HTML i CSS

Przewodnik dla początkujących

SOLIDNE PODSTAWY KODOWANIA
I PROJEKTOWANIA RESPONSYWNYCH
STRON INTERNETOWYCH

David DuRocher

Helion

Tytuł oryginału: HTML and CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering The Fundamentals of Modern Web Design

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-8322-655-2

Translated and published by *Helion S.A.* with permission from ClydeBank Media LLC. This translated work is based on *HTML and CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering The Fundamentals of Modern Web Design* by David DuRocher © 2020 by ClydeBank Media LLC. All rights reserved. ClydeBank Media LLC is not affiliated with *Helion S.A.* or responsible for the quality of this translated work. Translation arrangement managed RussoRights LLC on behalf of ClydeBank Media LLC.

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

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

WPROWADZENIE 1
HTML i CSS są wszędzie 1
Moja historia 1
Dlaczego warto znać HTML i CSS 3
Kto może skorzystać na lekturze tej książki 4
Praca, w której warto znać HTML i CSS 5
Struktura i treść książki 7
Środowisko pracy 8
Podstawowy zestaw narzędzi 13
Zwięzły przewodnik po Visual Studio Code 15
Jak najlepiej wykorzystać tę książkę 17

CZĘŚĆ I PRZYGOTOWANIA

| 1 | PODSTAWY I OGÓLNY OBRAZ 21
Podstawowa struktura HTML i CSS 21
Relacja między HTML-em a CSS 25
Porównywalne i dopełniające się języki 27
Systemy zarządzania treścią 29
Znajomość odbiorcy 30

| 2 | HTML DOGŁĘBNIE 35
Zwięzła historia języka HTML 35
Obecne zastosowania 36
Jak to działa 37
Ważne zmiany i aktualizacje 40
HTML5 41
Przyszłość 42

| 3 | CSS DOGŁĘBNIE 45
Tło 46
Jak to działa 47
Stosowanie stylów do elementów 52
Silniki renderujące 59
Poza siecią 61

CZĘŚĆ II W GŁĄB

| 4 | STRUKTURA HTML 65

Elementy 65
Komentarze 68
Format dokumentu HTML (podstawowa struktura) 69
Zagnieżdżanie 72
Składanie wszystkiego razem 73

| 5 | PODSTAWOWE ELEMENTY HTML 77

Akapity 79
Nagłówki 80
Listy 83
Łącza 85
Obrazy 88
Inne znaczniki 91
Elementy div i span 91
Elementy semantyczne 93
Składanie wszystkiego razem 98

| 6 | STRUKTURA CSS 103

Gdzie „mieszka" CSS 103
Przykładowy plik CSS 106
Selektory 107
Pseudoklasy 112
Pseudoelementy 114
Kaskadowa hierarchia CSS 116

| 7 | USTAWIANIE ROZMIARU ELEMENTÓW I ODSTĘPÓW ZA POMOCĄ CSS 119

Treść 120
Dopełnienie 124
Obramowanie 125
Marginesy 127
Określanie wymiarów bloku 129

| 8 | FORMATOWANIE TEKSTU 131

Czcionka 131
Własność color 136
Cień tekstu 137

Niestandardowe czcionki internetowe — Google Fonts 139
Ćwiczenie praktyczne 140

| 9 | UKŁAD I FORMAT 145
Własność position 145
Elementy pływające 148
Własność display 155
Pasek nawigacyjny 158
CSS Flexbox 160
Dalsze kroki 163

CZĘŚĆ III TECHNIKI ZAAWANSOWANE

| 10 | SZUFLADA ZE SZPARGAŁAMI HTML 167
Indeks górny i indeks dolny 167
Skróty 168
Cytaty blokowe i źródła cytatów 168
Elementy pre i code 170
Znaki specjalne 171
Emoji 172
Audio i wideo 173
Zestawy obrazów 175
Tabele 176
Ramki wewnętrzne 181

| 11 | FORMULARZE HTML 183
Formularze — informacje ogólne 183
Najważniejsze elementy 187
Walidacja HTML5 198
Przetwarzanie danych z formularza za pomocą PHP 201
Składanie wszystkiego razem 204

| 12 | EFEKTOWNE SZTUCZKI 209
Gradienty CSS 209
Duszki 214
Przejścia 218
Transformacje 220
Nakładka modalna bez JavaScriptu 223
Animacja poklatkowa 225
Wartości obliczone 228

| 13 | ZAPYTANIA O MEDIA 233
Struktura 234
Wybór punktów kontrolnych 236
Metaznacznik viewport 237
Symulacja rozmiarów ekranu 238

| 14 | BOOTSTRAP 241
Instalacja Bootstrapa 242
Układ siatki 244
Kolory 248
Komponenty 250
Narzędzia pomocnicze 269
Formularze 272
Typografia 275
Dalsza nauka 278
Co dalej 279

CZĘŚĆ IV ŚRODOWISKO PRACY

| 15 | ORGANIZACJA PRACY 283
Tworzenie projektu i zarządzanie nim 283
Rusztowanie 284
Testowanie i debugowanie 285
Publikacja w internecie 287
Modyfikowanie istniejącego kodu 289

| 16 | GIT 291
Czym jest Git 291
Pobieranie i instalacja Gita 292
Wiersz poleceń 293
Tworzenie witryny internetowej w Gicie 294
Import istniejącej witryny do Gita 295
Proces pracy Gita 295
Gałęzie Gita 296
Automatyczna synchronizacja Gita w produkcji 297
GitHub i GitLab 298

| 17 | CO DALEJ 299
WordPress 299
JavaScript i jQuery 300

Języki backendowe, takie jak PHP i Python 301
David's Perfect Cup 302

PODSUMOWANIE 305

DODATEK A | HOSTING SIECIOWY 307

DODATEK B | FTP 311
Popularne klienty FTP 311
Łączenie się z serwerem FTP 312
Parę dodatkowych uwag na temat FTP 312

