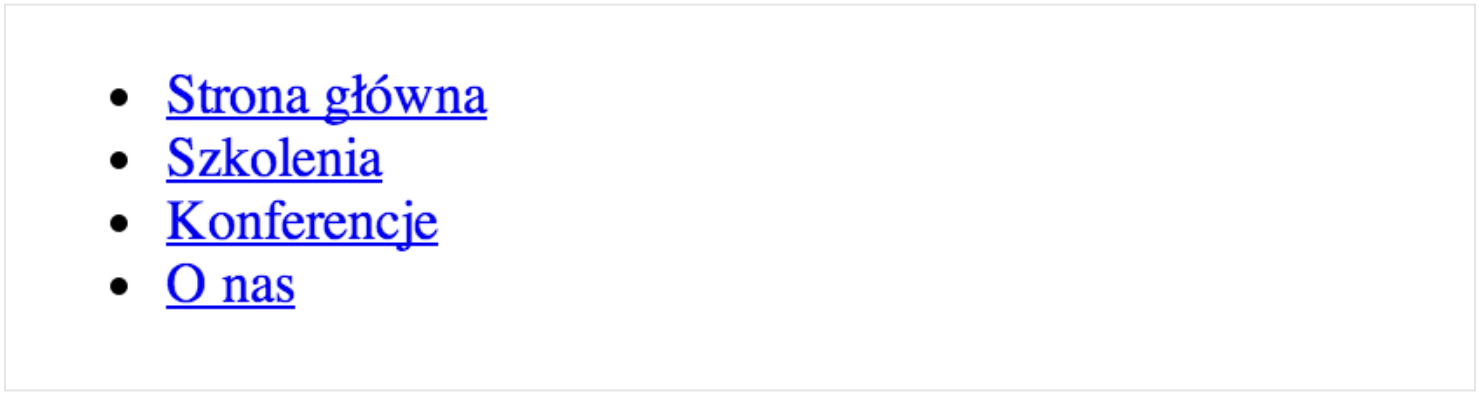
Docelowo ma to wyglądać tak:



Zapytasz zapewne, dlaczego w kodzie odpowiadającym za menu użyłem elementu <nav> i <ul> oraz <li>. Otóż <nav> służy do oznaczania wszelkiego rodzaju nawigacji na stronach internetowych, które zawierają linki do wewnętrznych lub zewnętrznych stron. A więc umieszczając <nav> w kodzie mówimy tak: wszystko, co będzie wewnątrz <nav> będzie służyło do nawigowania na stronie.

W <nav> z kolei umieściłem <ul> i kilka elementów <li>. Otóż <ul> oznacza nieuporządkowaną listę elementów, a więc taką w której kolejność jest nieistotna. Z kolei <li> oznacza pojedynczy element takiej listy. W świecie tworzenia stron internetowych utarło się, że – biorąc pod uwagę wszystkie dostępne tagi HTML – lista nieuporządkowana wydaje się najrozsądniejszym wyborem, jeśli chodzi o odwzorowanie menu na stronach. Tak naprawdę menu jest swego rodzaju listą linków, która została stworzona bez jakiejś z góry określonej reguły co do kolejności jej elementów.

Bez gotowego pliku stylów main.css, nasza nieuporządkowana lista wyświetla się domyślnie tak:



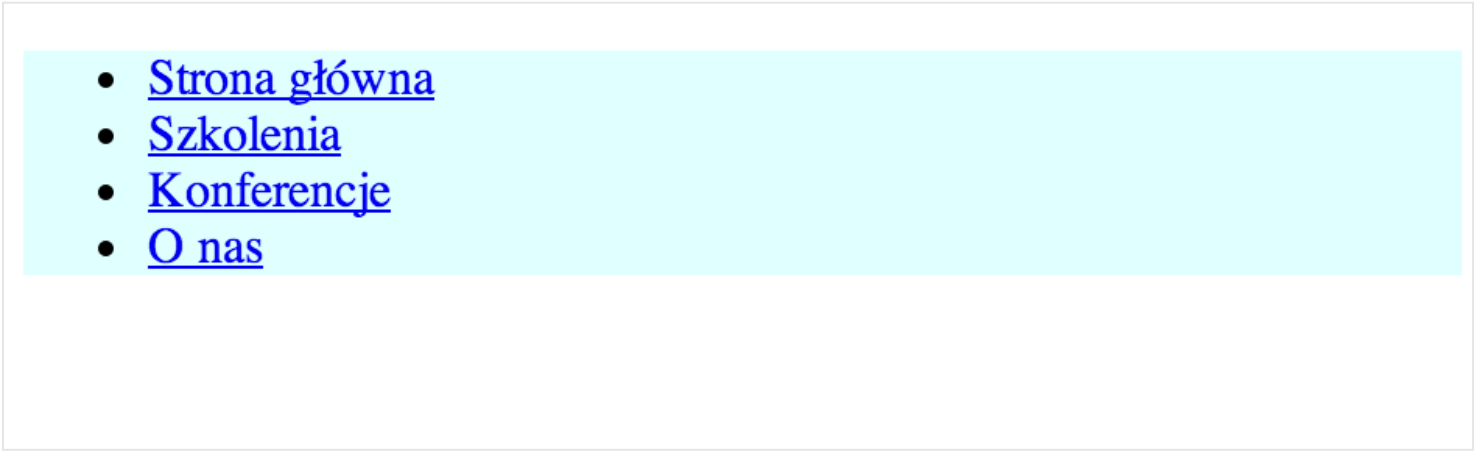
Zapewne wielokrotnie widziałeś już podobną, choćby podczas tworzenia plików tekstowych, kiedy chciałeś wymienić kilka elementów w postaci listy punktowej. Bez kodu CSS lista <ul> wygląda podobnie - każdy jej element rozpoczyna się od kropki. Natomiast nasze menu jest nieco inne. Ma obramowanie, co więcej, poszczególne jego elementy również. Każdy element listy ma też tło, a linki w tym menu są koloru czarnego (domyślnie w HTML każdy link wyświetlany jest niebieskim kolorem, co widać na załączonym obrazku). Spróbujmy odwzorować to wszystko w naszym kodzie CSS.

Standardowo, zaczynamy od najbardziej ogólnego, zewnętrznego tagu w kodzie HTML. Jest nim <nav>. Jak powiedzieliśmy, <nav> jest elementem, który mówi, że w jego środku zawiera się wszelki kod odpowiedzialny za nawigację. A więc oprócz semantycznego znaczenia, nie ma tutaj dla nas za wiele pracy, ponieważ element ten jako taki nie wymaga w obecnej sytuacji zmian w wyglądzie (tym bardziej, że domyślnie przeglądarka nie aplikuje mu żadnych stylów).

Następnie mamy tag <ul>, który oznacza listę nieuporządkowaną. Chcemy, by nasza lista była wyświetlana nieco inaczej, niż domyślnie. Najważniejsze, by miała tło:

```
nav ul {
  background-color: LightCyan;
}
```

Dla tła wybraliśmy kolor o nazwie LightCyan. Spójrzmy, czy coś się zmieniło, dzięki naszym poprawkom:



Rzeczywiście! Udało się zaaplikować kolor cyjan jako tło elementu <ul>. Wszystko dlatego, że w <nav> poszukaliśmy tagów <ul> i zaaplikowaliśmy im tło. Wyraża to poniższy selektor, jak zapewne pamiętasz:

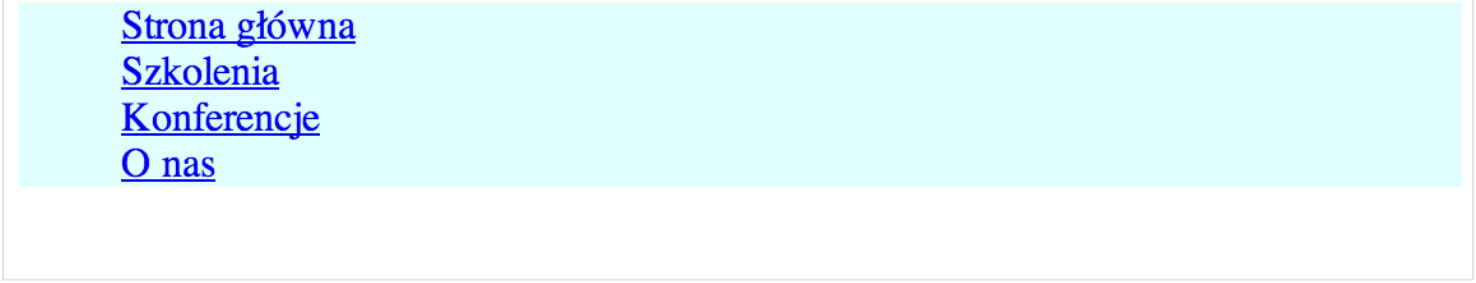
```
nav ul {}
```

Niestety, nadal widoczne są czarne, okrągłe punkty. Możemy je schować, dzięki właściwości "list-style":

```
nav ul {
  background-color: LightCyan;
  list-style: none;
}
```

Ustawiona na "none" sprawia, że lista nie będzie miała żadnych znaków wyróżniających poszczególne punkty.

Wygląda to już nieco lepiej:



Dziwi jednak duży odstęp od lewej. Wyzerujmy go, stosując znany nam z wcześniejszego ćwiczenia "padding":

```
nav ul {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
}
```

Jak widać, był to dobry pomysł. Powoli zbliżamy się do ideału:

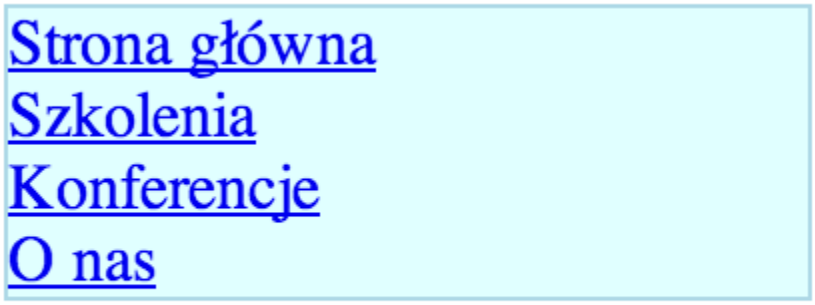
Teraz pora popracować nad wymiarami. Nasza nawigacja ma być szeroka na 200 pikseli:

```
nav ul {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
  width: 200px;
}
```

Na sam koniec dodajmy obramowanie listy. Niech będzie ono wyrażone linią ciągłą o szerokości 1 piksel. Kolor? Wybierzmy "LightBlue":

```
nav ul {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
  width: 200px;
  border: 1px solid LightBlue;
}
```

Wynik tej zmiany jest cokolwiek zadowalający:



Pojawiła się ramka – pięknie. Pora teraz zająć się elementami listy, do których możemy się odnieść używając poniższego selektora CSS:

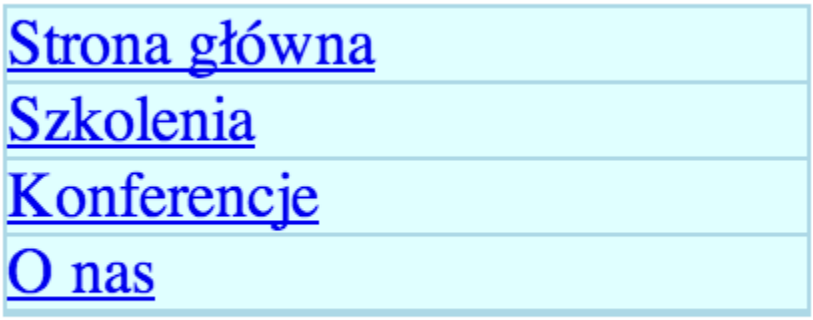
```
nav ul li {}
```

A więc w <nav> poszukamy <ul>, a w <ul> wszystkich elementów <li> i im nadamy jakieś własności. Co rzuca się najbardziej w oczy to brak obramowań dolnych dla każdego elementu listy, więc dodajmy je:

```
nav ul li {
  border-bottom: 1px solid LightBlue;
}
```

W tym celu użyliśmy właściwości border-bottom, a więc z angielskiego można to przetłumaczyć właśnie jako obramowanie (border) dolne (bottom).

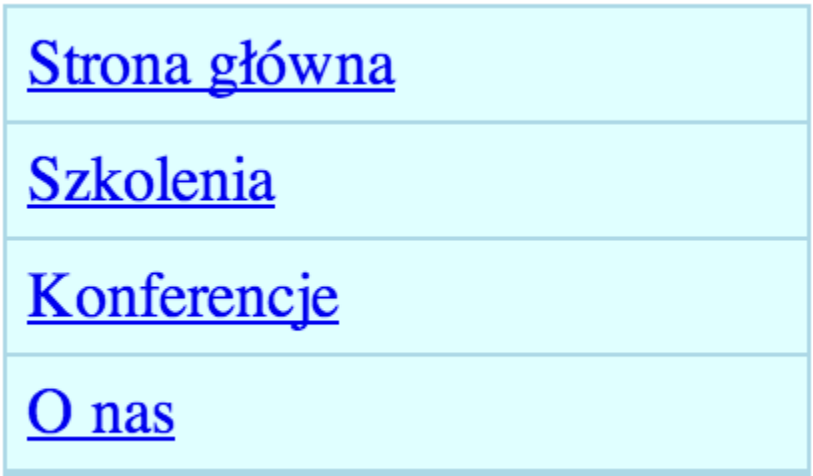
Obecnie, nasze menu wygląda tak:



Uwidoczniły się przy okazji niestety dwa problemy. Pierwszy z nich to bardzo mało przestrzeni pomiędzy poszczególnymi elementami listy. Zmieńmy to, wykorzystując stary, dobry "padding":

```
nav ul li {
  border-bottom: 1px solid LightBlue;
  padding: 5px;
}
```

Jest znacznie lepiej, prawda?



Drugim problemem jest dziwna, podwójna linia na samym dole naszego menu. To dlatego, że "naszły" na siebie dwa obramowania - jedno ustawione dla <ul>, a drugie dla <li> (w tym przypadku dla ostatniego <li> - "O nas").

Można łatwo to obejść, przy okazji poznając kolejny, pomocny element, którego możemy użyć tworząc nasze selektory w CSS. Powiedzieliśmy sobie bowiem, że obramowanie dolne ostatniego elementu w liście (a więc ostatniego tagu <li>) pokrywa się z ogólnym obramowaniem, jakie założyliśmy dla <ul> tutaj:

```
nav ul {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
  width: 200px;
  border: 1px solid LightBlue;
}
```

Pamiętasz zapewne, że w CSS możemy resetować różne wartości. Zrobiliśmy tak na przykład z graficznym wyróżnieniem elementu w liście:

```
list-style: none;
```

Zróbmy to samo, tyle że używając właściwości border-bottom, którą "zresetujemy" i ustawimy wartość "none". Jedyna różnica jest taka, że tym razem wybierzemy poprzez selektor tylko ostatni element <li>:

```
nav ul li:last-child {
  border-bottom: none;
}
```

Efekt jest jak najbardziej zadowalający:

Podwójne obramowanie znikło, a wszystko dlatego, że poszukaliśmy ostatni element <li> w <ul> i wyłączyliśmy mu obramowanie dolne. Stało się to dzięki selektorowi tzw. pseudoklasy o nazwie "last-child":

```
nav ul li:last-child {}
```

Powyższy selektor można przetłumaczyć następująco:

"Poszukaj elementu <nav>, a w nim elementów <ul>, z kolei w elementach <ul> znajdź ostatni (a więc "last-child") element typu <li> i jemu przypisz poniższe wartości". Warto zaznaczyć, że wszelkie selektory pseudoklas używamy poprzez dwukropek i nazwę selektora.

Ostatnia rzecz, jaką musimy zrobić to dostosowanie tekstu w linkach. Otóż domyślnie linki tworzymy w HTML jak poniżej:

```
<a href="adres">tekst, po kliknięciu którego przeniesiemy się do podanego w href adresu</a>
```

A więc używamy tagu <a> z atrybutem href. Jako wartość tego atrybutu podajemy adres, do którego chcemy przenieść użytkownika, jeśli kliknie on w nasz link. W co kliknie

zależy od tego, jaką treść umieścimy między `<a>` i `</a>`. W naszym przykładzie mamy cztery linki, jeden z nich wygląda następująco w kodzie HTML:

```
<a href="szkolenia.html">Szkolenia</a>
```

A więc w przeglądarce ujrzymy napis "Szkolenia", który możemy najechać kursorem myszki i kliknąć. Zostaniemy wtedy przeniesieni do strony, która została zapisana w pliku "szkolenia.html".

