REFORMA 2017

Część 2

EE.09

Programowanie i tworzenie stron internetowych oraz baz danych i administrowanie nimi

Podręcznik do nauki zawodu

• TECHNIK INFORMATYK

WSiP

REFORMA 2017

Część 2

EE.09

Programowanie i tworzenie stron internetowych oraz baz danych i administrowanie nimi

Agnieszka Klekot
Tomasz Klekot

Podręcznik do nauki zawodu

• TECHNIK INFORMATYK

WSiP

Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych do kształcenia w zawodach na podstawie opinii rzeczoznawców: **dr Moniki Szymańskiej**, **mgr inż. Janiny Grobelnej** i **mgr Zofii Gondy-Ciupy**.

Rok dopuszczenia: 2018

Typ szkoły: **technikum, szkoła policealna.**
Zawód: **technik informatyk.**
Kwalifikacja: **EE.09. Programowanie, tworzenie i administrowanie stronami internetowymi i bazami danych.**

Wydanie I

ISBN 978-83-02-17917-4 (całość)
ISBN 978-83-02-17362-2 (część 2)

Opracowanie merytoryczne i redakcyjne: **Zbigniew Dziedzic** (redaktor koordynator)
Konsultacja: **dr inż. Krzysztof Pytel**
Redakcja językowa: **Ewa Wojtyra**, **Hanna Kus**
Redakcja techniczna: **Elżbieta Walczak**
Projekt okładki: **Dominik Krajewski**
Fotoedycja: **Natalia Marszałek**
Skład i łamanie: **Ledor**

Wydawnictwa Szkolne i Pedagogiczne Spółka Akcyjna
00-807 Warszawa, Aleje Jerozolimskie 96
KRS: 0000595068
Tel.: 22 576 25 00
Infolinia: 801 220 555
www.wsip.pl
Druk i oprawa: ArtDruk Zakład Poligraficzny Andrzej Łuniewski

1. Technologie webowe

2. Tworzenie stron internetowych w języku HTML

3. Kaskadowe arkusze stylów CSS

4. Skrypty po stronie klienta – JavaScript

5. Grafika na stronach internetowych

6. Multimedia na stronach internetowych

7. Edytory spełniające funkcje WYSIWYG

8. Projektowanie stron internetowych

9. Testowanie, publikacja oraz optymalizacja stron internetowych

10. Systemy zarządzania treścią CMS

1. Technologie webowe

- Podstawy komunikacji w internecie
- Technologie webowe

1.1 Podstawy komunikacji w internecie

ZAGADNIENIA

- **Jak komunikujemy się w sieci internetowej?**
- **Protokoły wykorzystywane przez aplikacje internetowe**
- **Jak działa protokół HTTP?**

Do podstawowych elementów każdego systemu informacyjnego należą: nadawca, odbiorca, kanał komunikacyjny i oczywiście informacja. W systemach informatycznych, w których informacja w formie danych jest przetwarzana za pomocą systemów komputerowych, głównym kanałem komunikacyjnym jest sieć komputerowa wykorzystująca różne media transmisyjne – przewodowe i bezprzewodowe. Obecnie trudno jest wyobrazić sobie działanie jakiegokolwiek systemu informatycznego bez komunikacji z serwerami przez rozległe sieci komputerowe (MAN, WAN) – internet czy VPN. Zjawisko jest tak powszechne, że mówimy już o strukturze zwanej internetem wszechrzeczy (ang. *Internet of Everything* – IoE), definiującej sieć ludzi, procesów, danych i rzeczy podłączonych do sieci internetowej.

Wszystko to wymagało wstępnego utworzenia zasad, schematów i standardów komunikacji sieciowej oraz przesyłania i prezentacji danych w sieci.

W pierwszych latach nauki w technikum omawiano model komunikacji sieciowej TCP/IP. Jest on modelem odniesienia dla użytkowników i twórców sieciowych rozwiązań komunikacyjnych, który zapewnia kompatybilność komunikacyjną różnych protokołów odpowiedzialnych za przesyłanie i prezentację danych w sieci. Dzięki temu na przykład stronę internetową możemy otwierać w różnych systemach operacyjnych i na dowolnym urządzeniu (laptop, pecet czy smartfon). Nas najbardziej interesuje warstwa aplikacji, ponieważ obsługuje ona aplikacje internetowe, zapewnia im usługi wymiany i dostępu do danych.