DODATEK C | JEDNOSTKI WIELKOŚCI 313
Piksele 313
Procenty 313
Jednostki vw i vh 314
Jednostki em/rem 314

DODATEK D | OPEN GRAPH I METADANE 315

DODATEK E | ROZWIĄZANIA ZADAŃ 317
Pobieranie witryny Kawiarni Helion 317
Wprowadzenie — dodawanie strony „Informacje" 317
Rozdział 4. — dodawanie opisu i tytułu 317
Rozdział 5. — strona Informacje 318
Rozdział 5. — nawigacja 318
Rozdział 8. — wygląd i styl 320
Rozdział 9. — reklama 321
Rozdział 11. — formularz kontaktowy 322
Rozdział 12. — duszki 322
Rozdział 12. — gradienty 325
Rozdział 12. — animacja poklatkowa 326
Rozdział 13. — urządzenia mobilne 327

O AUTORZE 329
David DuRocher 329

SŁOWNICZEK 331

ŹRÓDŁA 333

3

CSS dogłębnie

W rozdziale:

- CSS stylizuje elementy HTML.
- Style można stosować do wszystkich, niektórych lub pojedynczych elementów.
- Nie wszystkie przeglądarki identycznie renderują stylizowaną treść.

Jak napisaliśmy wcześniej, HTML jest narzędziem do organizacji, kategoryzacji i strukturyzacji treści. CSS umożliwia zmianę wyglądu i kształtu tej struktury, aby zapewnić lepsze wrażenia wizualne. Podczas gdy HTML stanowi szkielet **informacji** na stronie, style CSS dodają do niego mięśnie, które podnoszą komfort użytkowania.

Wróćmy do naszego porównania strony internetowej do domu. HTML umożliwia nam określenie liczby pokoi, pięter i mebli. Natomiast CSS pozwala określić wygląd i kształt każdego pokoju, rozmieszczenie mebli w pokojach oraz kolor i styl wszystkiego, co jest wewnątrz, od ścian po podłogi. HTML oznacza elementy, a CSS stosuje do nich reguły stylistyczne za pomocą tych oznaczeń.

Jak napisałem w rozdziale 1., akronim CSS pochodzi od angielskich słów Cascading Style Sheets oznaczających kaskadowe arkusze stylów. Za pomocą tej technologii możemy tworzyć globalne reguły formatujące wszystkie elementy określonego typu. Możemy na przykład utworzyć regułę odnoszącą się do wszystkich akapitów, która zmienia głębokość wcięcia i rozmiar czcionki. Ten styl będzie „spływać" jak kaskada po wszystkich akapitach w dokumencie. CSS umożliwia tworzenie ogólnych reguł stylistycznych, a następnie dodawanie wyjątków w razie potrzeby. Arkusze stylów opisuję bardziej szczegółowo później, a na razie rozwiniemy naszą domową analogię.

Powiedzmy, że chcemy, aby wszystkie okna w naszym domu miały wymiary 1×1,2 m oraz białe ramy. Te informacje możemy wprowadzić do globalnego arkusza stylów okien. Jednak od tej ogólnej reguły mogą być wyjątki. Na przykład okna w łazience będą miały tylko 70 cm szerokości i szare ramy. Możemy to wyrazić za pomocą specjalnych instrukcji w rodzaju „Zastosuj ten alternatywny arkusz stylów do wszystkich okien, które *dodatkowo* są oknami łazienkowymi". Teraz wyobraź sobie, że jedno wybrane okno chcesz mieć

zielone. Musisz wprowadzić odpowiednią informację na jego temat. CSS umożliwia stosowanie takich reguł stylistycznych zarówno w szerokim, jak i wąskim zakresie do dowolnej treści utworzonej w dokumentach HTML.

Wszystkie przeglądarki internetowe mają wbudowany zestaw zasad interpretacji instrukcji obecnych w plikach CSS. W większości przypadków robią to dokładnie tak samo, ale w pewnych obszarach występują między nimi różnice. Nie wszystkie reguły CSS są obsługiwane przez każdą przeglądarkę. Podczas pracy z CSS zawsze dobrze jest sprawdzać wszystkie wyjątkowe albo wymyślne efekty w przeglądarkach, z których najprawdopodobniej będą korzystać użytkownicy.

Tło

W pierwszych latach istnienia języka HTML i przeglądarek internetowych to przeglądarka decydowała o sposobie organizacji i rozmieszczenia kodu HTML. W efekcie każda przeglądarka wyświetlała strony inaczej. Proste rzeczy, takie jak czcionka, rozmiar tekstu czy kolor tekstu, jeśli nie zostały wprost zdefiniowane w kodzie HTML, były wybierane przez przeglądarkę, a nie przez autora dokumentu. Twórcy stron mieli pełną kontrolę nad swoją treścią, ale nie nad jej rozmieszczeniem, szatą graficzną itd.