Znając już to, jaki tag odpowiada za umieszczanie linków w kodzie HTML, możemy stworzyć specjalny selektor CSS, który ich "poszuka". Wtedy z łatwością umieścimy nowe deklaracje dotyczące wyglądu:

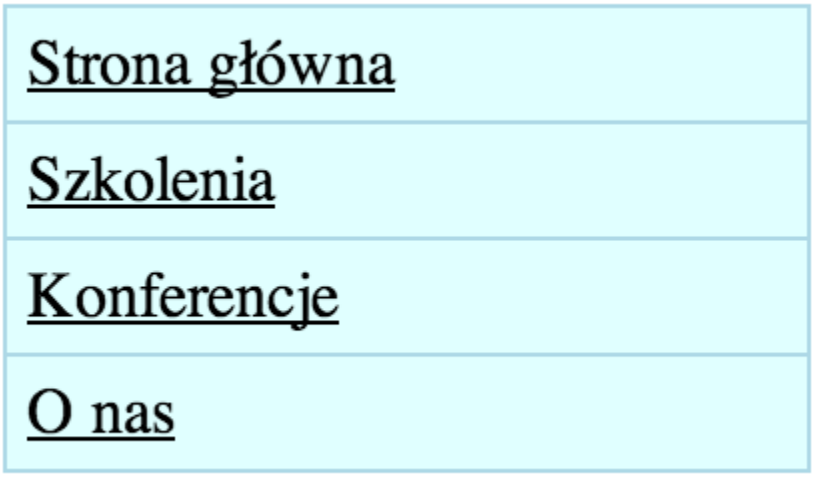
```
nav ul li a {}
```

Voilà!

Uzupełnijmy teraz ten selektor o nowe właściwości. Przede wszystkim zmieńmy kolor fonta na czarny:

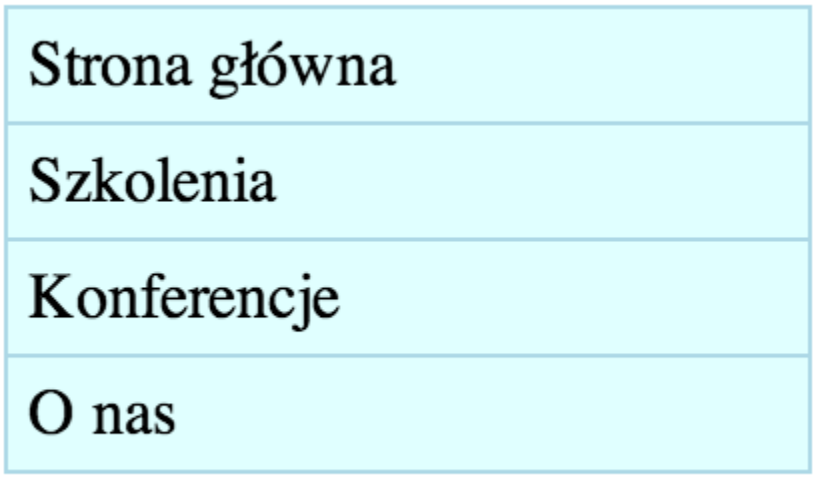
```
nav ul li a {
  color: black;
}
```

Podejrzyjmy od razu w przeglądarce, co się zmieniło:



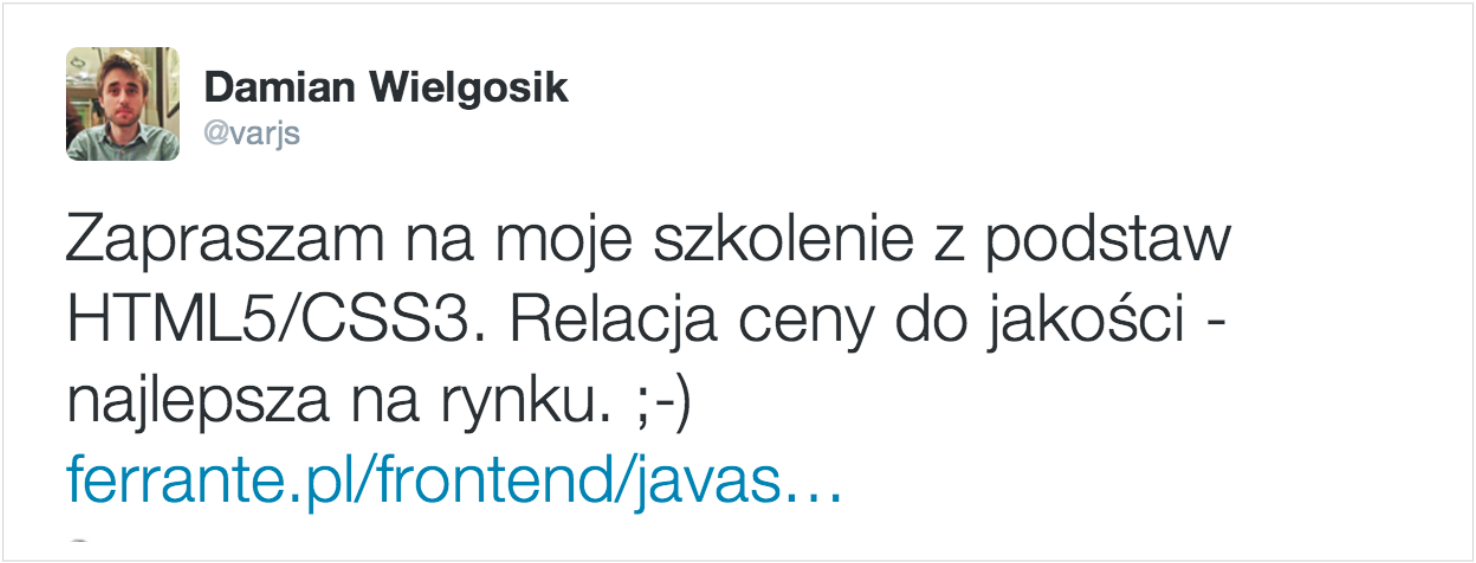
Świetnie, mamy czarny kolor linków. Pora teraz zrobić coś z podkreśleniem. Otóż musisz wiedzieć, że domyślnie, podobnie jak z odstępem od lewej strony w przypadku elementów listy, przeglądarka ustawia linkom podkreślenie w postaci "text-decoration: underline" w CSS. Musimy zresetować tę wartość, jak robiliśmy to już kilka razy podczas wykonywania ćwiczeń przy użyciu wartości "none":

```
nav ul li a {
  color: black;
  text-decoration: none;
}
```

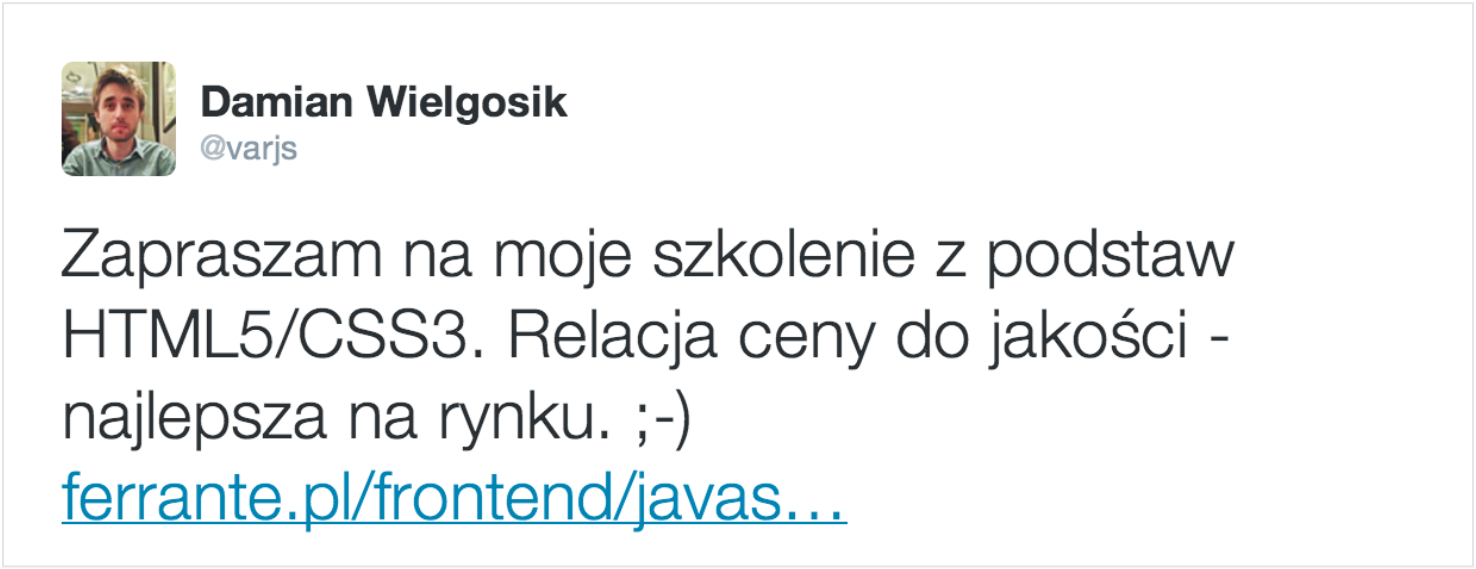


Pięknie! Uzyskaliśmy to, o co nam chodziło!

Przy okazji, jeśli pracujemy już z linkami, może pamiętasz, że na wielu stronach po najechaniu kursorem myszki linki podkreślają się. Na przykład na moim Twitterze umieściłem link do ogłoszenia dotyczącego szkolenia z HTML5:



Kiedy najeżdżam na niego kursorem myszki dzieje się coś ciekawego, co jako użytkownik internetu dobrze znasz. Tekst linku podkreśla się:



Spróbujmy zrobić coś podobnego w naszym menu. Niech po najechaniu każdego linka ten podkreśli się. Wykorzystamy do tego selektor pseudoklasy "hover":

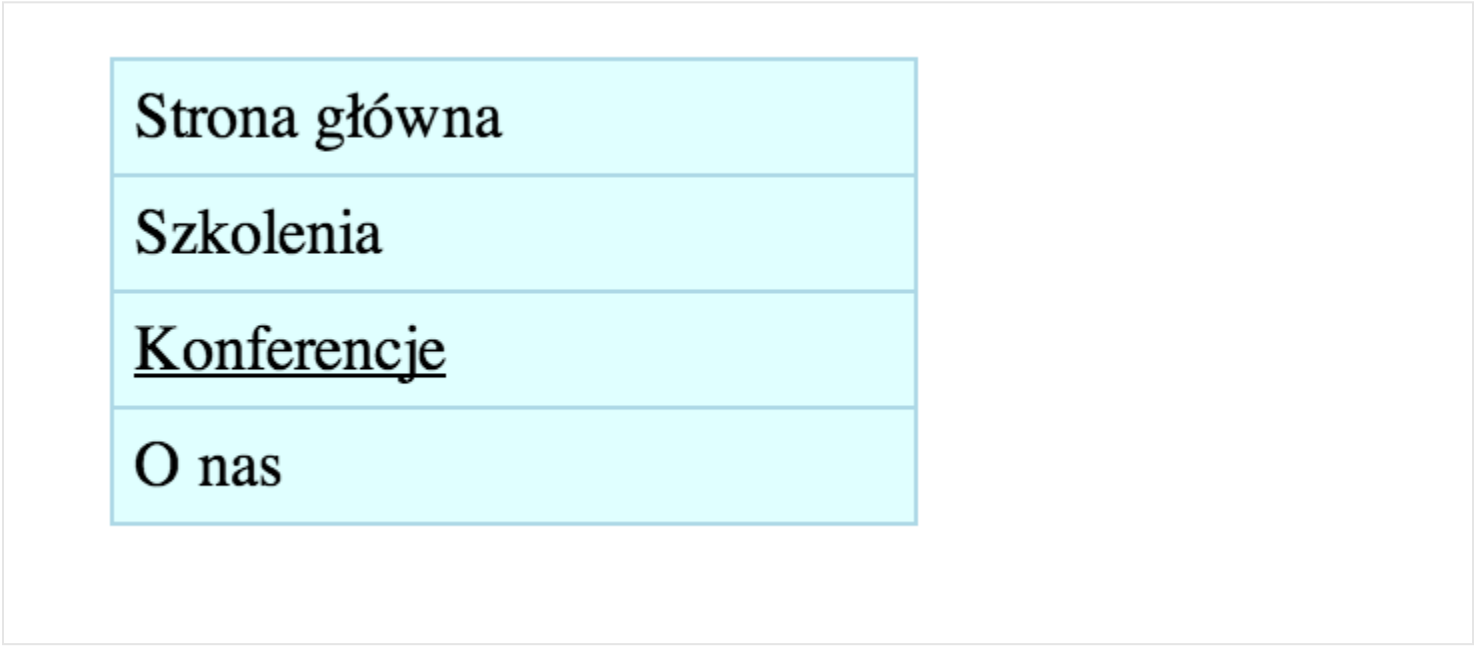
```
nav ul li a:hover {
  text-decoration: underline;
}
```

Tym razem dodaliśmy go do linków (<a>). Selektor ten oznacza stan, kiedy najeżdżamy kursorem myszki na link. Równie dobrze moglibyśmy skojarzyć go z innymi tagami:

```
div:hover
li:hover
img:hover
```

I tak dalej. Działanie ":hover" tłumaczymy sobie, odpowiadając na pytanie: "jak będzie wyglądał element użyty razem z :hover po najechaniu na niego kursorem myszki".

Wynik działania pokazuje poniższy zrzut ekranu:



Umieściliśmy kursor nad linkiem "Konferencje", więc jego tekst samoistnie podkreślił się.

Podsumowując, finalny kod CSS wygląda tak:

```
nav ul {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
  width: 200px;
  border: 1px solid LightBlue;
}

nav ul li {
  border-bottom: 1px solid LightBlue;
  padding: 5px;
}

nav ul li:last-child {
  border-bottom: 0;
}

nav ul li a {
  color: black;
  text-decoration: none;
}

nav ul li a:hover {
  text-decoration: underline;
}
```

Przeanalizuj go spokojnie. Z nowości pojawiły się selektory pseudoklas ":last-child" oraz ":hover". Na pewno przydadzą Ci się one w przyszłości.

W tym rozdziale poznałeś też, jak wstawia się linki. Na razie udało się nam dodać odsyłacze do fizycznych plików (szkolenia.html etc.). Nic nie stoi jednak na przeszkodzie, by "linkować" do adresów www. Jako przykład podajmy mojego bloga:

```
<a href="http://ferrante.pl">Mój blog</a>
```

Taki link wyświetli się tak: Mój blog. Zauważ, że w atrybucie href podałem "http://ferrante.pl" (link koniecznie musi zaczynać się od "http://", a nie "www", inaczej

nie zadziała!). Z kolei pomiędzy znacznikiem `<a>` umieszczamy tekst, którego kliknięcie spowoduje przejście do danej strony.

ROZDZIAŁ 7

Zrozumieć klasy w CSS

Przykład z menu, oprócz poznania dwóch selektorów pseudoklas, pozwoli nam nauczyć się jeszcze jednej ważnej rzeczy. Zapewne zauważyłeś, że dotychczas każdy selektor w CSS składał się z nazw tagów HTML. Ma to jedną, poważną wadę – kiedykolwiek zmienimy coś w strukturze kodu HTML, musimy dostosować również nasze CSSy. Może się tak stać na przykład w sytuacji, kiedy dany tekst nie jest już nagłówkiem. Wtedy zamiast nagłówka

```
<h1>Justin Bober zgolił wąsy</h1>
```

Powstaje kolejny akapit `<p>`:

```
<p>Justin Bober zgolił wąsy</p>
```

Stosując selektory oparte o nazwy tagów, narazimy się w takim przypadku na błędy w wyświetlaniu strony. Aby użytkownicy nie zauważyli różnicy, należy zmienić kod CSS, by zwykły akapit wyświetlał się tak samo jak element `<h1>`. Aby zminimalizować to ryzyko, w CSSie używa się klas i identyfikatorów.