TCP/IP Layers	TCP/IP Protocols				
Warstwa aplikacji	HTTP	FTP	Telnet	SMTP	DNS
Warstwa transportowa	TCP		UDP		
Warstwa internetowa	IP		ARP	ICMP	IGMP
Warstwa dostępu do sieci	Ethernet		Token Ring	Inne protokoły dostępu do sieci	

Rys. 1.1. Zestaw protokołów warstwy aplikacji TCP/IP

W pracy nad witrynami i aplikacjami internetowymi wykorzystujemy następujące protokoły komunikacji sieciowej:

HTTP (ang. *HyperText Transfer Protocol*) – podstawowy protokół wykorzystywany do transportu i prezentacji danych w postaci strony internetowej. Obecnie protokół pozwala na przesyłanie stron w postaci plików, jak również może obsługiwać np. strumieniową transmisję danych. Aplikacją obsługującą ten protokół jest każda przeglądarka internetowa. Kluczowym dla pobierania treści i ich prezentacji jest adres **URL** (ang. *Uniform Resource Locator*) definiujący lokalizację pliku i usługę protokołu.

PRZYKŁAD 1.1.

https://sportowefakty.wp.pl/pilka-nozna.html

HTTPS (ang. *HyperText Transfer Protocol Secure*) – protokół umożliwiający przesyłanie szyfrowanych plików stron internetowych. Zwiększa to znacznie bezpieczeństwo pracy, np. na formularzach wykorzystywanych w aplikacjach internetowych. Do szyfrowania używa się w tym przypadku protokołu **SSL** (ang. *Secure Socket Layer*).

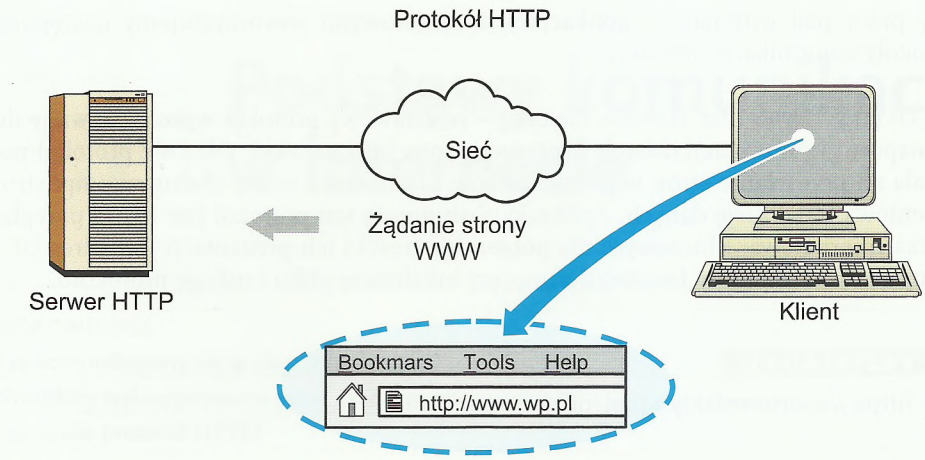
Definicja **FTP** (ang. *File Transfer Protocol*) – protokół umożliwiający przesyłanie plików. W aplikacjach internetowych pozwala na pobieranie i wgrywanie plików między użytkownikiem aplikacji a serwerem oraz na publikowanie na serwerze plików stron internetowych przez administratora. Dostępny jest również protokół oferujący szyfrowanie danych **SFTP** (ang. *Secure File Transfer Protocol*).

Definicja **DNS** (ang. *Domain Name System*) – protokół pozwalający na nadawanie użytkownikom internetu alfanumerycznych nazw adresów logicznych. Protokół tłumaczy je na odpowiadający im adres IP. Dzięki temu użytkownik łatwo zapamięta adres naszej strony i nie musi pamiętać zapisu binarnego adresu IP, pod którym strona się znajduje.

Wszystkie opisane protokoły zaliczamy do protokołów publikowanych w standardzie otwartym. Oznacza to, że ich specyfikacja nie jest ograniczona prawnie i każdy może na ich podstawie tworzyć własne rozwiązania sieciowe.

PRZYKŁAD 1.2.

Jak działa protokół HTTP?
Użytkownik wpisuje adres URL strony internetowej i tym samym wysyła do sieci żądanie pobrania i wyświetlenia danego zasobu. Jeżeli zostaną odnalezione odpowiedni serwer HTTP oraz pliki strony, zostaną one przesłane na komputer użytkownika. Proces ten potocznie nazywamy ładowaniem strony. Jeżeli po domyślnym czasie nie zostanie znaleziony określony w żądaniu plik, przeglądarka wyświetli komunikat o błędzie numer 404 (brak pliku lub zła lokalizacja pliku).