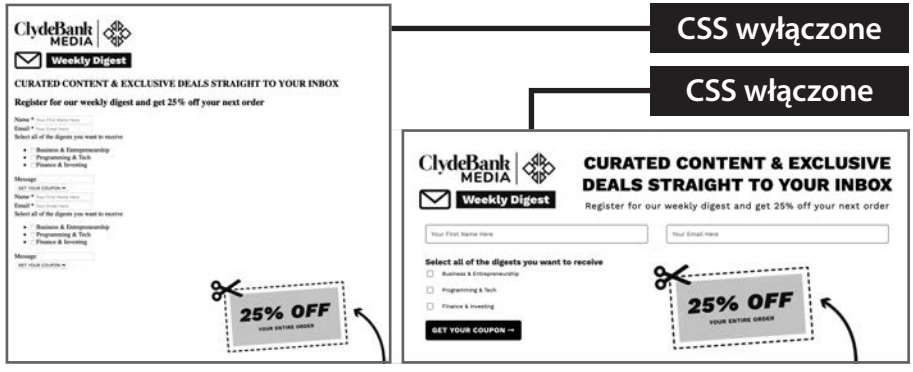
Choć w pierwszych wersjach HTML-a istniała możliwość stylizowania treści, wymagało to używania różnych niezręcznych znaczników (na przykład `<font>` czy `<center>`) w kodzie strony. Obecność tych znaczników powodowała, że kod stawał się rozdęty i trudny do odczytania. Ponadto prowadziło to do niespójności wyglądu różnych stron w obrębie jednej witryny. Większe serwisy, składające się z wielu stron, mogły zawierać różne strony stylizowane w całkiem inny sposób. To sprawiało, że witryny reprezentowały niespójny styl oraz były obciążone niezmiernie skomplikowanym kodem HTML. Ponadto problem pogarszały różnice między przeglądarkami, przez co proces wprowadzania zmian lub aktualizacji w witrynach stawał się nieznośnie czasochłonny i podatny na błędy. Duże aktualizacje wyglądu i stylu witryny często wymagały przerabiania całej treści. W efekcie tych wszystkich trudności związanych z wprowadzaniem zmian strony były bardzo duże i długo się ładowały.

Producenci różnych przeglądarek internetowych, zainspirowani chęcią formatowania stron internetowych jak w tradycyjnych mediach drukowanych, zaczęli publikować „arkusze stylów". Projektanci stron internetowych tworzący strony specjalnie z myślą o konkretnej przeglądarce mogli wykorzystywać te arkusze, aby globalnie nadawać swoim stronom bardziej spójny wygląd i układ. Stopniowo wszystkie najważniejsze przeglądarki zaczęły używać arkuszy stylów i drogą konsensusu wyłoniono standard, który dziś nosi nazwę CSS.

Jak to działa

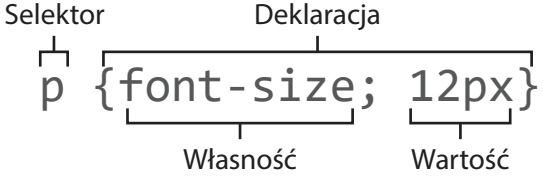
Jeśli HTML dostarcza bloki do budowy stron internetowych, CSS umożliwia określenie ich wyglądu, zachowania i interakcji z innymi blokami znajdującymi się w ich otoczeniu. Podczas gdy HTML określa strukturę treści, CSS zapewnia zestaw reguł umożliwiający określenie wyglądu tych struktur na stronie. Do tego zestawu należą między innymi selektory CSS, które umożliwiają wybieranie elementów HTML do stylizacji (rysunek 3.1).

Rysunek 3.1. Strona witryny ClydeBank Media z wyłączonymi (po lewej) i włączonymi (po prawej) arkuszami stylów



Jak napisałem wcześniej, litera C w nazwie CSS pochodzi od słowa „kaskada", które oznacza, że reguły stylistyczne rozpływają się kaskadowo po witrynie. Reguła CSS składa się z selektora (na przykład p wybiera akapity) i listy odnoszących się do niego własności określających na przykład szerokość czy czcionkę. (rysunek 3.2).

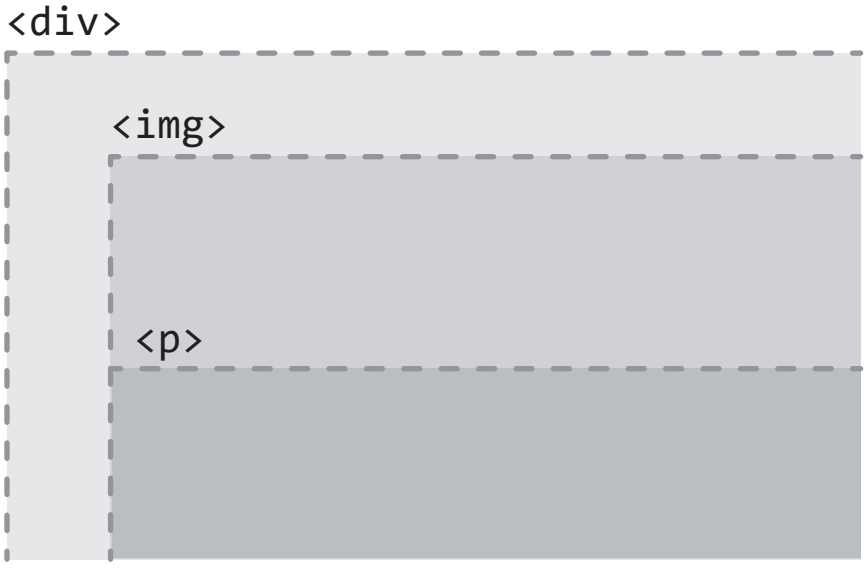
Rysunek 3.2



W CSS występuje ważne pojęcie „dziedziczenia". Aby zrozumieć, jak to działa, musimy przyjrzeć się strukturze naszego kodu HTML. W HTML-u elementy można „zagnieżdżać". Elementy zewnętrzne to „rodzice", a elementy znajdujące się wewnątrz nich to „dzieci". Kiedy stosujemy CSS, elementy potomne dziedziczą wszystkie własności swoich rodziców (takie jak krój i rozmiar czcionki, kolor pisma itd.), ale projektant według swojego uznania może zmienić reguły odnoszące się do elementów potomnych.