Zacznijmy od klas. To łatwe. Otóż każdy element HTML może mieć specjalny atrybut o nazwie "class", gdzie możemy podać jedną lub kilka nazw. Oto kilka przykładów, jak można dodać taki atrybut:

- `<p class="news-item">` – klasa o nazwie "news-item"
- `<li class="menu-item">` – klasa o nazwie "menu-item"
- `<article class="news">` – klasa o nazwie "news"
- `<q class="important-quote">` – klasa o nazwie "important-quote"

I tak dalej. Nazwy klas są dowolne, aczkolwiek przyjęło się, że są to pojedyncze wyrazy lub ich zbitka z wykorzystaniem myślników "-" np. "my-item" i podkreśleń "_" - np. "special_item".

Można również dodać ich kilka naraz dla jednego tagu:

- <p class="news-item special-info"> – klasy o nazwie "news-item" i "special-info"
- <li class="menu-item selected"> – klasa o nazwie "menu-item" i "selected"

Zapytasz zapewne, jakie to ma znaczenie i do czego się przydaje. Otóż załóżmy, że tworzymy stronę z informacjami ze świata sportu. Uznajmy, że mamy na niej jeden główny, wyróżniony news (pogrubiłem odpowiedni fragment) oraz kilka normalnych. Kod HTML mógłby wyglądać tak:

```
<article><p>zwykły news</p></article>
<article><p>Wyróżniony news</p></article>
<article><p>zwykły news</p></article>
<article><p>zwykły news</p></article>
<article><p>zwykły news</p></article>
```

Gdybyśmy chcieli napisać kod CSS, który wyróżniłby specjalnym tłem pierwszy główny news, mielibyśmy problem. Nasza wiedza pozwala nam "ostylować" wszystkie elementy <article> w postaci takiego selektora, który poszuka wszystkie elementy <article>:

```
article {}
```

Pomoc przychodzi, kiedy do gry włączymy klasy CSS. Spróbujmy opisać każdy z elementów <article> taką nazwą klasy, która odpowiadałaby jego znaczeniu na całej stronie. A więc dla głównego newsa niech będzie to "main-news", a dla normalnego - "normal-news".

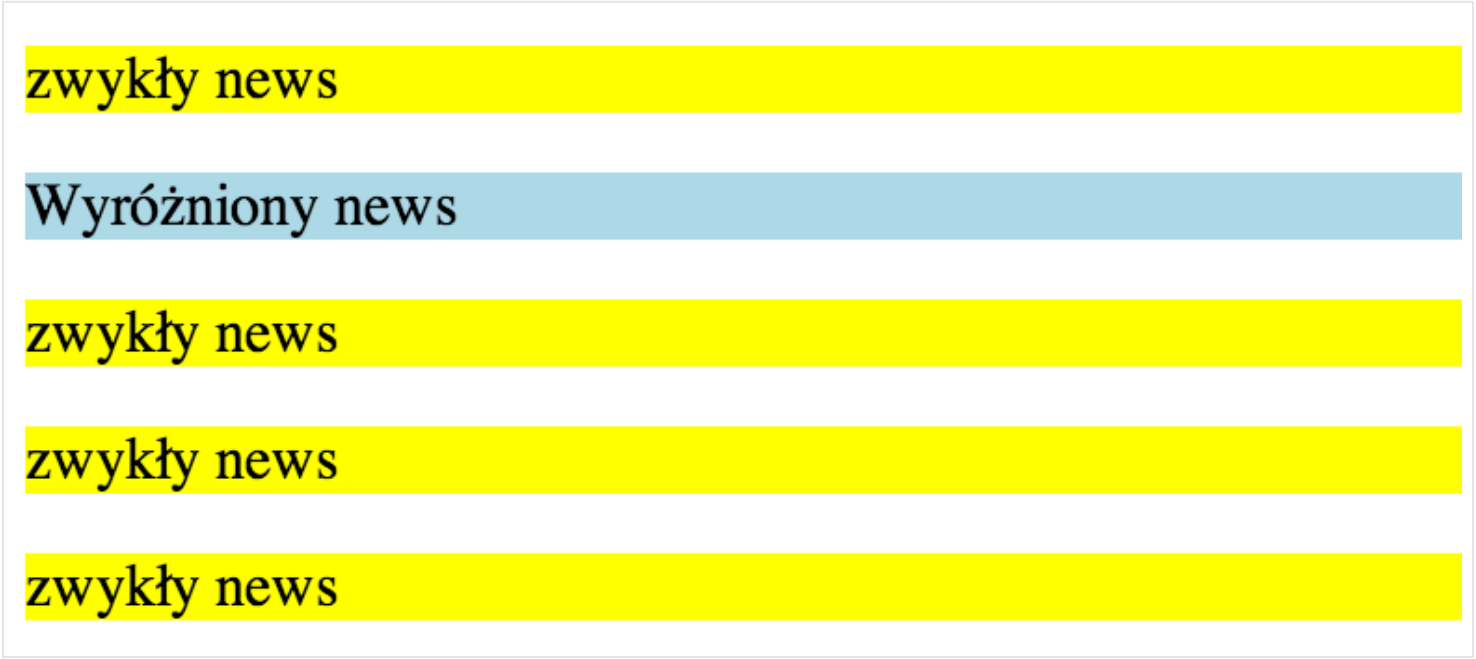
```
<article class="normal-news"><p>zwykły news</p></article>
<article class="main-news"><p>Wyróżniony news</p></article>
<article class="normal-news"><p>zwykły news</p></article>
<article class="normal-news"><p>zwykły news</p></article>
<article class="normal-news"><p>zwykły news</p></article>
```

W CSS na szczęście możemy użyć poniższych selektorów:

```
.main-news {
  background-color: LightBlue;
}

.normal-news {
  background-color: yellow;
}
```

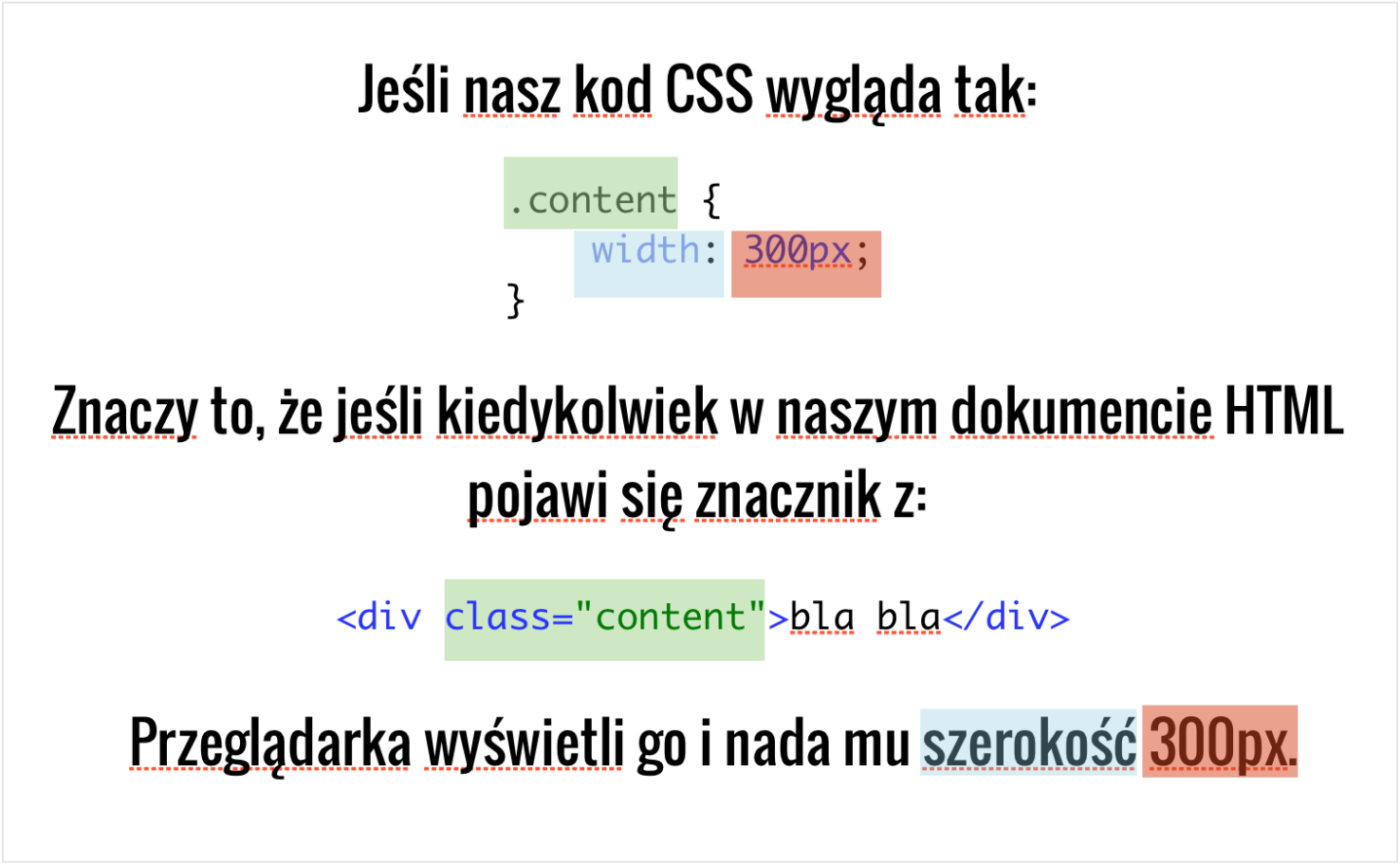
Każdy z nich, co ważne, zaczyna się kropką i składa się z nazwy klasy. A więc kiedykolwiek będziemy chcieli "wybrać" w selektorze elementy o danej klasie w HTML, używamy takiej konstrukcji. Uruchamiając powyższy kod, główny news będzie miał tło koloru jasno niebieskiego (LightBlue), a każdy normalny - żółtego.

W przeglądarce wygląda to tak:



Klasami CSS oznaczamy charakterystyczne dla strony elementy, pozbywając się zależności od nazw tagów. Być może w przyszłości zmienimy nasz kod i zamiast <article> pojawi się coś innego, na przykład będziemy nasze artykuły przetrzymywać w liście <ul>. Wtedy wystarczy tylko nanieść zmiany w HTML, nie ruszając kodu CSS, aby wygląd pozostał taki sam.

Istota klas wyjaśniona została w poniższym schemacie:



Zwróć uwagę na kolory zaznaczenia - tym samym kolorem wyróżnione zostały elementy analogiczne.

Oprócz klas wymyślono również identyfikatory. Używamy wtedy atrybutu o nazwie "id" i dowolnej wartości, według takich samych zasad jak klasy (a więc preferowane są różne zbitki wyrazowe). Przykładowy identyfikator w HTML-u może wyglądać tak:

```
<p id="main-content"></p>
```

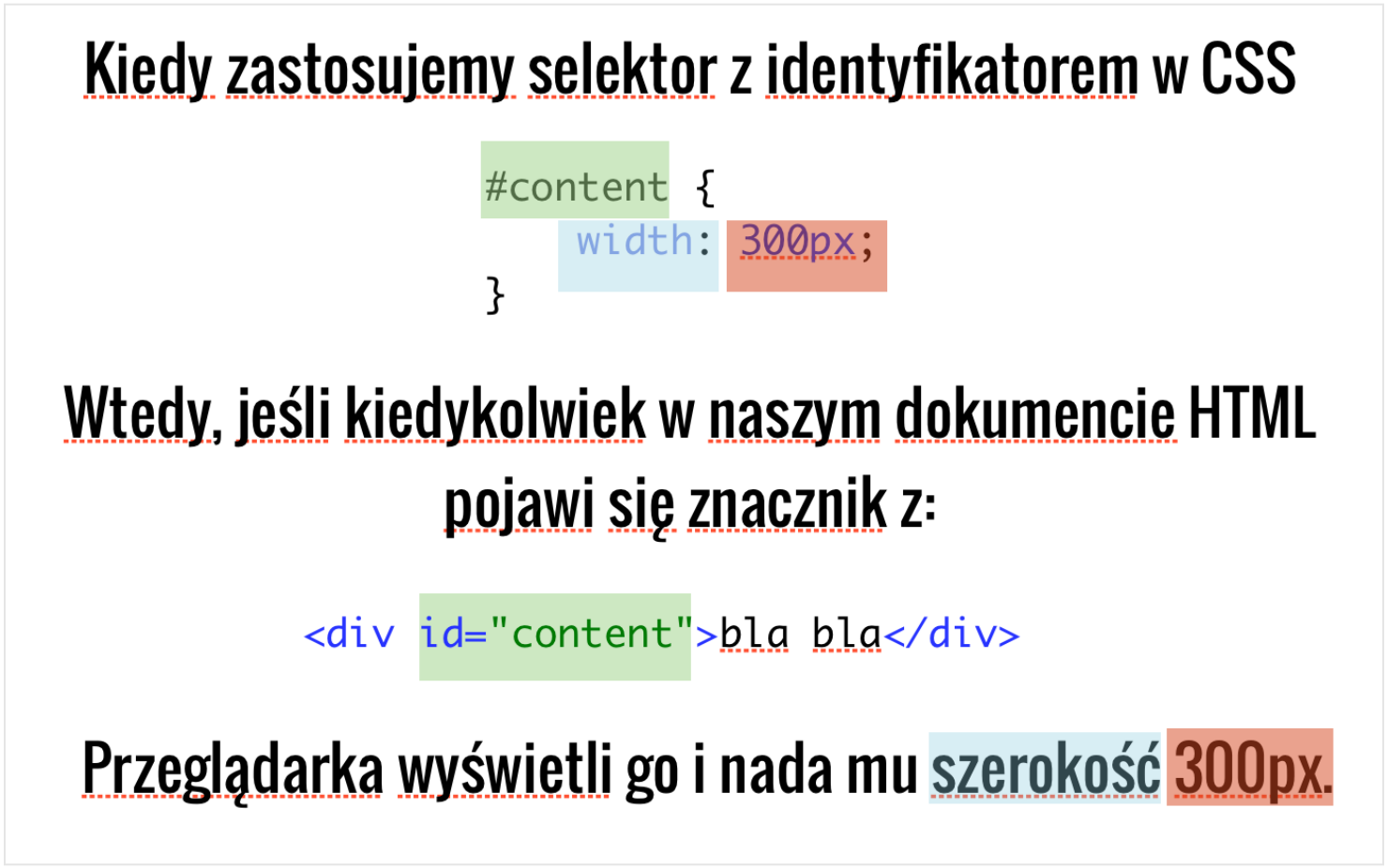
Kod CSS dla identyfikatorów wygląda z kolei tak:

```
#main-content {
  background-color: red;
}
```

W ten sposób dla elementu z identyfikatorem "main-content" ustawimy czerwone (red) tło. Inaczej niż ma to miejsce w klasach, tutaj, zamiast kropki, wstawiamy przed nazwę identyfikatora znak kratki - "#".

Bardzo ważne jest to, by pamiętać, że jak sama nazwa wskazuje, identyfikatory są unikalne, a więc **dany identyfikator może być użyty tylko raz w dokumencie HTML.** Stosowanie atrybutu "id" powinno być poprzedzone staranną analizą – zwykle wystarczają same klasy. Nie należy nadużywać identyfikatorów, ponieważ rzadko zdarza się, że strona składa się z kilkunastu unikatowych elementów.

Sposób działania identyfikatorów podsumowuje poniższy schemat:



Wykorzystując nową wiedzę, spróbujmy przerobić kod naszego menu, by stał się bardziej odporny na zmiany w HTML-u. Być może w przyszłości uznamy, że chcielibyśmy zrobić je używając innych tagów niż `<ul>` i `<li>`. Jeśli użyjemy odpowiednich klas, możemy wtedy spać spokojnie, bez konieczności równoległych zmian w kodzie CSS.