Rys. 1.2. Działanie protokołu HTTP

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień nazwy protokołów występujących w warstwie aplikacji.
2. Co to jest adres URL?
3. Czym różni się adres URL od bezwzględnej ścieżki dostępu do zasobu w systemie Windows 7?

1.2 Technologie webowe

ZAGADNIENIA
- **Rodzaje technologii webowych**
- **Historia języka HTML**
- **Podstawowe pojęcia dotyczące stron WWW**

Rodzaje technologii webowych

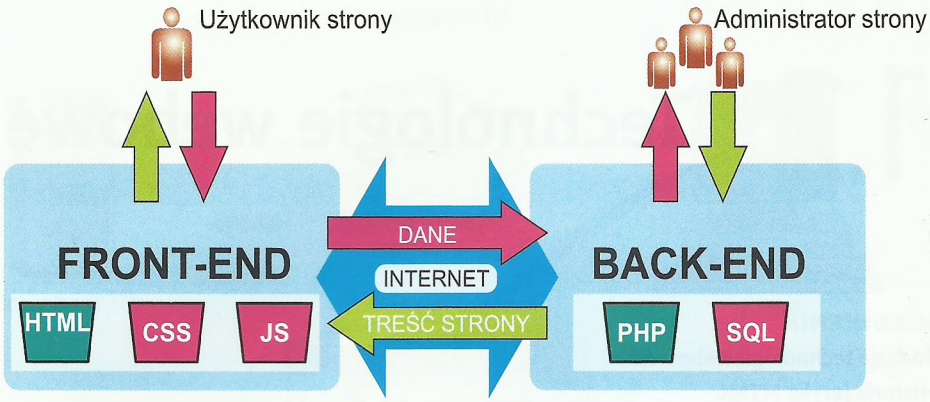
Jedną z najpopularniejszych usług sieciowych jest tworzenie i publikowanie w sieci dokumentów hipertekstowych, zwanych powszechnie stronami **WWW** (ang. *World Wide Web*). Do tworzenia stron WWW używa się różnych technologii określanych jako webowe. Technologie webowe są to wszystkie języki programowania i techniki multimedialne oraz sieciowe, pozwalające na jak najlepsze, kreatywne i uniwersalne tworzenie stron i aplikacji internetowych – interpretowanych zarówno na urządzeniach stacjonarnych, jak i mobilnych.

Początkowo strony internetowe zawierały głównie tekst, hiperłącza oraz grafikę statyczną i były tworzone wyłącznie z wykorzystaniem języka HTML.

HTML (ang. *HyperText Markup Language*) – hipertekstowy język znaczników wykorzystywany do tworzenia dokumentów hipertekstowych, które są interpretowane przez przeglądarkę internetową.

Z czasem jednak, kiedy przepustowość sieci rosła, a przeglądarki pozwalały na więcej, zaczęto tworzyć witryny internetowe korzystające z różnych technologii. Obecnie technologie webowe klasyfikujemy według obszarów działania. Wyróżniamy:
- FRONT-END – technologie wykorzystujące kod jawny, wykonywany po stronie przeglądarki klienta/użytkownika;
- BACK-END – technologie wykorzystujące kod ukryty (tajny), wykonywany po stronie serwera.

Do głównych technologii front-end zaliczamy HTML, CSS, JavaScript czy grafikę wektorową uruchamianą po stronie klienta **SVG** (ang. *Scalable Vector Graphics*). Natomiast back-end opiera się głównie na frameworkach .NET, Java czy Python oraz skryptach PHP współpracujących z sieciowymi bazami danych SQL.



Rys. 1.3. Podział technologii webowych

Wybór technologii strony, którą zamierzamy stworzyć, jest bardzo ważny. Określa możliwości zastosowania poszczególnych obiektów, interaktywności z użytkownikiem, szybkości działania oraz funkcjonalności. Może również w zasadniczy sposób wpłynąć na koszt całego projektu. Wybór technologii powinien nastąpić na początku prac nad projektem strony WWW i stanowić wspólne ustalenie zleceniobiorcy i zleceniodawcy.

Historia i rozwój języka HTML

Od początków lat 80. XX wieku trwały prace nad stworzeniem standardu przesyłania dokumentów w sieci internet. Jednak zniesienie zakazu wykorzystywania zasobów sieci internet do celów komercyjnych w roku 1991 przyspieszyło rozwój tej usługi. W tym samym roku Tim Bernes-Lee zamieścił w sieci dokument *HTML Tags*, w którym opisał 24 znaczniki do zapisu dokumentów udostępnianych w internecie. Każdy z nich zawierał słowo, które było otoczone nawiasami ostrymi. Dokument ten powstał na podstawie standardu **SGML** (ang. *Standard Generalized Markup Language*) – metajęzyka umożliwiającego definiowanie dowolnych znaczników i zasad ich użycia. Można uznać, że był to standard **HTML1**, chociaż nie pojawiła się oficjalna specyfikacja tego języka.