Na rysunku 3.3 widać element `<div>`, który jest rodzicem elementów `<img>` i `<p>`. Jeszcze ich nie omawialiśmy, więc nie przejmuj się, jeśli nie wiesz, do czego służą — na razie interesuje nas tylko struktura. Elementy potomne są w całości objęte przez element

nadrzędny i dziedziczą wszystkie jego własności. W związku z tym elementy `img` (obraz) i `p` (akapit) są formatowane zgodnie z regułami dotyczącymi elementu `div`, chyba że zdefiniujemy osobne reguły dla tych elementów potomnych za pomocą odpowiednich selektorów.



GRAFIKA

Rysunek 3.3

Teraz przyjrzymy się trzem sposobom dołączania arkuszy stylów do kodu HTML oraz omówimy ich zalety i wady.

Śródliniowy CSS

Śródliniowe arkusze stylów są dodawane wewnątrz indywidualnych elementów HTML. Ten rodzaj arkuszy najbardziej przypomina poprzednie wersje HTML-a, w których każdy element był stylizowany przy użyciu specjalnych znaczników HTML, takich jak `<center>`, `<font>` czy `<u>`. Ta metoda ma najmniejszy zasięg, ponieważ zdefiniowany za jej pomocą arkusz stylów odnosi się tylko do jednego elementu (rysunek 3.4).

Wady używania śródliniowych arkuszy stylów można wyczytać z samej nazwy tej technologii: **kaskadowe** arkusze stylów. Jako że kod CSS osadzony w elemencie HTML dotyczy tylko tego elementu, ta metoda mija się z podstawowym zastosowaniem CSS w ogóle, którym jest stosowanie do dokumentu (lub wielu stron) uniwersalnych reguł stylistycznych, aby obejmowały kaskadowo wszystkie elementy, które nie wymagają indywidualnej stylizacji.

Rysunek 3.4. Śródliniowy arkusz stylów odnosi się tylko do elementu, w którym został zdefiniowany

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:darkgrey;font-size:50px;text-align:center;">To jest nagłówek
zawierający śródliniowy kod CSS</h1>
<p style="color:black;font-size:20px;text-align:center;">To jest akapit
zawierający śródliniowy kod CSS.</p>

    <h2>To jest nagłówek bez śródliniowego kodu CSS</h2>
    <p>To jest akapit bez śródliniowego kodu CSS.</h2>

</body>
</html>
```

Wewnętrzny CSS

Wewnętrzny arkusz stylów odnosi się do wybranych grup elementów w obrębie jednej strony HTML. Zazwyczaj wstawia się go na stronę między znacznikami `<style>` i `</style>` (które powinno się umieścić między znacznikami `<head>` i `</head>`) (rysunki 3.5 i 3.6). Tę metodę stosuje się najczęściej wtedy, gdy jest tylko jedna strona do sformatowania. W witrynach składających się z wielu stron jego zastosowanie staje się nieefektywne.

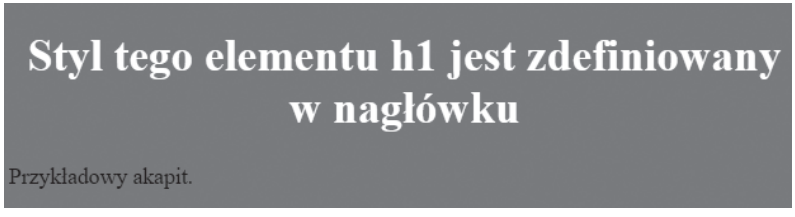
Rysunek 3.5. Wewnętrzny arkusz stylów w elemencie <style> na stronie internetowej

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: gray;
}

h1 {
  color: white;
  text-align: center;
}
</style>
</head>
<body>

<h1>Styl elementu h1 jest zdefiniowany w elemencie style w nagłówku</h1>
<p>To jest akapit.</p>

</body>
</html>
```

Listing 3.1. Plik 03-01.html

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body { background-color: gray; color: black; }
      h1 { color: white; text-align: center; }
    </style>
  </head>
  <body>
    <h1>Styl tego elementu h1 jest zdefiniowany w nagłówku</h1>
    <p>Przykładowy akapit.</p>
  </body>
</html>
```

Rysunek 3.6



W repozytorium pobranym pod adresem *https://ftp.helion.pl/przyklady/htcspo.zip* znajdują się pliki, których zawartość możesz łatwo kopiować i wklejać w inne miejsca.

Zewnętrzny CSS

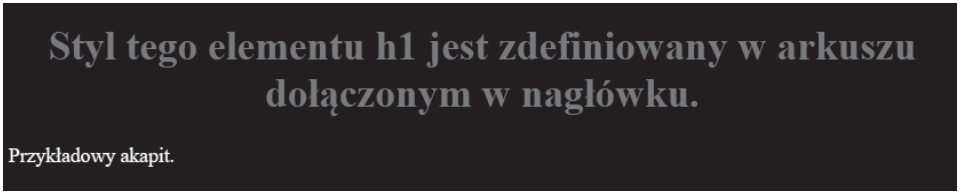
Reguły zawarte w zewnętrznym arkuszu stylów są stosowane do wielu stron naraz. Ta metoda dołączania CSS jest używana najczęściej w bardziej rozbudowanych serwisach internetowych. Zewnętrzny kod CSS wpisuje się w osobnym pliku, który dołącza się do wszystkich stron HTML, na których ma być używany. Odpowiedni element dołączający taki arkusz należy umieścić w części nagłówkowej dokumentu (rysunek 3.7).

Arkusz stylów CSS

```
body { background-color: black; color: white; }
h1 { color: gray; text-align: center; }
```

Strona HTML z łączem do arkusza stylów

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <h1>Styl tego elementu h1 jest zdefiniowany
     w arkuszu dołączonym w nagłówku.</h1>
    <p>Przykładowy akapit.</p>
  </body>
</html>
```