Wypada zacząć od HTML. Obecnie kod odpowiedzialny za reprezentację menu wygląda tak:

```
<nav>
  <ul>
    <li>
      <a href="index.html">Strona główna</a>
    </li>
    <li>
      <a href="szkolenia.html">Szkolenia</a>
    </li>
    <li>
      <a href="konferencje.html">Konferencje</a>
    </li>
    <li>
      <a href="onas.html">O nas</a>
    </li>
  </ul>
</nav>
```

Wprowadźmy teraz klasy Nadajmy je dla całego kontenera, przechowującego menu (tag <ul>) oraz odpowiednich elementów tego menu (tagi <li>):

```
<ul class="site-nav">
  <li class="site-nav-item">
    <a href="index.html">Strona główna</a>
  </li>
  <li class="site-nav-item">
    <a href="szkolenia.html">Szkolenia</a>
  </li>
  <li class="site-nav-item">
    <a href="konferencje.html">Konferencje</a>
  </li>
  <li class="site-nav-item">
    <a href="onas.html">O nas</a>
  </li>
</ul>
```

Dla całego menu użyłem klasy "site-nav", natomiast dla jego elementów wybrałem klasę "site-nav-item". Pora teraz dostosować kod CSS:

```
.site-nav {
  background-color: LightCyan;
  list-style: none;
  padding: 0;
  width: 200px;
  border: 1px solid LightBlue;
}

.site-nav .site-nav-item {
  border-bottom: 1px solid LightBlue;
  padding: 5px;
}

.site-nav .site-nav-item:last-child {
  border-bottom: 0;
}

.site-nav .site-nav-item a {
  color: black;
  text-decoration: none;
}

.site-nav .site-nav-item a:hover {
  text-decoration: underline;
}
```

Aby lepiej zrozumieć, co się zmieniło, spójrz na porównanie starego (z lewej) kodu CSS i nowego (z prawej):

```css
nav ul {
    background-color: LightCyan;
    list-style: none;
    padding: 0;
    width: 200px;
    border: 1px solid LightBlue;
}

nav ul li {
    border-bottom: 1px solid LightBlue;
    padding: 5px;
}

nav ul li:last-child {
    border-bottom: 0;
}

nav ul li a {
    color: black;
    text-decoration: none;
}

nav ul li a:hover {
    text-decoration: underline;
}
```

```css
.site-nav {
    background-color: LightCyan;
    list-style: none;
    padding: 0;
    width: 200px;
    border: 1px solid LightBlue;
}

.site-nav .site-nav-item {
    border-bottom: 1px solid LightBlue;
    padding: 5px;
}

.site-nav .site-nav-item:last-child {
    border-bottom: 0;
}

.site-nav .site-nav-item a {
    color: black;
    text-decoration: none;
}

.site-nav .site-nav-item a:hover {
    text-decoration: underline;
}
```

Spróbuj porównać każdy selektor. Jak łatwo zauważyć, zamieniliśmy po prostu tagi na klasy, co daje nam znacznie większą elastyczność w pisaniu kodu. Przy okazji skróciliśmy każdy z nich, usuwając "nav".

Staraj się używać klas w selektorach i minimalizować użycie tagów, chyba że jest ono oczywiste - na przykład dla linków w `<a>`. Ich nie da się podmienić innym tagiem HTML. Identyfikatory stosuj tylko wtedy, kiedy dany element jest unikalny w skali strony. W powyższym przypadku, mogło to być właśnie menu, ale nie musiało - często menu występuje kilka razy - na przykład na dole i górze strony.

Działanie i definiowanie klas oraz identyfikatorów podsumowuje poniższy schemat:

elementów z tym samym atrybutem „class” (klas) może być w dokumencie HTML wiele

```
<div class="news">news</div>
<div class="news">inny news</div>
<div class="news">jeszcze inny news</div>
```

w CSS dodajemy cechy wyglądu do klas poprzez kropkę i nazwę klasy

```
.news {
    width: 300px;
}
```

może być tylko jeden element z danym atrybutem „id”

```
<div id="article">article</div>
```

w CSS dodajemy cechy wyglądu do identyfikatorów poprzez # i nazwę klasy

```
#article {
    width: 300px;
}
```

ROZDZIAŁ 8

Czas podsumowań

Pora na chwilę odetchnąć. Na poprzednich stronach poznałeś wiele zasadniczych konceptów tworzenia stron internetowych. Myślę, że to dobry moment na podsumowanie tego, co do tej pory wiemy, by być przygotowanym na kolejne rozdziały.

Zaczęliśmy od określenia, jak zbudowane są strony internetowe. Ustaliliśmy, że jest to zwykły tekst z pewnymi dodatkami, które opisują jego poszczególne fragmenty. Wybraliśmy więc najmniejsze, niepowtarzające się części strony według funkcji, jaką pełnią w całym dokumencie. Porównaliśmy je do klocków, które ułożone byłyby jeden pod drugim.

Dokonując takiej analogii ustaliliśmy, że każdemu klockowi odpowiada jakiś tag HTML. Tagi HTML są z góry zdefiniowane i nie wymyślamy ich. Jeśli musimy zaznaczyć na stronie akapit z tekstem - używamy tagu `<p>`. Jeśli oznaczamy cały artykuł - `<article>`. Jeśli mamy do czynienia ze swego rodzaju nagłówkiem – wybieramy `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` lub `<h6>`, w zależności od tego kontekstu. Całą listę tagów HTML, które możemy użyć znajdziemy na stronie specyfikacji HTML.

Stwierdziliśmy, że tagi możemy zagnieżdżać. Na przykładzie artykułu z portalu internetowego zgodziliśmy się, że składa się on z tytułu, dodatku multimedialnego (najczęściej obrazek) i akapitów z treścią. Dokonując takiego podziału, wybraliśmy tagi HTML, które pełnią te właśnie funkcje. Były to `<header>`, `<h1>`, `<figure>` i `<p>`, które znalazły się z kolei w obrębie `<article>`. Nazwaliśmy je "dziećmi" `<article>`, który jest z kolei "rodzicem". W HTML staramy się właśnie myśleć hierarchicznie. Jeśli widzimy na stronie jakiś prostokąt, dajmy na to z tłem i obramowaniem, a w nim kolejnych kilka (np.

akapity tekstu), to zapewne w kodzie HTML znajdziemy odpowiednik tego prostokąta, a w jego obrębie zapisane pozostałe elementy. Nic w przeglądarce nie wyświetla się bez przyczyny. Każda warstwa, każdy element, który widzisz na stronie ma swój odpowiednik w kodzie HTML i CSS.

Następnie ustaliliśmy, że strony internetowe zbudowane tylko w HTML-u są nieatrakcyjne wizualnie. Dzięki CSS można to na szczęście zmienić. Pisząc kod CSS wybieramy tagi, które chcemy ozdobić, zapisując to w specjalnym selektorach. A więc jeśli chcemy "dostać się"" do wszystkich akapitów w `<article>` piszemy tak:

```
article p {}
```

Oprócz tego, bardzo pomocne są **klasy** w CSS. Dzięki nim możemy uniezależnić kod CSS od tagów w pliku HTML (przecież to w sumie dwa, osobne pliki!). Każdy tag może być oznaczony klasą poprzez dodanie mu atrybutu `class="nazwa klasy"`. Nazwy klas wymyślasz sam. Utarło się, że są to angielskie wyrazy, które opisują funkcję, jaką pełni w dokumencie dany element HTML. Na przykład jeśli przechowujemy w nim informację prasową, możemy napisać tak:

```
<article class="news">...</article>
```

Wtedy w CSS znajdziemy taki element w poniższy sposób:

```
.news {}
```

Oprócz klas możemy używać **identyfikatorów**. Dodajemy je w ten sposób:

```
<article id="main-news">...</article>
```

Za każdym razem należy dobrze przemyśleć użycie atrybutu id, ponieważ identyfikatorami oznaczamy unikatowe w dokumencie elementy.

Używając identyfikatora "main-news", w kodzie CSS dostajemy się do nich tak:

```
#main-news {}
```

W ten sposób nauczyliśmy się najbardziej istotnych założeń HTML i CSS. Tworząc strony zawsze dokonuj najpierw podziału treści na mniejsze części i oceń, jaką funkcję pełnią w dokumencie. Wyróżnij menu, poszczególne artykuły, a w nich tytuły, daty, akapity, cytaty i obrazki. Bądź szczegółowy jak tylko się da. Następnie opisz wyróżnione części odpowiednimi tagami HTML, które pasują do treści, jaką chcesz w nich zawrzeć. Jeśli dany element jest istotny z wizualnego punktu widzenia, prawdopodobnie warto dodać mu klasę w CSS, by potem łatwo zmienić jego wygląd, jak i innych elementów tej samej klasy. Zapamiętaj, że HTML odpowiada za treść, a CSS za wygląd. Nigdy nie sugeruj się tym, że przeglądarka wyświetla dany element pogrubiony lub podkreślony bez Twojej ingerencji. Jest to zapewne jej domyślne zachowanie i nie należy na tym polegać, bo równie dobrze kolejna wersja programu może je zmienić.

Wielu zapyta zapewne, jak to się dzieje, że po wpisaniu danego adresu widzimy konkretną stronę. Na przykład po wpisaniu w przeglądarce adresu ferrante.pl pokazuje się mój blog. Otóż aby publikować nasze strony w Internecie musimy zdobyć dostęp do serwera www i wykupić domenę internetową (w moim wypadku nazwa domeny to "ferrante.pl"). Domenę można nabyć w wielu firmach (np. nazwa.pl) i jest to koszt kilkudziesięciu złotych. Kupując na przykład "jankowalski.pl" zakładamy, że nasze strony będą dostępne pod właśnie tym adresem. Zdobywając domenę, należy następnie wykupić powierzchnię serwerową. Tak jak strony najpierw testujemy u siebie na dysku, tak aby były dostępne w internecie należy wgrać je na serwer, który sprawia, że nasza witryna będzie widoczna dla innych internautów, nie tylko dla nas. Z reguły w tym

momencie trzeba skonfigurować naszą domenę, by wskazywała na nasz świeżo zakupiony serwer. Czas teraz na umieszczenie w internecie pierwszego pliku ze stroną w HTML. Uznajmy, że nazywa się on blog.html. Musimy więc wgrać go na serwer (podobnie jak wgrywamy zdjęcia czy inne pliki). Posłuży nam do tego dowolny program tzw. klient FTP (np. Filezilla). Po wszystkim, gdy nasz plik blog.html wyląduje na serwerze, strona będzie dostępna pod jankowalski.pl/blog.html.

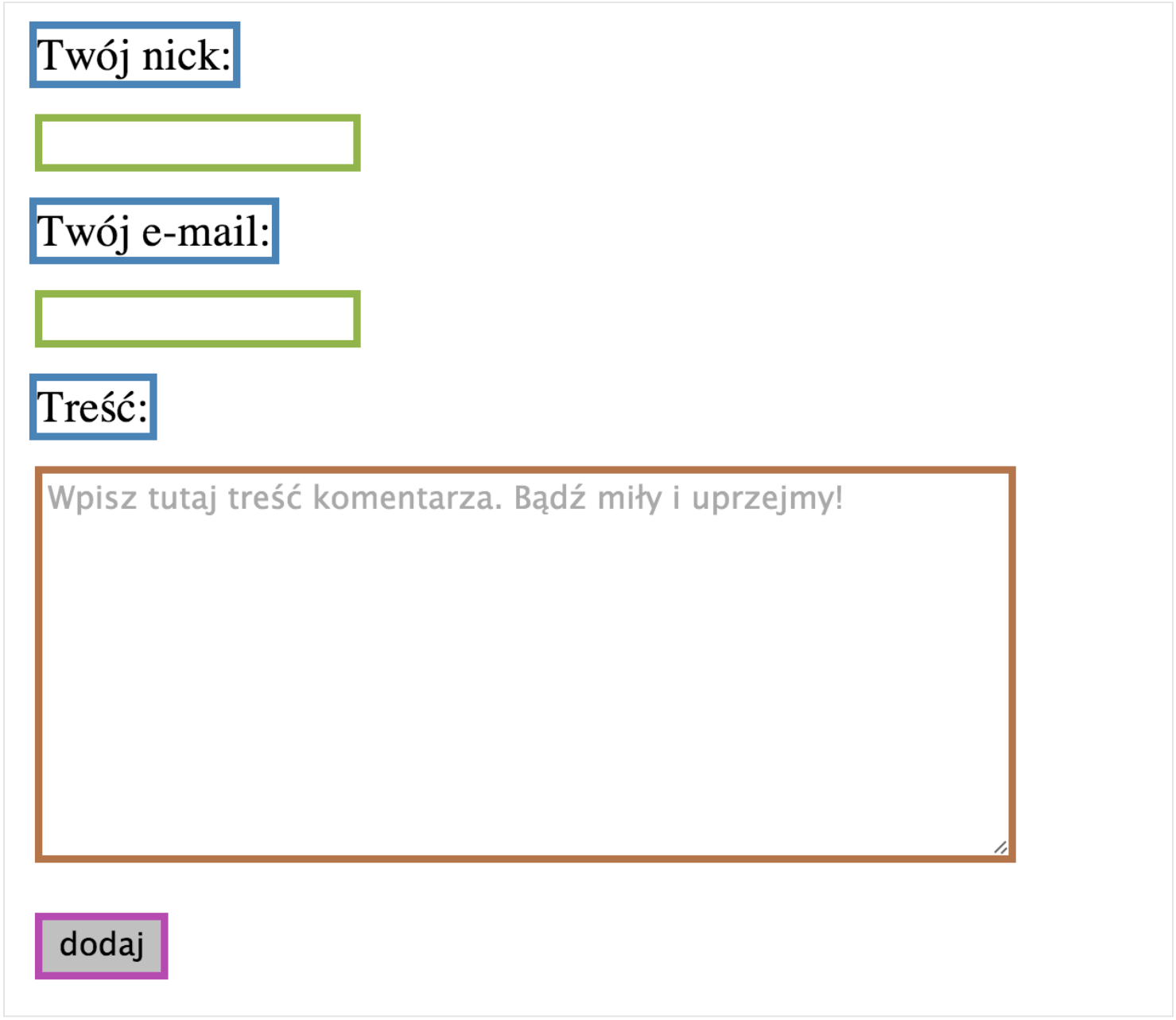
ROZDZIAŁ 9

Formularze

Do tej pory udało się nam stworzyć kilka elementów strony internetowej. Nie wspomnieliśmy jednak nic o formularzach, w których użytkownik mógłby wpisać dane. Formularze w internecie używane są w wielu miejscach - wyszukiwarka, Facebook, czy Gmail - wszędzie mamy do czynienia z miejscami, gdzie możemy coś wpisać i wysłać dalej. Spróbujmy stworzyć prosty formularz do komentowania naszego artykułu.

Niech wygląda on następująco:

Spróbujmy zaznaczyć teraz poszczególne elementy, z których składa się formularz. Użyjemy tego samego koloru zaznaczenia dla tych, które pełnią podobną funkcję:



Jak widać, na niebiesko zaznaczyliśmy opisy poszczególnych pól formularza. Na zielono elementy, gdzie można wpisać tekst w jednej linijce, na brązowo pole do wpisywanie dłuższego tekstu, a kolorem różowym zaznaczyliśmy przycisk do wysyłania formularza.

Pytanie teraz, jakie tagi wybrać dla każdej z tych grup. Dla oznaczeń będzie to `<label>`. Dla krótkich pól – `<input type="text">`, do dłuższych wypowiedzi będzie `<textarea>`.

Przyciski dodawania zrobimy natomiast z <input type="submit">. Są to najpopularniejsze elementy HTML, jakich używa się do tworzenia formularzy na stronach internetowych.

Zaczynamy standardowo od pustego szablonu strony HTML, który będziemy stopniowo uzupełniać.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
      <meta charset="utf-8">
      <title>Formularz</title>
      <link rel="stylesheet" href="main.css">
  </head>
  <body>
  </body>
</html>
```

Jako że mamy do czynienia z formularzem, musimy zastosować odpowiedni tag, który "powie" przeglądarce, hej, od tej pory zaczyna się formularz (podobnie jak to było dla artykułu z tagiem <article>). W HTML jest na szczęście <form>!

```
<!DOCTYPE html>
<html lang="pl">
  <head>
      <meta charset="utf-8">
      <title>Formularz</title>
      <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <form>
    </form>
  </body>
</html>
```

Następnie czas na wstawienie pierwszego pola formularza wraz z opisem "Twój nick". Od tej pory będę umieszczał wycinki z kodu zaczynając od <form>, ponieważ reszta jest Ci dobrze znana.