W 1993 roku Mark Andreessen udostępnił pierwszą graficzną przeglądarkę stron internetowych – **Mosaic**. Pierwsza oficjalna specyfikacja języka **HTML2.0** została opublikowana w 1995 roku przez **IETF** (ang. *Internet Engineering Task Force*). Niedługo potem powołano do czuwania nad rozwojem standardów pisania stron internetowych organizację **W3C** (ang. *World Wide Web Consortium*).

Po opublikowaniu specyfikacji **HTML3.2** w roku 1997 przystąpiono do prac nad specyfikacją języka HTML, związanego z kaskadowymi arkuszami stylów **CSS**, które przejmowały główny ciężar formatowania stron internetowych. Tym samym oddzielono strukturę logiczną tekstu od struktury prezentacji. W roku 1999 ukazała się specyfikacja **HTML4**. Jednak dopiero w 2001 roku udało się zakończyć pracę nad stabilną, interpretowaną przez wszystkie przeglądarki specyfikacją **HTML4.1**.

Następnym krokiem w rozwoju języka HTML było dostosowanie składni do standardów wprowadzonych przez metajęzyk **XML** (ang. *Extensible Markup Language*). XML wykorzystujemy do budowania strukturyzowanych form prezentacji danych w przeglądarkach internetowych i aplikacjach bazodanowych. W ten sposób powstała specyfikacja języka **XHTML1.0** (ang. *Extensible HyperText Markup Language*), która jest prostą ewaluacją **HTML4.1** pozwalającą współpracować z dokumentami XML.

Dynamiczny rozwój przeglądarek internetowych spowodował rozłam w planowaniu przyszłości języków hipertekstowych. Oprócz W3C powstała druga organizacja zajmująca się rozwojem stron WWW – **WHATWG** (ang. *Web Hypertext Application Technology Working Group*). Tworzyli ją głównie przedstawiciele Apple, Mozilli i Opery. Głównym celem tej organizacji był rozwój technologii umożliwiających tworzenie na podstawie kodu HTML bardziej zaawansowanych aplikacji sieciowych.

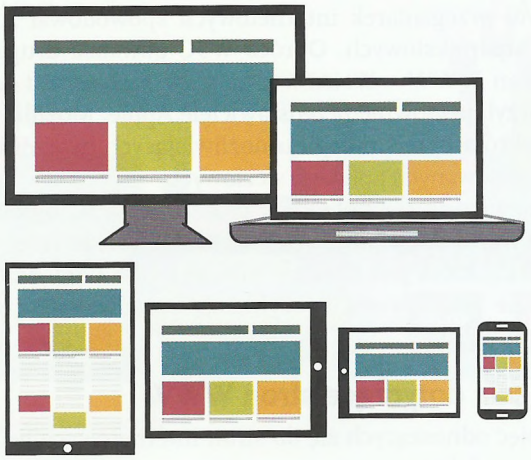
Po kilku latach rozwijania równoległych projektów W3C ogłosiła chęć współpracy z organizacją WHATWG w rozwijaniu języka HTML. Efektem tej współpracy jest specyfikacja języka **HTML5**, która jest obecnie zalecana jako narzędzie do tworzenia stron internetowych i będzie podstawową specyfikacją przedstawianą w tym podręczniku. Opublikowano ją w roku 2014.

Podstawowe pojęcia dotyczące stron WWW

Spośród mnóstwa pojęć odnoszących się do stron internetowych warto wybrać i objaśnić te, które przydadzą się w dalszej nauce.