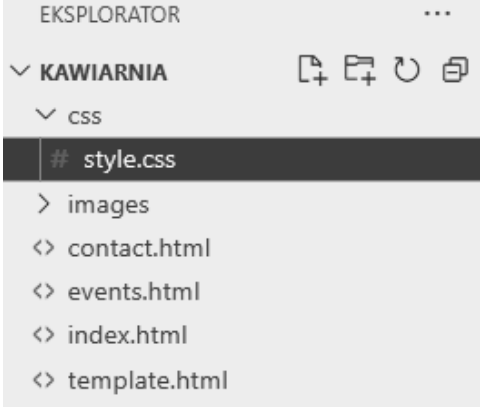
Rysunek 3.7. Zewnętrzny CSS został umieszczony w osobnym pliku, który następnie został dołączony do dokumentu HTML

CSS: (wpisz ten kod w pliku o nazwie *style.css* zapisanym w folderze *CSS* — rysunek 3.8):

Listing 3.2. Plik 03-02.css

```
body { background-color: black; color: white; }
h1 { color: gray; text-align: center; }
```

Rysunek 3.8. Plik style.css w folderze CSS w okienku Eksploratora programu Visual Studio Code



HTML: (wpisz ten kod w pliku o nazwie *index.html* i zwróć uwagę na ścieżkę *css/style.css* w elemencie łącza, która prowadzi do pliku *style.css* znajdującego się w folderze *css*):

Listing 3.3. Plik 03-03.html

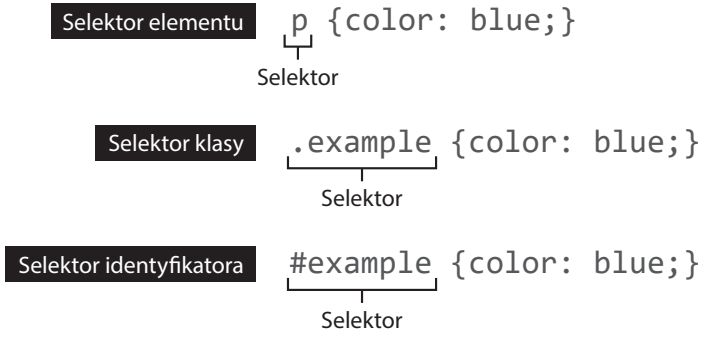
```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <h1>Styl tego elementu h1 jest zdefiniowany w arkuszu
dołączonym w nagłówku.</h1>
    <p>Przykładowy akapit.</p>
  </body>
</html>
```

Projektant nie musi ograniczać się do używania tylko jednej z tych trzech metod dołączania CSS do HTML-a. Jeśli jednak zdecydujesz się je mieszać, musisz pamiętać o hierarchii. Przeglądarki najpierw stosują zewnętrzne arkusze stylów w takiej kolejności, w jakiej zostały dołączone do strony. Następnie po kolei brane pod uwagę są style zdefiniowane wewnątrz elementu `<style>`. Na koniec przeglądarka stosuje style śródliniowe.

Stosowanie stylów do elementów

Style do elementów można stosować na trzy sposoby: przy użyciu selektorów elementów, selektorów klas oraz selektorów identyfikatorów. Przypomnij sobie, że reguła CSS musi zaczynać się od selektora określającego, co ma modyfikować (rysunek 3.9).

Rysunek 3.9



Selektory elementów

Selektory elementów identyfikują typ elementu (`body`, `p`, `a` itd.), wybierają wszystkie elementy określonego rodzaju i stosują do nich odpowiednie formatowanie.

Weźmy na przykład element `<body>`. Zawiera on całą widoczną w oknie przeglądarki treść strony internetowej. Za jego pośrednictwem można zdefiniować zestaw „domyślnych" stylów dla całego dokumentu (listing 3.4).

Listing 3.4. Plik 03-04.css

```
body { color: blue; }
```

W tym przypadku wybieramy element `body` na stronie HTML. Ta reguła odnosi się do wszystkiego, co znajduje się między znacznikami `<body>` i `</body>`.

Po wpisaniu selektora wstawiamy nawias klamrowy `{}`, w którym wpiszemy nasze deklaracje własności.

Każda deklaracja własności określa własność (w tym przypadku kolor). Po niej znajduje się dwukropek i wartość (w tym przypadku definiujemy kolor niebieski). Na końcu każdej deklaracji własności znajduje się średnik.

Efektem zastosowania tego prostego kodu CSS byłaby zmiana koloru tekstu na stronie na niebieski.

Użycie dwukropka jest wymagane. Jeśli go zabraknie, to zarówno własność, w której go nie ma, jak i następna własność nie zostaną wzięte pod uwagę przez większość przeglądarek.

Selektory klas

Czasami chcemy zastosować styl do pewnego podzbioru elementów. Możemy na przykład zmienić wygląd wybranego akapitu (lub zbioru akapitów) bez wpływania na inne. Aby to zrobić, musimy zdefiniować klasę akapitów. Elementy HTML można przypisywać do klas za pomocą specjalnego atrybutu.