```
<form>
  <label for="nickname">Twój nick:</label>
  <input type="text" id="nickname" name="nickname">
</form>
```

Wstawiliśmy dwa elementy. <label> dla opisu oraz <input> dla pola do wpisywania tekstu. <input> ma trzy atrybuty – type o wartości text, name o wartości nickname i id również o wartości nickname. Pierwszy, type, oznacza, że jest to krótkie pole tekstowe. Za każdym razem, gdy będziesz potrzebował takiego pola:

użyjesz w kodzie <input type="text">. Wiedz też, że takie pole może zostać specjalnie ostylowane w CSS, a więc możemy na przykład zmienić mu ramkę, szerokość czy odstępy pomiędzy wpisanym tekstem a obramowaniem:

Oprócz atrybutu type, użyłem również atrybutu name. Otóż służy on do nazwania każdego z pól. Jest to przydatne kiedy wysyłamy taki formularz do serwera – wtedy łatwo zidentyfikować z jakiego pola pochodzi dana treść. Zauważ korelację pomiędzy wartością identyfikatora id a wartością atrybutu for w <label>:

```
<label for="nickname">Twój nick:</label>
<input type="text" id="nickname" name="nickname">
```

W atrybucie for należy podać id pola, jakie opisuje element <label>. Dzięki temu klikając na opis w <label>, dane pole automatycznie uaktywni się i będziemy mogli w nim coś wpisać.

Pora na kolejny element formularza – e-mail.

```
<form>
  <label for="nickname">Twój nick:</label>
  <input type="text" id="nickname" name="nickname">
  <label for="user-email">Twój e-mail:</label>
  <input type="email" id="user-email" name="user-email">
</form>
```

Jedyna różnica w porównaniu z poprzednim polem leży w atrybucie type. Tym razem jego wartością jest email, co jest całkiem logiczne, ponieważ mamy do czynienia z miejscem na wpisanie swojego adresu e-mail. Co ciekawe, stosując taką wartość, wiele urządzeń mobilnych dostosowuje wygląd klawiatury, by jak najwygodniej wpisywało się adres. Przy okazji wszystko, co wpiszemy w to pole, a co nie będzie spełniało założeń poprawnego adresu e-mail (np. brak znaku @) zostanie sprawdzone, a przeglądarka wyświetli stosowny komunikat i nie pozwoli wysłać takiego formularza w świat.

Kolejne pole, jakie musimy dodać, to miejsce na wpisanie treści komentarza. Do dłuższych tekstów jest w HTML tag <textarea>:

```
<form>
  <label for="nickname">Twój nick:</label>
  <input type="text" id="nickname" name="nickname">
  <label for="user-email">Twój e-mail:</label>
  <input type="email" id="user-email" name="user-email">
  <label for="content">Treść:</label>
  <textarea rows="10" cols="50" id="content"  name="content"
></textarea>
</form>
```

Zauważ, że pojawiły się tutaj dwa nowe atrybuty - rows i cols. rows oznacza liczbę linijek tekstu, jakie można wpisać do <textarea>, a które będą widoczne od razu, bez przesuwania (tzw. "skrolowania") paska bocznego. W cols podajemy maksymalną liczbę znaków na jedną linijkę. Spróbuj zmienić te własności, by zobaczyć, jak zwiększa się lub zmniejsza to pole.

Ostatnia rzecz, jaką musimy dodać to przycisk do wysyłania formularza.

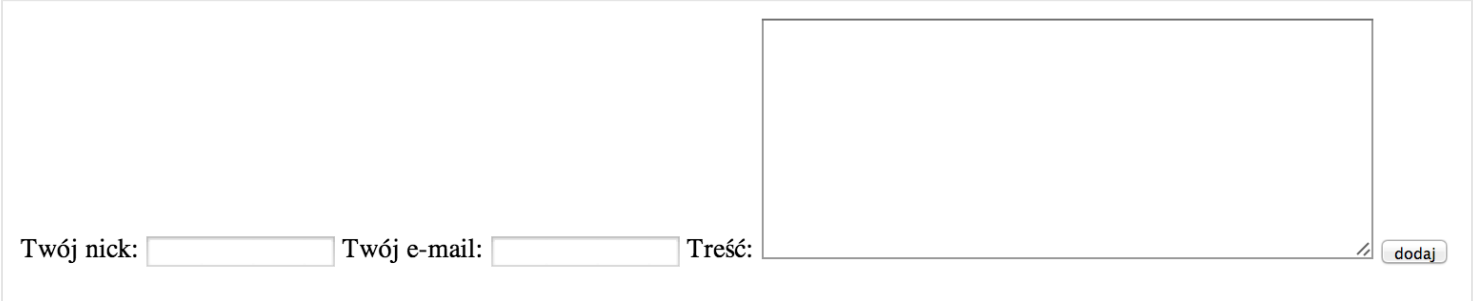
```
<form>
  <label for="nickname">Twój nick:</label>
  <input type="text" id="nickname" name="nickname">
  <label for="user-email">Twój e-mail:</label>
  <input type="email" id="user-email" name="user-email">
  <label for="content">Treść:</label>
  <textarea rows="10" cols="50" id="content" name="content"
></textarea>
  <input type="submit" value="dodaj">
</form>
```

W tym celu stosuje się element <input> z atrybutem type równym submit. Z kolei to, co wpiszesz do atrybutu "value" wyświetli się jako tekst na przycisku.

Kod naszej strony z formularzem wygląda teraz tak:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
      <meta charset="utf-8">
      <title>Formularz</title>
      <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <form>
      <label for="nickname">Twój nick:</label>
      <input type="text" id="nickname"  name="nickname">
      <label for="user-email">Twój e-mail:</label>
      <input type="email" id="user-email"  name="user-email">
      <label for="content">Treść:</label>
      <textarea rows="10" cols="50" id="content" name="content"
></textarea>
      <input type="submit" value="dodaj">
    </form>
  </body>
</html>
```

Z kolei przeglądarka wyświetla ją następująco:



Nie jest to wymarzona sytuacja. W następnym rozdziale dowiemy się, jak to naprawić.

ROZDZIAŁ 10

Różnica między a

W poprzednim rozdziale wszystkie elementy wyświetliły się w tej samej linijce, jeden za drugim. Dzieje się tak dlatego, że `<label>`, `<input>` i `<textarea>` są tzw. elementami liniowo-blokowymi. Zachowują się one jak zwykły tekst i cokolwiek byś w nich nie umieścił, wyświetlone zostaną obok siebie. Różnią się od pozostałych tym, że możesz nadać im w CSS różne wymiary (dlatego też jeden z członów tej nazwy brzmi "blokowe").

Zastanówmy się, jak możemy rozwiązać problem elementów formularza wyświetlających się obok siebie. Przydałby się jakiś tag, który powiedziałby przeglądarce: "hej, jestem pojemnikiem, który zawiera kilka tagów.". Takim swoistym kontenerem na inne elementy, który przy okazji wprowadzi "złamanie treści" do nowej linii, jest tag `<div>`.

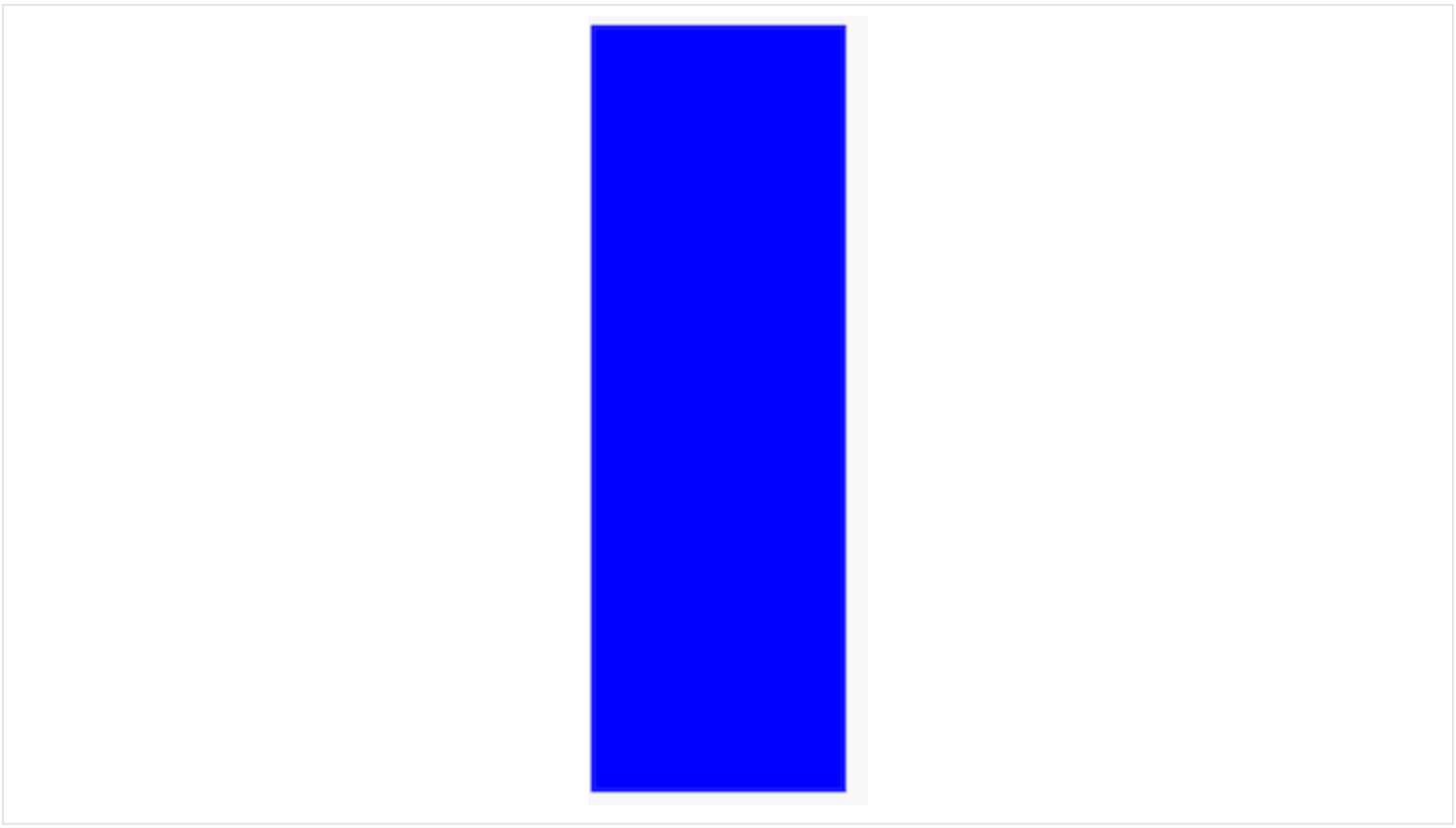
Aby uzmysłowić sobie, czym są elementy `<div>`, użyjmy w naszym kodzie trzech divów i nadajmy im szerokość, wysokość oraz tło. HTML spełniający te założenia będzie wyglądał tak:

```
<div class="container"></div>
<div class="container"></div>
<div class="container"></div>
```

W CSS natomiast napiszemy tak, odnosząc się do klasy "container":

```
.container {
  width: 200px;
  height: 200px;
  background-color: blue;
}
```

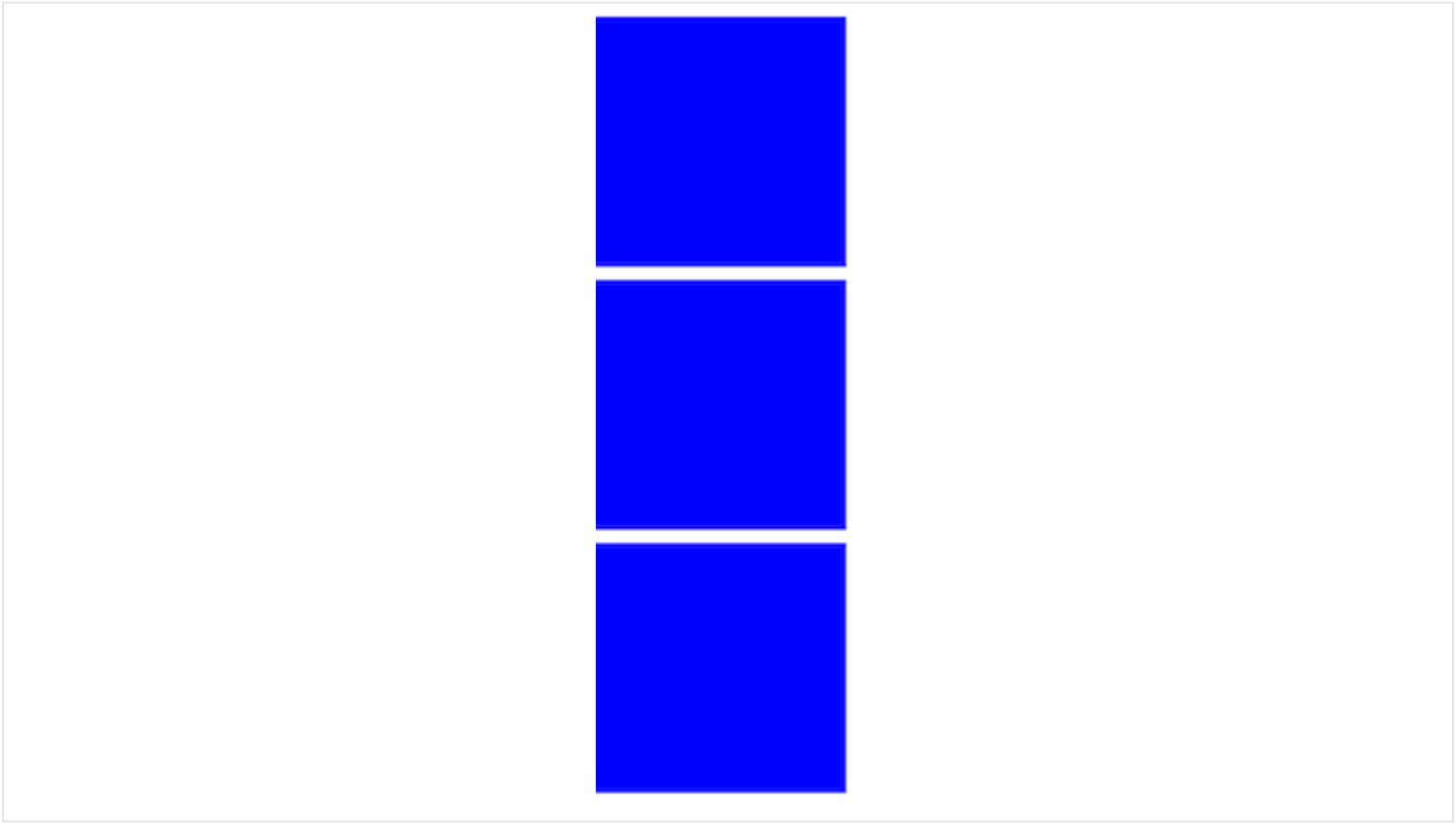
Jak widzisz, dodałem dla wszystkich elementów z klasą container szerokość 200px, taką samą wysokość i kolor tła "blue". Sprawdźmy, jak to wygląda w przeglądarce:



Jak widać, jest to jedna, niebieska masa. Odzielmy więc od siebie każdy z elementów o 10px:

```
.container {
  width: 200px;
  height: 200px;
  background-color: blue;
  margin-bottom: 10px;
}
```

Aby dodać przerwę w pionie między elementami użyłem właściwości margin-bottom. Sprawdźmy, jak wygląda to teraz:



Widać teraz wyraźnie, że przeglądarka wyświetla elementy `<div>` jeden pod drugim, w przeciwieństwie do elementów formularza, które pojawiły się jednej linii.

Skąd taka różnica? Otóż jeśli chodzi o wyświetlanie tagów, przeglądarka rozróżnia trzy grupy elementów:

- liniowe
- blokowe
- liniowo-blokowe

Najprościej uzmysłowić sobie, że nazwy tych grup nie są przypadkowe i tak: elementy liniowe nie spowodują przejścia do nowej linii, będą wyświetlać się obok siebie, w jednej… linii. Blokowe z kolei będą zachowywać się jak pudełka, które ustawiasz jeden pod drugim, a więc nigdy nie wyświetlą się obok siebie, chyba że użyjesz magicznych sztuczek w CSS, o których dowiesz się w następnym rozdziale. Liniowo-blokowe z kolei będą zachowywać się jak liniowe, jeśli chodzi o położenie (a więc wyświetlać będą się obok siebie), natomiast różnią się od nich tym, że możesz nadać im różne wielkości. Ma

to sens, np. `<textarea>` wyświetla się jako duży prostokąt i zdecydowanie "wychodzi" poza pojedynczą linijkę tekstu.