Fragment `<p class="callout">` sygnalizuje początek akapitu, który będzie zdefiniowany przez klasę `callout`. Nazwa `callout` została wybrana przez użytkownika. Klasę można nazwać w dowolny sposób. Kod CSS będzie określał atrybuty stylistyczne tej klasy. Spójrz na kod HTML z listingu 3.5:

Listing 3.5. Plik 03-05.html

```
<p>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Aenean commodo ligula eget dolor. Aenean massa. Cum sociis
natoque penatibus et magnis dis parturient montes, nascetur
ridiculus mus. Donec quam felis, ultricies nec, pellentesque
eu, pretium quis, sem.
</p>

<p class="callout">
Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhon-
cus, sem quam semper libero, sit amet adipiscing sem neque
sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hen-
drerit id, lorem. Maecenas nec odio et ante tincidunt tempus.
</p>

<p>
Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget,
imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices
mauris. Integer ante arcu, accumsan a, consectetuer eget,
posuere ut, mauris. Praesent adipiscing. Phasellus ullamcor-
per ipsum rutrum nunc. Nunc nonummy metus.
<p>

<p class="callout">
Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhon-
cus, sem quam semper libero, sit amet adipiscing sem neque
sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hen-
drerit id, lorem. Maecenas nec odio et ante tincidunt tempus.
</p>
```

Pytanie: Lorem, co?

Odpowiedź: „Lorem ipsum…” to fikcyjny tekst od dawna używany jako tekst zastępczy w branży drukarskiej. Ma w miarę realistyczny wygląd, dzięki czemu nadaje się do celów demonstracyjnych. W Visual Studio Code są rozszerzenia umożliwiające generowanie tego tekstu.

Elementy akapitów przypisane do klasy `callout` odziedziczą jej style, natomiast akapity, które nie są do niej przypisane, zostaną wyrenderowane w domyślny sposób lub odziedziczą style elementu nadrzędnego.

Aby określić styl akapitów przypisanych do klasy `callout`, musimy użyć selektora klasy w pliku CSS. Selektor ten zaczyna się od kropki, po której następują nazwa klasy oraz lista atrybutów w klamrze (listing 3.6).

Listing 3.6. Plik 03-06.css

```
.callout {
     color: gray;
     font-style: italic;
     margin-left: 20px;
}
```

Jak widać, definicja każdej własności ma następującą postać: *`[typ atrybutu]`*`:` *`[szczegóły atrybutu];`*. Całość zamyka znak zamknięcia klamry.

Zastosowanie powyższej reguły CSS na naszej stronie dałoby efekt pokazany na rysunku 3.10.

Rysunek 3.10

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.

Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus.

Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetuer eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Nunc nonummy metus.

Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus.

W CSS obowiązuje ścisła hierarchia. Na przykład selektory klas mają wyższą rangę niż selektory elementów. Jeśli zastosujesz selektor `p` do akapitu, do którego odnosi się także selektor klasy, to zostaną zastosowane deklaracje przypisane klasie, a deklaracje selektora elementu zostaną zignorowane.

Selektory identyfikatorów

Klasy można przypisywać do wielu różnych elementów, natomiast selektor identyfikatora odnosi się tylko do jednego elementu na stronie. Jest on podobny do klasy pod względem sposobu definiowania, ponieważ również wymaga dodania specjalnego atrybutu do kodu HTML (listing 3.7).

Listing 3.7. Plik 03-07.html

```
<h1 id="pageTitle">...</h1>
```

Reguła zawierająca selektor identyfikatora powinna zaczynać się od znaku `#`, po którym należy wpisać identyfikator wybranego elementu (listing 3.8).

Listing 3.8. Plik 03-08.css

```
#pageTitle {font-weight: bold;}
```

Reguły zawierające selektor identyfikatora mają wyższą rangę niż selektory klas, które również mogą być przypisane do tych samych elementów, i domyślne style przeglądarki.

Porównanie klas i identyfikatorów

Klasy są przeznaczone do formatowania wielu elementów, natomiast identyfikatory odnoszą się tylko do jednego elementu na stronie. Klasy i identyfikatory są rodzajem atrybutów elementów HTML, które same nie pełnią bezpośrednio żadnej specjalnej funkcji. Aby dokładniej to zilustrować, przeanalizujemy typowe przypadki użycia każdego z nich.

Powiedzmy, że mamy kilka akapitów na stronie sklepu dla surferów. Już je oznaczyliśmy za pomocą elementów `<p>` w naszym dokumencie HTML. Jednak na naszej stronie występują dwa rodzaje akapitu. Jeden służy do przedstawiania standardowych informacji o sklepie, a drugi jest bardziej efektowny, ponieważ prezentuje informacje o aktualnych wyprzedażach. Chcielibyśmy mieć możliwość zastosowania pewnych specyficznych reguł formatowania do wszystkich akapitów „wyprzedażowych”. Dlatego w naszym arkuszu stylów utworzymy regułę o nazwie `bold-red` (listing 3.9).

Listing 3.9. Plik 03-09.css

```
.bold-red { font-weight: bold; color: red; }
```

Teraz w dokumencie HTML możemy wywołać tę klasę przez odpowiednie zmodyfikowanie znacznika otwierającego (listing 3.10).

Listing 3.10. Plik 03-10.html

```
<p class="bold-red">
W tym tygodniu oferujemy 40% rabatu na wszystkie deski
i kombinezony. Nie przegap okazji!
</p>
```

Klasę `bold-red` możemy przypisać każdemu elementowi. W tym konkretnym przypadku zostanie ona zastosowana tylko do jednego akapitu.

Gdyby to był jeden jedyny akapit wymagający pogrubienia i zmiany koloru tekstu na czerwony na całej stronie (albo w całej witrynie), to zamiast klasy moglibyśmy użyć identyfikatora. Identyfikatory odnoszą się do jednego elementu, podczas gdy klasy można przypisywać dowolnej liczbie elementów. Generalnie reguły CSS mogą odnosić się do kilku typów elementów, więc zachowanie ich elastyczności ułatwia pracę przy bardziej skomplikowanych stronach. Klasę `bold-red` moglibyśmy z łatwością zastosować także do elementu `h1` lub dowolnego innego elementu na stronie, który chcemy wyróżnić. O wiele łatwiej jest odnieść się do wybranej klasy, zamiast ponownie kodować wszystkie deskryptory i atrybuty.

Może zastanawiasz się, po co w ogóle używać identyfikatorów, skoro klasy są tak elastyczne i potężne. To świetne pytanie. Na pierwszy rzut oka identyfikatory wydają się niepotrzebne. Ponieważ jednak umożliwiają odwoływanie się do konkretnych elementów, możemy za ich pomocą stosować reguły stylistyczne do tych elementów bez używania arkuszy śródliniowych.

Identyfikatorów często używa się do oznaczania pojedynczego wyjątku od dominującej klasy: przypomnij sobie nasze porównanie do domu. Standardowo wszystkie okna mają białe ramy, ale jeśli chcemy zmienić kolor okna w łazience na szary, to możemy mu przypisać identyfikator i za jego pomocą zmienić tę właściwość. W ten sposób zmodyfikujemy tylko to jedno okno.

Identyfikatory umożliwiają także odnoszenie się do konkretnych elementów na stronie za pomocą JavaScriptu.

Style śródliniowe

Style śródliniowe są używane rzadziej niż omówione powyżej selektory. Są mniej poręczne w zastosowaniu, ale przydają się do wprowadzania drobnych modyfikacji. Dodaje się je w znaczniku otwierającym elementu HTML (listing 3.11). Style śródliniowe mają wyższą rangę niż selektory identyfikatorów, klas i elementów, jak również style domyślne przeglądarki.

Listing 3.11. 03-11.html

```
<h2 style="color: blue;">Matterhorn</h2>
```

Używana jest standardowa składnia CSS (*typ atrybutu*: *szczegóły atrybutu*;), dokładnie tak samo jak w przypadku reguł globalnych, mimo że style śródliniowe z definicji nie są globalne i muszą być zdefiniowane w każdym elemencie, który chcemy sformatować, osobno.

Stosowanie śródliniowych arkuszy stylów powinno być ograniczone do pojedynczych wierszy tekstu.

Przy użyciu szablonu *starter.html* z FTP poćwicz dodawanie różnych stylów, takich jak pogrubienie, kursywa i kolor, do próbek tekstu. Aby utrwalić sobie wiadomości na temat wszystkich metod dodawania arkuszy stylów, zastosuj style śródliniowe bezpośrednio do elementu akapitu, za pomocą klasy oraz przy użyciu identyfikatora. Eksperymentuj do woli z różnymi kolorami, które możesz definiować za pomocą nazw. Edytor Visual Studio Code pokazuje podgląd koloru podczas wpisywania jego nazwy, jak widać na rysunku 3.11.


```
p {
    color: ■black;
}
```

Rysunek 3.11. Podgląd koloru w Visual Studio Code

W Visual Studio Code istnieje wygodny skrót ułatwiający tworzenie początkowego szablonu strony. Wystarczy napisać *!* i nacisnąć klawisz *Enter*, aby edytor wstawił odpowiedni kod.

Silniki renderujące

Kiedy strona jest ładowana, jej kod jest przetwarzany od góry do dołu. Przeglądarka analizuje plik HTML, tworząc po kolei wszystkie elementy i formatując je zgodnie z regułami CSS. Część przeglądarki odpowiedzialna za te czynności nazywa się **silnikiem renderowania**. Obecnie istnieje kilka takich algorytmów, które są wykorzystywane przez nowoczesne przeglądarki. Safari firmy Apple używa algorytmu o nazwie WebKit. Chrome i Microsoft Edge używają wariantu WebKit o nazwie Blink, a Firefox wykorzystuje silnik o nazwie Gecko (rysunek 3.12). Istnieją jeszcze inne, ale WebKit, Blink i Gecko są używane najczęściej.

Rysunek 3.12

Przeglądarka	Silnik renderowania
Lunascape	Gecko, Trident, WebKit
Internet Explorer	Trident
Firefox	Gecko
Google Chrome	WebKit (Blink)
Safari	WebKit
Opera v.15+	WebKit (Blink)

Jako początkujący projektant może zastanawiasz się, po co omawiamy takie techniczne szczegóły jak silniki renderowania wykorzystywane przez różne przeglądarki. Choć HTML i CSS zostały już w dużej mierze ustandaryzowane, wciąż występują pewne różnice w sposobie obsługi kodu przez przeglądarki. Największe znaczenie ma to w przypadku wyboru składników CSS. W wielu przypadkach coś, co wygląda fantastycznie w przeglądarce Chrome, może wyglądać dziwnie (albo w ogóle się nie wyświetlać) w Internet Explorerze lub Edge.

Identyfikacja i uwzględnienie tych niespójności należy do procedur testowania projektu w różnych przeglądarkach przed uznaniem go za ukończony. Prawie zawsze istnieje sposób na ominięcie różnic między przeglądarkami, ale często wymaga to napisania wielu dodatkowych reguł CSS.

Kiedy do przeglądarek są dodawane nowe funkcje, mogą występować różnice w ich interpretacji. Z czasem jednak wszystko się stabilizuje i ujednolica.

Skąd wiedzieć, dla których przeglądarek pisać kod?

Wiele osób doradziłoby, aby sprawdzić statystyki swojej witryny i dowiedzieć się, z jakich przeglądarek korzysta większość użytkowników. Zazwyczaj to dobry pomysł. Ale co zrobić, jeśli witryna jeszcze nie jest opublikowana w internecie? Zasadniczo najlepiej jest uwzględnić trzy lub cztery najważniejsze przeglądarki: Chrome, Internet Explorer, Edge i Firefox.

Wiele narzędzi analitycznych, takich jak Google Analytics, jest dostępnych za darmo. Google Analytics to bardzo popularna usługa, która dostarcza wielu informacji na temat ruchu w witrynie, w tym także o przeglądarkach używanych przez odwiedzających. Inne świetne usługi tego typu to Matomo (wcześniej zwana Piwik) i AWStats.