Podam Ci teraz kilka przykładów elementów wraz z ich kwalifikacją do każdej z tych grup:

- liniowe – `<span>`, `<em>`, `<strong>`
- blokowe – `<div>`, `<p>`, `<article>`
- liniowo-blokowe – `<input>`, `<textarea>`

Domyślnie, elementom liniowym przeglądarka ustawia właściwość "display" na "inline". Blokowym na "block", a liniowo blokowym na "inline-block". Na przykład tag `<span>` nie spowoduje przejścia tekstu do nowej linii, bo jest elementem liniowym i przeglądarka z automatu nadaje mu styl `display` na `inline`:

```
<span>raz</span> <span>dwa</span> <span>trzy</span>
```

Przeglądarka wyświetli więc tekst "raz dwa trzy" w jednej linii:

raz dwa trzy

Możliwa jest jednak zmiana sposobu wyświetlania danych tagów. Gdybyśmy zapragnęli, aby każdy `<span>` zachowywał się jak element blokowy, a więc powodował przejście do nowej linii, moglibyśmy napisać w CSS tak:

```
span {
  display: block;
}
```

Wtedy nasz przykład wyświetlałby się następująco:

raz

dwa

trzy

Wracając do problemu z formularzem, spośród elementów blokowych wybrałem . Zapytasz zapewne, czy nie zburzy to semantycznego sensu o jaki dbamy od początku tej książki. Drugie pytanie, które trzymasz jako as w rękawie z pewnością dotyczy tego, co tak naprawdę ten tag oznacza. Z rozbrajającą szczerością odpowiadam, że… nic. Stosujemy go w wypadkach, kiedy spośród wszystkich innych tagów nie znajdujemy zastosowania dla tego, co chcemy w dokumencie umieścić. Utarło się więc, że , najczęściej w parze z klasami, stosujemy do uzyskania różnych efektów wizualnych, skoro nie pełni on żadnej funkcji w opisywaniu treści. Na przykład kiedy chcemy zrobić trzy kolumny na naszej stronie, czy też w przypadku, który rozważamy – kiedy poszukujemy elementu, który rozdzieli kilka elementów liniowych lub liniowo-blokowych. Mimo wszystko staraj się nie nadużywać tego elementu – w pierwszej kolejnosci poszukaj tagów, które będą odpowiadać funkcji danego tekstu w dokumencie.

Bogatsi o wiedzę na temat elementów blokowych, spróbujmy uzupełnić kod naszego formularza w ten sposób, by pola formularza i ich opisy wyświetlane były jeden pod drugim:

```
<form>
  <div>
    <label for="nickname">Twój nick:</label>
  </div>
  <div>
    <input type="text" id="nickname" name="nickname">
  </div>
  <div>
    <label for="user-email">Twój e-mail:</label>
  </div>
  <div>
    <input type="email" id="user-email" name="user-email">
  </div>
  <div>
    <label for="content">Treść:</label>
  </div>
  <div>
    <textarea rows="10" cols="50" id="content"
name="content"></textarea>
  </div>
  <div>
    <input type="submit" value="dodaj">
  </div>
</form>
```

Wygląda to nieco bardziej skomplikowanie, ale tak naprawdę jedyne co zrobiliśmy, to dodaliśmy tagi `<div>` wokół każdego z elementów formularza.

Kolejny raz spójrz na porównanie poprzedniego kodu z nowym:

```
<form>
  <label for="nickname">Twój nick:</label>
  <input type="text" id="nickname" name="nickname">
  <label for="user-email">Twój e-mail:</label>
  <input type="email" id="user-email" name="user-email">
  <label for="content">Treść:</label>
  <textarea rows="10" cols="50" id="content" name="
  content"></textarea>
  <input type="submit" value="dodaj">
</form>
```

```
1  <form>
2    <div>
3      <label for="nickname">Twój nick:</label>
4    </div>
5    <div>
6      <input type="text" id="nickname" name="nickname">
7    </div>
8    <div>
9      <label for="user-email">Twój e-mail:</label>
10   </div>
11   <div>
12     <input type="email" id="user-email" name="user-email">
13   </div>
14   <div>
15     <label for="content">Treść:</label>
16   </div>
17   <div>
18     <textarea rows="10" cols="50" id="content" name="content"></textarea>
19   </div>
20   <div>
21     <input type="submit" value="dodaj">
22   </div>
23 </form>
```

A teraz zobacz, jak to się wyświetla w przeglądarce!

Wygląda na to, że właśnie spełniliśmy założenia tego zadania, brawo!

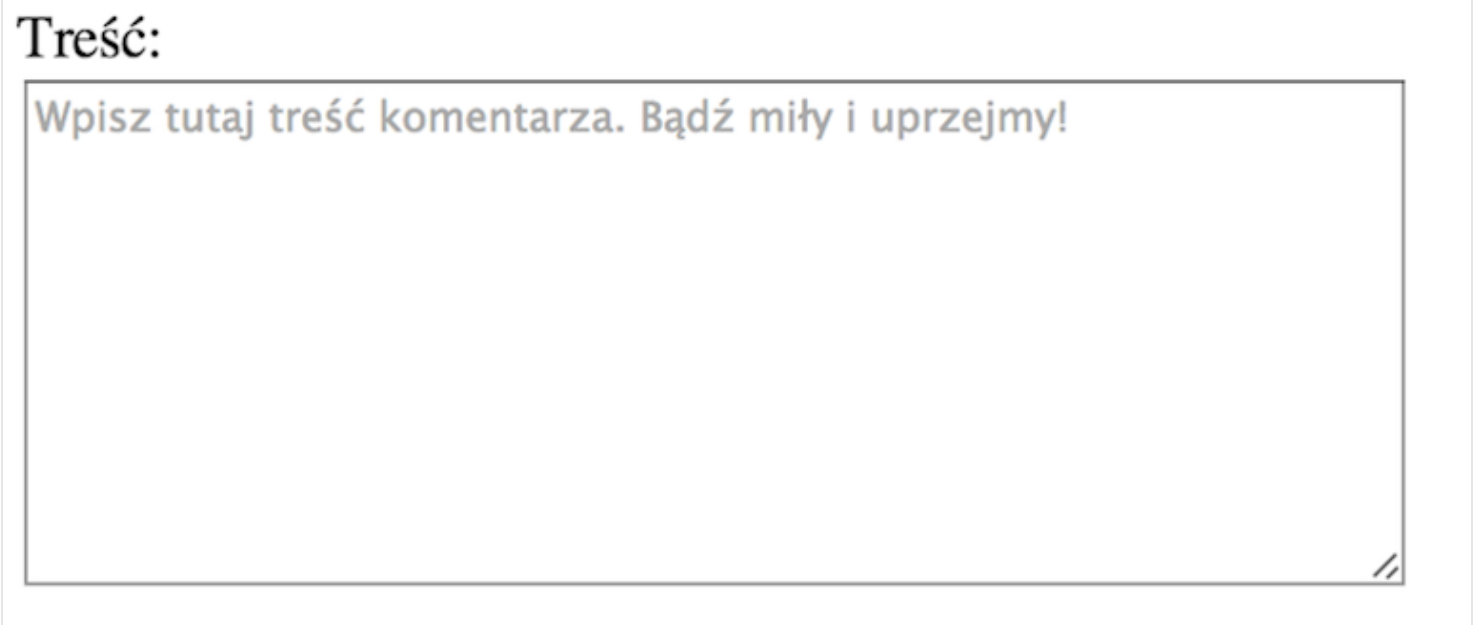
ROZDZIAŁ 11

Dodatki do formularzy

Mam dla Ciebie jeszcze kilka niespodzianek. W HTML5 jest kilku atrybutów, które pomogą nam znacznie uatrakcyjnić nasz formularz. Jednym z nich jest placeholder. Dodajmy go do jednego z elementów, na przykład do <textarea>:

```
<textarea rows="10" cols="50" id="content" name="content" placeholder="Wpisz tutaj treść komentarza. Bądź miły i uprzejmy!"></textarea>
```

Efekt?



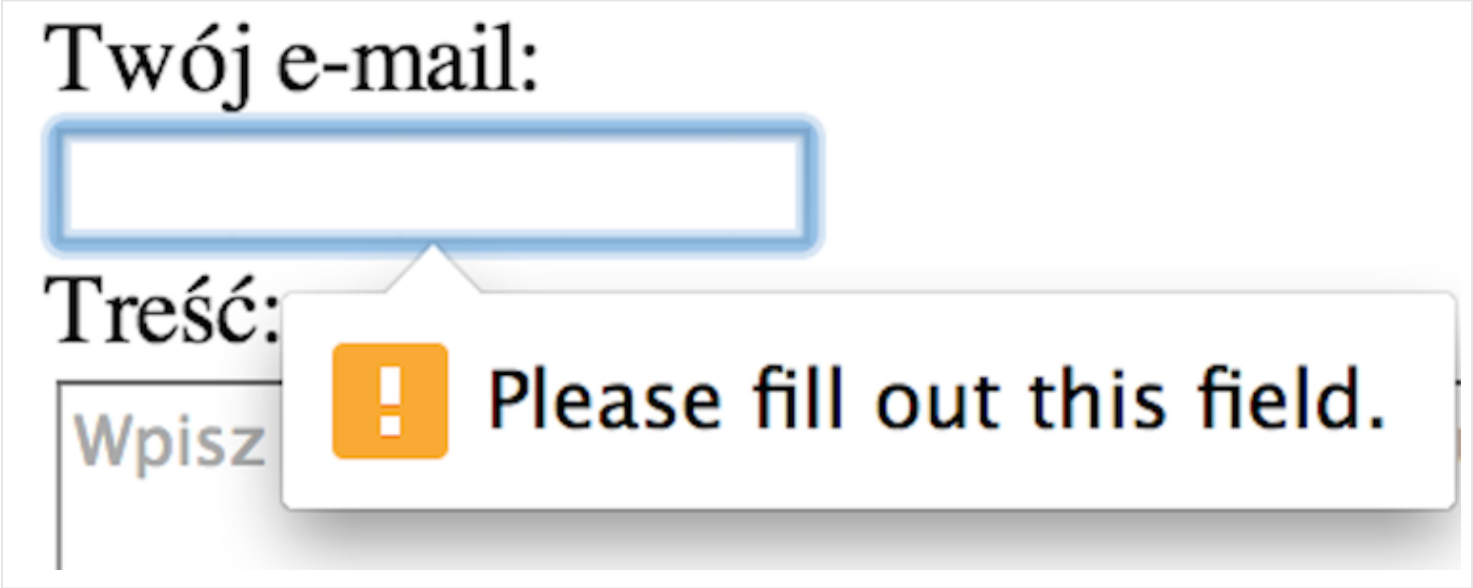
Jak widzisz, w środku <textarea> pojawił się dodatkowy opis, który odpowiada temu, co umieściliśmy w atrybucie placeholder. Spokojna głowa, tekst ten zniknie po kliknięciu , a Ty będziesz mógł wpisać własną treść. Gdybyś zostawił to pole puste również nic nie nie stanie – zawartość z "placeholder" nie zostanie wysłana dalej, służy ona tylko do podpowiedzi dla użytkownika. Atrybutu tego możesz używać również z elementami

<input>. Jego zadaniem jest zasugerowanie użytkownikowi formatu odpowiedzi. Np. w polu dotyczącym maila możemy wpisać placeholder="jan@kowalski.pl".

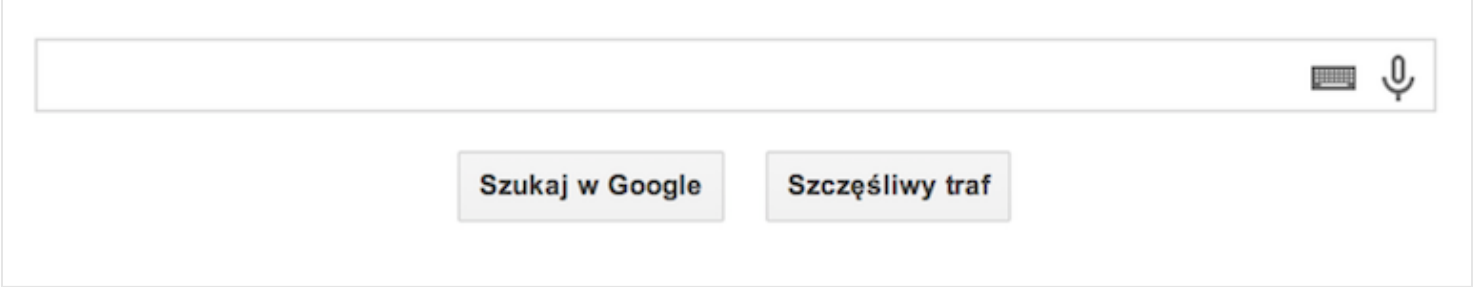
Kolejnym, bardzo przydatnym atrybutem jest required, który dodajemy do elementów formularza bez żadnej wartości. Na przykład dla e-maila będzie to wyglądało tak:

```
<input type="email" id="user-email" name="user-email" required>
```

Teraz, gdy zostawimy to pole puste, podczas wysyłania formularza ujrzymy błąd:



Tego typu atrybutów jest więcej, wymieniłem tylko dwa, najbardziej pomocne. W swojej pracy jako twórca stron internetowych na pewno bardzo szybko nauczysz się tych najważniejszych, bo będzie tego wymagał już pierwszy projekt – w zasadzie większość stron zawiera przynajmniej jeden formularz. Na przykład w wyszukiwarce Google, główne miejsce do wpisywania szukanej przez Ciebie frazy wygląda tak:



A zakodowane zostało jak poniżej (stan na listopad 2013):

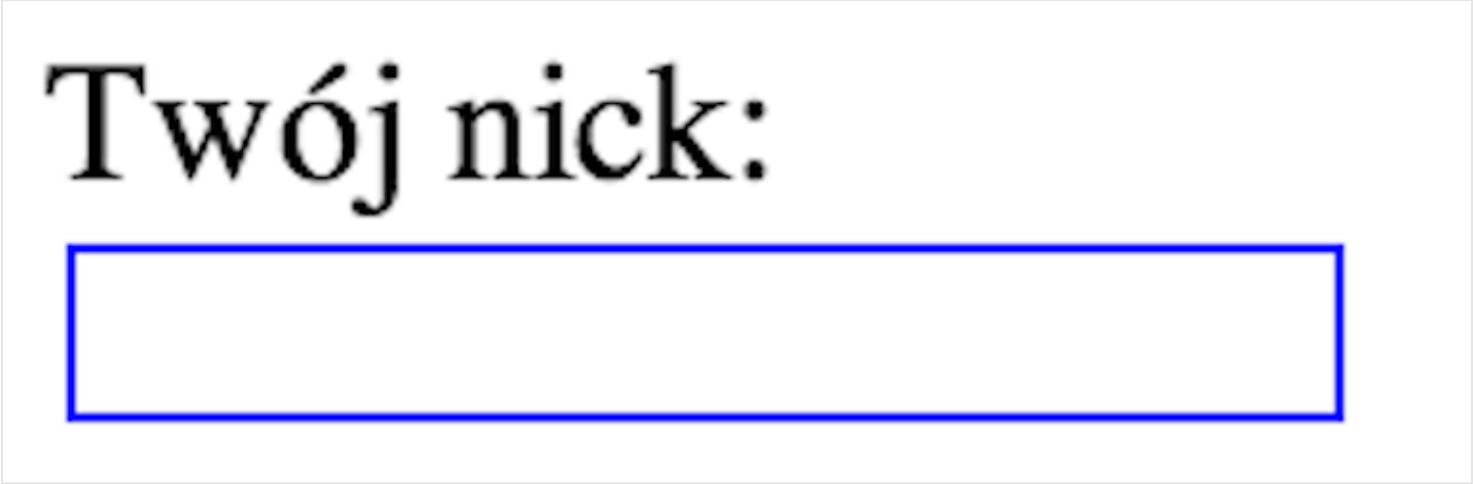
```
<input id="gbqfq" class="gbqfif" name="q" type="text"
autocomplete="off" value="" dir="ltr" spellcheck="false">
```

Jak widzisz, element <input> ma nieco więcej atrybutów niż zwykle używaliśmy. Spróbuj odnaleźć w specyfikacji HTML znaczenie każdego z nich. Nie będzie to trudne. Z pewnością pomoże Ci tutaj wyszukiwarka Google ;-)

Atrybuty są pomocne również do nadawania elementom stylów CSS. Na przykład każdy element <input> z atrybutem type równym text można wyszukać w selektorze następująco:

```
input[type="text"] {
  border: 1px solid blue;
}
```

Niniejszym każdy element <input> z atrybutem type równym text dostanie niebieską ramkę:



Analogicznie możemy postąpić dla adresu e-mail.

```
input[type="text"] {
  border: 1px solid blue;
}

input[type="email"] {
  border: 1px solid blue;
}
```

Wyglądać to będzie teraz tak:

Niniejszym stworzyliśmy dwa selektory w CSS, które nadają taką samą wartość, a mianowicie border. Gdybyśmy chcieli zastosować coś takiego dla większej liczby elementów, prawdopodobnie skopiowałbyś jeden z tych selektorów i go trochę zmodyfikować. W efekcie mielibyśmy więcej niezależnych selektorów robiące to

samo (czyli ustawiających obramowanie). W CSS można skrócić taką operację, podając selektory po przecinku, tworząc ich grupę:

```
input[type="text"],
input[type="email"] {
  border: 1px solid blue;
}
```

Jak widzisz, zaoszczędziliśmy sporo miejsca i zyskaliśmy o wiele większą czytelność.