Skąd mam wiedzieć, jaki rozmiar powinny mieć elementy na moich stronach?

W dawnych czasach strony projektowało się tak, aby były jak najszersze. Wynikało to z tego, że projektanci chcieli lub musieli wykorzystać każdy piksel. Nowocześniejsze i bardziej dojrzałe podejście do kwestii szerokości strony głosi, że powinna ona odpowiadać rodzajowi treści i reagować na działania użytkownika.

Określając wymiary w CSS (na przykład marginesy i dopełnienie), dobrze jest stosować wartości procentowe, które umożliwiają przeglądarkom ich skalowanie proporcjonalnie do rozmiaru ekranu. Na przykład:

Listing 3.12. Plik 03-12.css

```
p { width: 200px; }
```

Ta reguła spowoduje, że akapity będą miały szerokość 200 pikseli. Natomiast poniższa reguła:

Listing 3.13. Plik 03-13.css

```
p { width: 80%; }
```

nakaże przeglądarce nadać akapitom szerokość równą 80% szerokości zawierającego je elementu, co pozwala na lepsze dostosowanie skali do różnych urządzeń.

W usłudze analitycznej, z której korzystasz, możesz sprawdzić, jakiej rozdzielczości ekranów najczęściej używają Twoi użytkownicy. Bardzo ważne jest, aby pamiętać, że

Twoja strona będzie oglądana także na telefonach i innych urządzeniach mobilnych. Z tego względu dobrze jest rozważyć możliwość zastosowania projektu responsywnego, który będzie zmieniał układ w zależności od wymiarów używanego urządzenia. Techniki projektowania responsywnego zostały omówione w rozdziałach 13. i 14.

Skąd wiedzieć, które funkcje są obsługiwane?

Przeglądarki nieustannie są ulepszane przez programistów, aby obsługiwały najnowsze i najlepsze funkcje. W takim dynamicznym środowisku dobrze jest mieć możliwość sprawdzenia, które funkcje są obsługiwane przez poszczególne silniki renderujące. Dobrym miejscem do tego jest serwis Can I Use dostępny pod adresem *www.caniuse.com*. W witrynie tej znajdziesz listę wersji przeglądarek obsługujących poszczególne funkcje oraz objaśnienia subtelnych różnic występujących między silnikami renderującymi.

Poza siecią

CSS, podobnie jak HTML, znajduje zastosowania także poza stronami internetowymi. Arkusze stylów w połączeniu z innymi językami są używane do formatowania książek elektronicznych, map i innych materiałów. Standard CSS jest dość dynamiczny i ciągle się rozwija, dzięki czemu regularnie pojawiają się w nim nowości. Tak jak HTML, jest to „żywy standard", który jest stopniowo dostosowywany na podstawie sposobu używania przez projektantów i uwag społeczności. CSS zapewne będziemy używać, dopóki nie wydarzy się jakaś dramatyczna zmiana sposobu prezentacji danych online.

Podsumowanie

- CSS można stosować do elementów przez śródliniowe, wewnętrzne lub zewnętrzne arkusze stylów.
- Reguły CSS mogą odnosić się do wszystkich elementów, wybranego typu elementów, zdefiniowanych przez użytkownika klas elementów oraz konkretnych elementów oznaczonych atrybutem `id`.
- Nie zaleca się używania śródliniowych arkuszy stylów w innych sytuacjach niż prace testowe.
- CSS, tak jak HTML, jest żywym standardem.

HTML i CSS. Odkryj przepis na piękne strony internetowe!

HTML i CSS służą do tworzenia stron internetowych, interfejsów telewizorów, gier konsolowych, a także urządzeń AGD. Opanowanie umiejętności pisania kodu HTML i CSS, oprócz tego, że pozwala tworzyć aplikacje internetowe, stanowi świetne wprowadzenie do nauki programowania. Znajomość tych języków przyda się zresztą każdemu, kto w pracy korzysta z Internetu i różnych technologii. By rozpocząć przygodę z HTML i CSS, wystarczy prosty edytor tekstowy, przeglądarka internetowa — i ten podręcznik.

Ta książka to doskonały wybór na początek dla osób, które chcą się nauczyć pisać kod. W łatwy i przyjemny sposób nauczy Cię podstaw HTML i CSS. Nie znajdziesz w niej nużącej i trudnej teorii — poszczególne zagadnienia zostały przystępnie i angażująco wyjaśnione, między innymi dzięki ilustracjom i przykładom, co ułatwia zrozumienie praktycznego znaczenia pojęć. Dowiesz się, w jaki sposób rozpocząć projekt i na co zwrócić uwagę podczas pracy. Niektóre ćwiczenia zawarte w tym podręczniku składają się na rzeczywisty, wieloetapowy proces tworzenia witryny internetowej. Szybko się przekonasz, że zbudowanie w pełni funkcjonalnej strony internetowej wcale nie jest trudne i może dać mnóstwo radości i satysfakcji!

Dzięki książce:

- poznasz podstawy projektowania stron internetowych
- dowiesz się, jak współdziałają HTML5 i CSS3
- zrozumiesz strukturę witryny i zasady projektowania responsywnego
- nauczysz się dodawać do projektów formularze, multimedia i animacje
- wzbogacisz swój warsztat o formatowanie, gradienty, testowanie, debugowanie i wiele innych technik

David DuRocher

od wielu lat uczy projektowania stron internetowych. Jest znany ze stosowania metod zorientowanych na wyniki i angażującego stylu nauczania, co pozwala mu na osiąganie znakomitych efektów. Mieszka z żoną w Stanach Zjednoczonych, w północnowschodnim Connecticut.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-8322-655-2

9 788383 226552

Cena: 69,00 zł