Można grupować nieskończenie wiele selektorów, na przykład tak (użyłem selektorów z poprzednich przykładów):

```
.news-item,
article p,
#main header h1 {
  padding: 5px;
  margin: 5px;
}
```

I tak dalej.

Zawsze miej na uwadze to, by grupować selektory, jeśli pewne wartości powtarzają się dla kilku z nich. Wtedy będziesz miał w jednym miejscu kod dla kilku elementów i nie będziesz zmuszony wyszukiwać ich osobno w pliku CSS.

ROZDZIAŁ 12

Układ kolumnowy

W poprzednim przykładzie poznaliśmy między innymi znaczniki . Wiesz już, że nie zawierają one żadnej treści, natomiast służą przede wszystkim jako elementy, dzięki którym możemy wprowadzić zmiany w wyglądzie naszej strony - podzielić je na większe sekcje, wiersze, czy zrobić na przykład układ kolumnowy.

Do tej pory udało nam się uzyskać widok artykułu, stworzyć menu oraz formularz do komentowania. Umieśćmy je wszystkie w jednym, 3-kolumnowym układzie (layoucie). Niech menu znajdzie się po lewej, w środku artykuł wraz z formularzem, a z prawej na przykład lista materiałów powiązanych z artykułem. Sprawmy, by cała strona miała maksymalnie 960 pikseli szerokości i była wyśrodkowana.

Spróbujmy znów podzielić to, co widzimy na grafice, na funkcjonalne części które odwzorujemy w HTML-u. Mamy więc lewą kolumnę z menu, środkową z główną treścią strony oraz prawą, z krótkim tekstem. Całość zaś musi być wyśrodkowana i mieć maksymalnie 960 pikseli szerokości. Bardzo dobrym pomysłem jest stworzyć dodatkowo element, który będzie rodzicem dla tych trzech kolumn.

Będzie on przydatny dlatego, aby ustaić szerokość strony na 960 pikseli. Nie posiadając takiego elementu nadrzędnego ciężko byłoby też wyśrodkować wszystkie 3 kolumny.

Niech naszym kontenerem będzie zwykły , ponieważ nasz "pojemnik na kolumny" nie pełni on żadnej, semantycznej funkcji w dokumencie. Dzięki temu, że będzie on elementem rodzicem, będziemy mogli nadać mu własności CSS takie jak szerokość i wyśrodkowanie, niniejszym wszystkie inne kolumny w nim się znajdujące też będą wyśrodkowane i będą mieścić się w obrębie 960pikseli.

```
<div class="main-container"></div>
```

Nadałem mu przy okazji klasę "main-container", która odwzorowuje, że jest to główny kontener na inne elementy na naszej stronie.

Następnie należy umieścić menu. Będzie ono stanowiło boczną kolumnę. Jak pamiętamy, do nawigacji na stronie używamy <nav>, który ostylujemy tak, by pełniło rolę bocznej, lewej kolumny. Użyjmy więc <nav class="site-menu">:

```
<div class="main-container">
  <nav class="site-menu"></nav>
</div>
```

Idąc dalej, umieszczamy kontener na główną treść, która będzie w środkowej kolumnie :

```
<div class="main-container">
  <nav class="site-menu"></nav>
  <div class="main-content"></div>
</div>
```

Następnie pozostała nam prawa kolumna <aside class="sidebar">.

```
<div class="main-container">
  <nav class="site-menu"></nav>
  <div class="main-content"></div>
  <aside class="sidebar"></aside>
</div>
```

Teraz przeanalizujmy, cały czas zerkając na obrazek, co powinna zawierać lewa kolumna. Jak nazwa wskazuje, powinno to być nasze menu z poprzedniego rozdziału.

```
<div class="main-container">
  <nav class="site-menu">
    tu wstaw kod z menu
  </nav>
  <div class="main-content"></div>
  <aside class="sidebar"></aside>
</div>
```

W środku `<nav>` celowo nie wstawiłem więcej kodu, ponieważ chciałem zachować jego czytelność. Kiedy będziemy już gotowi, by opublikować naszą stronę, wystarczy skopiować kod menu i wkleić go pomiędzy `<nav></nav>`.

W środkowej kolumnie będzie natomiast artykuł i formularz do komentowania.

```
<div class="main-container">
  <nav class="site-menu">
    tu wstaw kod z menu
  </nav>
  <div class="main-content">
    <article>tu wstaw kod artykułu</article>
    <form>tu wstaw kod formularza</form>
  </div>
  <aside class="sidebar"></aside>
</div>
```

Następnie w prawej kolumnie umieścimy poboczny element, niezwiązany z główną treścią. Niech będzie oznaczony zwykłym divem:

```
<div class="main-container">
  <nav class="site-menu">
    tu wstaw kod z menu
  </nav>
  <div class="main-content">
    <article>tu wstaw kod artykułu</article>
    <form>tu wstaw kod formularza</form>
  </div>
  <aside class="sidebar">
    <div>tu wstaw kod prawej kolumny</div>
  </aside>
</div>
```

Wygląda na to, że nasz kod HTML jest gotowy. Po przygotowaniu HTML-a pora na CSS. Naszym pierwszym zadaniem jest ustawić maksymalną szerokość dla głównego diva z klasą main-container:

```
.main-container {
  max-width: 960px;
}
```

Posłużyła nam do tego właściwość max-width. Sprawi ona, że cokolwiek by się nie stało, szerokość całego pojemnika z klasą main-container nie może być większa niż 960 pikseli.

Następnie wycentrujmy blok. Służy do tego ustawienie automatycznych marginesów:

```
.main-container {
  max-width: 960px;
  margin: auto;
}
```

Dzięki temu przeglądarka weźmie całą wolną przestrzeń pomiędzy .main-container, podzieli ją na dwa i ustawi wynik po lewej i prawej, sprawiając, że element zostanie wycentrowany, bo jego lewy i prawy margines będą miały taką samą wartość. Podobnie, jak na poniższym zrzucie ekranu:

Mając za sobą kod odpowiedzialny za główny kontener, sprawmy teraz, by nasze trzy kolumny były ustawione obok siebie.

Niech pojemnik z menu ma 20% dostępnej szerokości. Robi się to po prostu podając wartość w procentach:

```
.site-menu {
  width: 20%;
}
```

Podobną szerokość ustawmy dla prawej kolumny:

```
.sidebar {
  width: 20%;
}
```

Jako, że i lewa, i prawa kolumna mają te same właściwości, zgrupujmy je:

```
.site-menu,
.sidebar {
  width: 20%;
}
```

Teraz zajmijmy się środkową kolumną. Niech jej szerokość wynosi pozostałe 60%, ponieważ dwie boczne kolumny zajmują w sumie 40%.

```
.main-content {
  width: 60%;
}
```

Niestety, nasze kontenery nadal wyświetlają się jeden pod drugim. Aby ustawić je obok siebie, musimy nadać im specjalną właściwość CSS float. Używamy ją do tego, aby powiedzieć przeglądarce, że chcemy, aby dany element był zbliżony do lewej bądź prawej krawędzi kontenera, w którym się znajduje. Gdy w jednym kontenerze ustawimy kilku elementom, by "pływały" do lewej krawędzi, ustawią się one jeden obok drugiego. My sprytnie wykorzystamy ten fakt.

Dzieje się tak dlatego, że lewa kolumna "doklei" się do lewej krawędzi głównego pojemnika, z kolei środkowa do lewej kolumny, ponieważ przy samej krawędzi rodzica (którym jest główny pojemnik .main-container) nie ma już miejsca - zajmie go właśnie lewa kolumna - kto pierwszy, ten lepszy!

Zanim przejdziemy do użycia float w CSS, przedstawię Ci najpopularniejszy przypadek użycia tej własności. Wyobraź sobie, że masz obrazek ustawiony bezpośrednio w tekście:

```
<p><img src="cat.jpg" alt="angry cat is watching you">Lorem
ipsum dolor sit amet, consectetur adipiscing elit. Ut dui metus,
commodo vitae sem vel, tempus pellentesque nunc. Cum sociis
natoque penatibus et magnis dis parturient montes, nascetur
ridiculus mus.  Etiam consequat, nisi ac cursus mattis, mi eros
lacinia tortor, nec pellentesque ligula quam mattis nulla.</p>
```

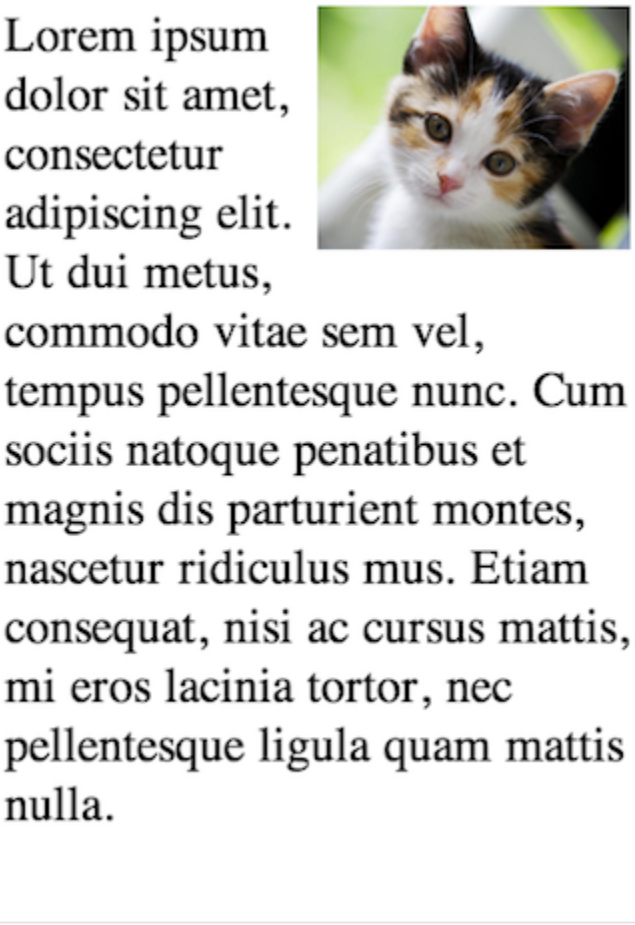
Domyślnie taki obrazek wyświetlałby się tak:

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut dui metus, commodo vitae sem vel, tempus pellentesque nunc. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam consequat, nisi ac cursus mattis, mi eros lacinia tortor, nec pellentesque ligula quam mattis nulla.

Dzieje się to dlatego, że tag <img> jest elementem liniowym i po prostu wstawiliśmy go tak samo jak inny tekst, więc wyświetla się w miejscu, gdzie go umieściliśmy, trzymając się wysokości linii tekstu.

Gdy jednak ustawimy dla obrazka float na wartość right, widzimy już coś znacznie lepszego.

```
img {
  float: right;
}
```



Obrazek zaczął "pływać" do prawej krawędzi elementu, w którym został umieszczony – w tym wypadku do krawędzi akapitu <p>. Pozostałą wolną przestrzeń z lewej, jaka została po "odpłynięciu" naszego obrazka, wypełnia natomiast tekst.

Spróbujmy teraz ustawić float na left:

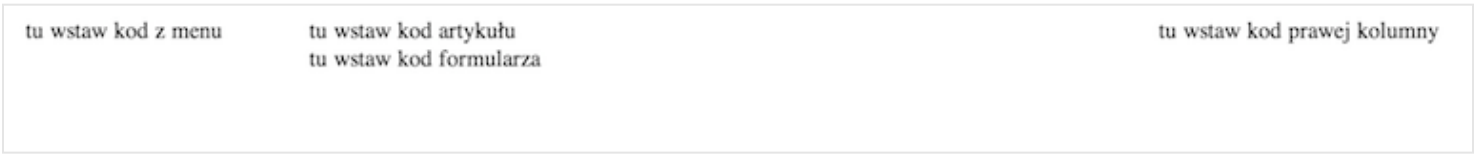
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut dui metus, commodo vitae sem vel, tempus pellentesque nunc. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam consequat, nisi ac cursus mattis, mi eros lacinia tortor, nec pellentesque ligula quam mattis nulla.

Jak widać, obrazek zaczął "pływać" do lewej krawędzi akapitu, a z prawej zaczął go oblewać naturalnie tekst.

Wracając do przykładu z kolumnami, sprawmy, by wszystkie dążyły w swojej pozycji do lewej krawędzi kontenera:

```
.site-menu,
.main-content,
.sidebar {
  float: left;
}
```

Bingo! Zobacz, jak teraz wyświetla to przeglądarka:

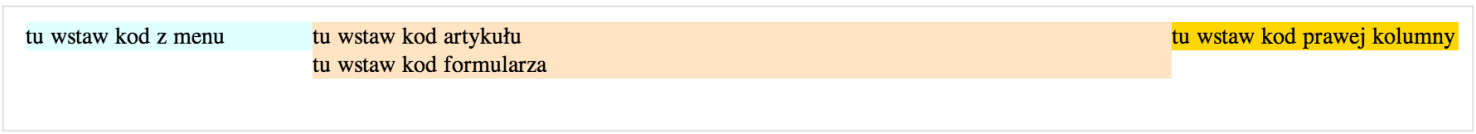


Zapytasz zapewne, dlaczego się tak stało. Otóż spróbuj przeanalizować sobie, jak zachował się obrazek, po ustawieniu float na left. Został on wyświetlony jak najbliżej lewej krawędzi, a pozostałą przestrzeń z prawej zaczął wypełniać tekst. Tutaj dzieje się coś podobnego - elementy mają ustawiony "float" na "left", a więc będą dążyć do lewej krawędzi. Pierwszy ustawi się tuż przy niej, kolejny wykorzysta wolną dostępną przestrzeń obok pierwszego. I tak jeden po drugim, dzięki czemu masz wrażenie, że są ustawione obok siebie.

Znakomicie! Uzyskaliśmy właśnie układ kolumnowy. Aby bardziej zobrazować, o co chodzi, dodajmy jeszcze kolory tła dla każdej z kolumn:

```
.main-content {
  width: 60%;
  background-color: Bisque;
}

.site-menu {
  background-color: LightCyan;
}

.sidebar {
  background-color: Gold;
}
```

Efekt jest jak najbardziej zadowalający. Doskonale pokazuje, że udało się podzielić główny kontener na kolumny:



Nasz kod CSS wygląda z kolei tak:

```
.main-container {
  max-width: 960px;
  margin: auto;
}

.site-menu,
.main-content,
.sidebar {
  float: left;
}

.site-menu,
.sidebar {
  width: 20%;
}

.main-content {
  width: 60%;
  background-color: Bisque;
}

.site-menu {
  background-color: LightCyan;
}

.sidebar {
  background-color: Gold;
}
```

W ten sposób stworzyliśmy prosty układ z trzema kolumnami. Teraz wystarczy wkleić pozostały kod HTML i CSS z poprzednich ćwiczeń. Niech to będzie Twoje zadanie!

ROZDZIAŁ 13

Jak dalej się rozwijać?

Właśnie zbudowałeś swoją, pierwszą pełnoprawną stronę internetową. Podzieliłeś umiejętnie stronę na trzy kolumny, dodałeś menu i formularz do komentowania. Kolejne kroki jakie wykonasz na swojej drodze postawisz już sam. Zapewne zapytasz teraz, co dalej? Mam dla Ciebie kilka porad.

Przede wszystkim wybierz swoją ulubioną stroną internetową i postaraj się ją odtworzyć, pisząc własnoręcznie kod od nowa. Podziel jej elementy na pojedyncze części według ich funkcjonalności. Najpierw stwórz kod HTML, a dopiero potem dodaj CSS. Zawsze pamiętaj o tej kolejności. Niezwykle ważny jest semantyczny kod HTML. Dzięki niemu inni programiści front-end (tak nazywają się osoby, zajmujące się warstwą interfejsu użytkownika, a więc językami HTML, CSS i JavaScript) będą mogli łatwo wgryźć się w Twój kod. Co więcej, Twój trud wynagrodzą wyszukiwarki internetowe - dobrze ustrukturyzowana treść pozycjonuje się w nich o wiele lepiej. Ponadto dobry HTML jest bardzo ważny dla różnych czytników, które nie obsługują CSS-a i wyświetlanie tekstu mogą oprzeć jedynie na znacznikach. Im lepiej one dobrane, tym większy komfort czytania – nikt nie chciałby przecież, by nagłówek wyświetlał się tak, jak zwykła treść.

Po skończonej pracy sprawdź, jak kod napisali autorzy oryginału. Poznawaj nowe techniki, dopytuj, bądź krytyczny.

SCHEMATY

Bardzo ważne jest też uświadomienie sobie, że programowanie front-endu to w większości praca oparta na schematach. Każda działka (HTML, CSS oraz JavaScript, który

zapewne poznasz w przyszłości) ma swoje własne niespodzianki i tajemnice. W każdej z nich czyha na Ciebie mnóstwo pułapek. W HTML jest to na przykład tzw "classitis", czyli nadmiarowe dodawanie niepotrzebnych klas. Jest też zjawisko "divitis", czyli nadużywanie elementów `<div>`, kiedy można je zastąpić czymś bardziej semantycznym. W takie pułapki będziesz wpadał bardzo często. Wychodząc z nich obronną ręką zdobędziesz doświadczenie, które potem zaprocentuje. Nie będzie już dla Ciebie problemem to, jak ułożyć parę elementów obok siebie. Pójdziesz dalej, poszukasz sposobu, jak zrobić to w najmniejszej liczbie linii kodu. Jak ulepszyć semantykę HTML, jak sprawić, by przeglądarka wyświetliła to najszybciej. Będziesz pracować nad własnym fachem popełniając błędy i ucząc się na nich. Poznasz wzorce postępowania w danych przypadkach, które społeczność koderów HTML i CSS zdążyła wypracować przez kilkanaście lat swojego istnienia.

TRZYMAJ RĘKĘ NA PULSIE!

Po przerobieniu tej książki, warto byś sięgnął po kolejną, która opisuje wszystkie znaczniki HTML i CSS – może to być na przykład Wstęp do HTML5 i CSS3. Ja pokazałem Ci tylko sposób myślenia, pora byś nauczył się reszty tagów HTML i własności CSS. Wiedz, że to dość dynamiczne języki. Z HTML-em jest ten problem, że nie zmienia się on rewolucyjnie – często zmieniają się jednak detale i pojedyncze tagi. Warto, byś śledził grupy W3C i WHATWG, które pracują nad rozwojem tego języka. Zdarza się bowiem, że jakiś tag HTML nagle przestaje być elementem specyfikacji, więc nie należy go używać. Innym razem zmienia się jego przeznaczenie. Trzymaj rękę na pulsie!

CSSOWE SZTUCZKI

CSS to z kolei kopalnia wiedzy, wzorców i sposobów obejścia różnych problemów. Poznaliśmy jego podstawy i zasady działania. Jak zauważyłeś, jest to prosty język.

Tworzysz selektor, dzięki któremu odnosisz się do tego, jak mają wyglądać dane elementy HTML. Aby zmodyfikować ich wygląd używasz jakichś własności np. height, gdy chcesz zmienić wysokość. Tych właściwości jest ich o wiele więcej. Możesz dodawać tła gradientowe, zaokrąglone rogi (poprzez border-radius), przeźroczystość (własność opacity), cienie tekstu (text-shadow), cienie elementów blokowych (box-shadow) i wiele więcej. Przejrzyj dokumentację CSS-a i poczytaj inne artykuły w sieci o CSS.

Bardzo istotne we front-endzie jest to, aby strony wyświetlały się tak samo w każdej przeglądarce. W niedalekiej historii był to dość duży problem. Wiele przeglądarek na rynku interpretowało standardy po swojemu (prym wiódł w tym Internet Explorer 6, który jest dziś owiany mroczną sławą). Dlatego też przez długi okres składanie stron polegało na wynajdowaniu różnych trików, które omijałyby niedogodności w danej przeglądarce. Najlepiej, gdyby były one semantyczne i nieinwazyjne. Dlatego też wertując archiwalne materiały (powiedzmy z 2006 roku) znajdziesz wiele technik takich jak faux columns (do uzyskania kolumn tej samej wysokości), PNGFix (wyświetlanie przeźroczystego obrazka PNG) czy clearfix, które znajdowały odpowiedź na problemy przeglądarek z tamtych lat. Część z nich jest nadal używana, jednak wiele odeszło w zapomnienie, ponieważ naprawiono błędy w popularnych przeglądarkach internetowych. Przebywając i czytając o CSS w internecie spotkasz je nie raz. Warto wtedy zapytać bardziej doświadczonego kolegę, czy jeszcze używa się danej techniki.

HISTORIA PRZEGLĄDAREK

Przeglądarki poprzedniej dekady miały dużo wad. Chyba największą było to, że nie aktualizowały się automatycznie, a więc gdy chciano dodać nowe opcje w CSS, musiano czekać kilka, kilkanaście miesięcy na nową wersję przeglądarki. Wyobraź sobie, że nie możesz zastosować tła składającego się z gradientu, używając wyłącznie CSS-a. Tak, w tamtych latach to była codzienność. Trzeba było przygotować obrazek w postaci

gradientu, wgrać go na serwer i ustawić go w CSS jako tło danego elementu. Robiło się to tak:

```
.news {
    background: url(gradient.jpg) no-repeat;
}
```

Dziś z kolei ustawianie tła gradientowego jest bardzo łatwe, definiuje się je w postaci własności linear-gradient:

```
.news {
    background: linear-gradient(to bottom, rgba(30,87,153,1) 0%,rgba(
}
```

PREFIKSOWE DZIWOLĄGI

Jak widać, przeglądarki internetowe cały czas ewoluują, a standardy pisania stron się zmieniają. Na przykładzie CSS-a jest to widoczne chyba najbardziej. Wyobraź sobie, że cały czas pojawiają się propozycje nowych opcji w CSS. Instytucja taka jak W3C, stojąca na straży standardów, nie może akceptować każdego pomysłu bez jego przetestowania i konsultacji. Jak to w życiu, być może istnieje lepsze rozwiązanie, może coś nie ma sensu i tak dalej. Wszystko trzeba zbadać. To trwa. Dlatego też producenci przeglądarek często wyprzedzają decyzje W3C, implementując nowe własności CSS poprzez dodanie do ich nazw przedrostki, które jednoznacznie identyfikują je z daną przeglądarką. Na przykład Chrome, gdy wprowadza jakąś eksperymentalną opcję, nazywą ją w ten sposób:

```
-webkit-nazwa
```

Wtedy, aby użyć takiej własności, musimy w CSS napisać:

```
.news {
    -webkit-nazwa: 20px;
}
```

Oczywiście -webkit-nazwa nie ma sensu i wymyśliłem to na potrzeby przykładu. Z istniejących własności można wymienić np. -webkit-box-sizing. Bywa więc, że musimy w CSS obsłużyć kilka wersji jednej własności CSS dla każdej z liczących się przeglądarek:

```
.news {
    -webkit-box-sizing: border-box; /* Chrome */
    -moz-box-sizing: border-box; /* Firefox */
    -ms-box-sizing: border-box;  /* Interner Explorer */
    box-sizing: border-box;
}
```

W powyższym przykładzie musiałem użyć aż czterech linijek. Najpierw dla Chrome, potem dla Firefoxa, następnie dla Internet Explorera, a na końcu zastosowałem nazwę bez prefiksu, na wypadek, gdyby któraś z przeglądarek przestała wspierać prefiksową wersję. Dzieje się tak, gdy W3C ostatecznie uznaje daną własność za prawidłową i wchodzi ona do standardów.

KOLORY W CSS

CSS to dzisiaj naprawdę spora kopalnia pomysłów i możliwości. Same kolory można wyrazić w kilku różnych formatach. Jak zapewne zauważyłeś, w tej książce używaliśmy kolorów zapisanych w formie słownej. Wybrałem ten format, by nieco uprościć sprawę. Wiedz jednak, że najbardziej popularnym formatem kolorów jest format heksadecymalny, czyli na przykład kolor czarny zapisywany jest tak: #000000. Innym formatem jest rgb. Wtedy kolor czarny zapiszemy tak: rgb(0, 0, 0). Chcesz, aby Twoje tło było czarne i w połowie przeźroczyste? Nie ma problemu, użyjesz rgba – rgba(0, 0, 0, 0.5). W parze z selektorami używamy tego tak:

```
.news {
    color: #000000;
}
```

Wtedy wszystkie elementy .news będą miały czarny font. Kolory w tym formacie uzyskasz w każdym programie graficznym. Jeśli chcesz przetestować różne warianty bez ich instalowania, możesz wejść na przykład na ColorPicker.com i dowolnie eksperymentować. Szybko da się zauważyć o wiele więcej możliwości w stosunku do kolorów zapisanych słownie, prawda?

RESPONSIVE WEB DESIGN

Będąc przy CSSie, ostatnio bardzo modny jest temat tzw. Responsive Web Design. Chodzi tutaj o to, by strona wyglądała inaczej w zależności od rozdzielczości. Często na urządzeniach mobilnych z małym ekranem nie ma sensu pokazywać całej strony, wystarczy skupić się na treści głównej i ukryć pozostałe kolumny. Do tego służy właśnie RWD – aby komfort przeglądania strony był na jak najwyższym poziomie. Bez zastosowania technik Responsive Web Design często użytkownik (szczególnie korzystając ze smartphone'a z dość małym ekranem) musi przybliżać całą stronę. Nie jest to wygodne. W RWD piszesz w CSSie mniej więcej tak: dla rozdzielczości ekranu do 320px użyj takich i takich oto reguł. Zapisać to można tak:

```
@media screen and  (max-width: 320px) {
    .news {
        display: none;
    }
}
```

Użyliśmy tutaj tzw. "media queries". Dzięki temu na urządzeniach wyświetlaczem do 320px szerokości, elementy .news będą ukryte (display: none).

NARZĘDZIA

Przeglądarki internetowe, oprócz stałego implementowania nowych opcji w HTML i CSS, zaczęły od pewnego czasu oferować również tzw. narzędzia developerskie, czyli wbudowane w przeglądarkę widgety, dzięki którym możesz m.in. obejrzeć swój kod HTML i zobaczyć, jakie style ustawiła im przeglądarka. A to tylko wierzchołek góry lodowej możliwości, jakie dają. Aby uruchomić je w Google Chrome, kliknij prawym klawiszem myszki na stronie i wybierz "Zbadaj element". Otworzy się wtedy nowy panel z przydatnym zestawem narzędzi o jakim właśnie wspomniałem. Spędź przy nich dłuższą chwilę – prawie wszystko można w nich zmienić, podejrzeć, zbadać. Podobne narzędzie znajdziesz w Firefoxie i Internet Explorerze.

FILOZOFIA

Tego typu smaczków jest więcej. Gdyby spróbować je opisać, ta książka mogłaby zapewne mieć kilkaset stron. Chciałem Ci tego zaoszczędzić. Wydaje mi się, że na początek najważniejsze jest, by poznać filozofię pisania HTML-a i CSS-a, a dopiero potem poszczególne techniki, pozwalające uzyskać dane efekty. Poznając dalej te technologie zapewne dowiesz się wkrótce o innych możliwościach uzyskania trójkolumnowego layoutu czy formularza, gdzie każdy wiersz jest pod sobą. To naturalne, ponieważ technologie te są na tyle elastyczne, że możliwości jest bardzo wiele. Byle były zgodne ze standardami! Jak więc być na czasie i dalej się rozwijać?

MIEJSCA W SIECI

Zadawaj pytania. Bądź aktywny na forach internetowych i grupach na Facebooku. Odwiedź forum.webhelp.pl, ForumWeb oraz grupę HTML5 i CSS3 - pierwsze kroki. Gdy poznasz JavaScript, niezbędna będzie wizyta na JS News na Facebooku. To miejsca, gdzie dyskutują początkujący i doświadczeni developerzy.

Uczęszczaj na konferencje, gdzie możesz spotkać osoby pracujące nad kształtem dzisiejszego internetu (choćby z Mozilli czy z Google). W Polsce organizuję Front-Trends, która trwa 3 dni. Spotkasz na niej programistów z Polski i ze świata, którzy chętnie podzielą się z Tobą wiedzą i doświadczeniem.

Bądź częstym gościem różnych meetupów technologicznych, które są **darmowe**. W Polsce bardzo prężnie działa meet.js, który odbywa się w 8 miastach w Polsce. Występują na nich prelegenci, którzy chcą dzielić się wiedzą z zakresu HTML5, CSS3 i JS. Każde wystąpienie trwa od 10 do 20 minut. Każdy może się zgłosić, także Ty! Oprócz tego, warte odwiedzenia są również DevMeetingi, darmowe warsztaty z technologii webowych.

Próbuj czytać blogi o HTML i CSS. Wejdź na mojego bloga - znajdziesz tam wiele artykułów na temat HTML, CSS i JavaScript. Sprawdź o czym piszą na CSS Tricks, które jest świetnym centrum artykułów o różnych trikach w CSS. Inne wartościowe blogi i serwisy to:

Zagraniczne blogi:

- Adactio.com
- Anna Debenham
- CSSWizardry.com
- Divya Manyan
- Estelle Weyl
- Lea Verou
- Mathias Bynens
- Paul Irish
- Simurai.com
- Tab Atkins

Zagraniczne serwisy:

- Smashing Magazine

- WebPlatform Daily

Polskie blogi:

- Andrzej Mazur
- Michał Maćkowiak
- Piotr Nalepa
- Zofia Korcz

Polskie serwisy:

- CSS3.pl
- JavaScript.pl
- webmastah.pl
- webhelp.pl

Inne:

- Poradnik HTML5 Comandeera

Próbuj coraz to nowych konceptów. Zainteresuj się bardziej poniższymi tematami. Warto je wygooglować:

- box model
- responsive web design
- progressive enhancement
- mobile first
- grid system
- css frameworks
- semantic web
- accessibility
- WAI-ARIA

Czytaj książki. Na początek polecam Dive Into HTML5 i HTML5 For Web Designers. Przejrzyj moją prezentację na temat tego, jak łatwe i przystępne są technologie webowe. Wkrótce zapewne wydam kolejny poradnik – **zapisz się do newslettera**, by otrzymać informacje o nowych materiałach, które pomogą Ci w nauce HTML5, CSS3 i JavaScript:

Poznaj więcej tagów HTML5 i próbuj nowych rzeczy z CSS3. Sprawdź, co jest obecnie możliwe we współczesnych przeglądarkach na Caniuse.com. Kiedy będziesz chciał sprawdzić, jak działa dana konstrukcja HTML-a lub CSS-a, sprawdzaj szczegóły na Mozilla Developer Network. Nie ufaj innym stronom, uważaj na w3schools, które mimo, że nazywa się podobnie do W3C, często zawiera niesprawdzone informacje. Ten rynek szybko ewoluuje, bądź na czasie. Załóż konto na Twitterze i obserwuj na nim osoby ze świata web developmentu. Dodaj też konto na GitHubie, publikuj tam kod i patrz, jak robią to inni. Próbuj różnych rzeczy na codepen.io i twórz własne. Sprawdź JSBin.com, gdzie możesz pisać kod i automatycznie dostać podgląd tego, jak wyświetliłaby go przeglądarka. Jest też w sieci wiele serwisów, które nauczają technologii, choćby Codecademy.com. Wypróbuj je.

Postaw na język angielski, to oficjalny język sieci. Dzięki niemu wszystko stanie się łatwiejsze. Zawodowo i prywatnie.

Najważniejsze jednak, byś nie przestawał się rozwijać.

PS Jeśli podobała Ci się książka, polub mój profil na Facebooku i podziel się opinią!