- **Statyczna strona internetowa** to strona z określonymi na stałe parametrami formatowania jej zawartości. Użytkownik nie ma wpływu na prezentację treści.
- **Dynamiczna strona internetowa** to strona generowana dynamicznie po stronie serwera na podstawie parametrów przekazywanych przez przeglądarkę internetową. Użytkownik ma wpływ na prezentację i zawartość strony.
- **Witryna internetowa** to grupa stron internetowych tworząca określoną strukturę oraz jednolita tematycznie.
- **Portal internetowy** to serwis internetowy zawierający informacje na różne tematy i z różnych dziedzin.
- **Wortal internetowy** to wyspecjalizowany serwis internetowy.
- **System zarządzania treścią CMS** (ang. *Content Management System*) to system tworzenia witryn internetowych i zarządzania witryną, sklepem lub blogiem, oparty na infrastrukturze klient–serwer z wykorzystaniem bazy danych, języka HTML i skryptów PHP.
- **Serwer WWW** to usługa (rola) oferowana przez komputery-serwery z sieciowym systemem operacyjnym. Najczęściej takie usługi są oferowane przez firmy hostingowe, które dysponują farmami serwerów.
- **Blog** (ang. *web log*) to dziennik internetowy, przeważnie o określonej tematyce. Blog może przybierać różne formy, np. fotobloga, wideobloga lub audiobloga.
- **Serwer bazy danych** to usługa oferująca zasoby do zarządzania sieciowymi bazami danych. W technologii webowej istotne jest realizowanie przez serwer zapytań SQL. Przykładowymi serwerami wykorzystywanymi w tworzeniu stron i aplikacji internetowych są MySQL i MSSQL.

Obecnie rozwój urządzeń mobilnych narzuca projektowanie i tworzenie stron WWW obsługiwanych w wielu przeglądarkach i wielu rozdzielczościach. Dlatego też popularne stało się tworzenie **stron responsywnych**, czyli takich, których wygląd i układ dostosowują się automatycznie do rozmiaru okna przeglądarki, na której jest wyświetlany.



Rys. 1.4. Zasada działania stron responsywnych

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wejdź na stronę organizacji W3C (http://www.w3.org) i odszukaj specyfikacje dotyczące języka HTML. Zapoznaj się ze strukturą opisu specyfikacji języka stosowanej przez W3C.
2. Wyjaśnij, dlaczego korzysta się z tworzenia stron responsywnych.
3. Jakie technologie webowe zastosowałbyś do tworzenia prostej strony informacyjnej zakładu zegarmistrzowskiego oraz sklepu internetowego? Swoje propozycje zapisz w tabeli.

Rodzaj strony	Proponowane technologie
strona informacyjna	
sklep internetowy	

2. Tworzenie stron internetowych w języku HTML

- Struktura dokumentu HTML
- Podstawowe znaczniki i atrybuty języka HTML
- Barwa na stronie internetowej
- Podział strony w języku HTML
- Grafika i multimedia w dokumentach HTML

2.1 Struktura dokumentu HTML

ZAGADNIENIA
- Składnia języka HTML
- Struktura dokumentu HTML
- Zasady edycji plików HTML

Dokument utworzony w języku HTML musi mieć określoną strukturę. Oznacza to, że kod dokumentu ma zawierać elementy niezbędne do jego poprawnej interpretacji przez przeglądarkę internetową. Dodatkowo cały dokument powinien być napisany zgodnie z regułami składni języka HTML.

Składnia języka HTML

Podstawowymi elementami w języku HTML są **znaczniki** (ang. *tag*). Znaczniki to polecenia interpretowane przez przeglądarkę internetową. Są one otoczone symbolami < oraz >. Znacznik może posiadać atrybut, a atrybut może mieć przypisaną wartość. Znaczniki mogą być pisane zarówno małymi, jak i wielkimi literami.

PRZYKŁAD 2.1.

```
<znacznik atrybut="wartość"> tekst </znacznik>
```

Znaczniki możemy podzielić na:
- **sparowane**, czyli tworzące parę znaczników, z których jeden otwiera element składni języka HTML, np. `<title>`, a drugi służy do jego zamknięcia, np. `</title>`;
- **niesparowane**, składające się tylko ze znacznika otwierającego, np. `<br>` lub `<hr>`.

Dodatkowo w składni języka HTML są stosowane **znaki specjalne**, np. ` ` – pozwala na wstawienie twardej spacji. Składnia języka HTML dopuszcza również stosowanie komentarzy. Budowa komentarza wygląda następująco:

```
<!--komentarz -->
```

Składnia języka HTML dopuszcza zagnieżdżanie znaczników. Jednak należy pamiętać o tym, że zamykamy je w kolejności odwrotnej do tej, w której je otwieraliśmy.

PRZYKŁAD 2.2.

```
<p><b><i><font color="green">przykład zagnieżdżenia znaczników
</font></i></b></p>
```

Struktura dokumentu HTML

W strukturze poprawnego dokumentu HTML należy wyróżnić następujące elementy:

- deklarację typu dokumentu – **DOCTYPE**;
- definicję dokumentu – **HTML**;
- nagłówek dokumentu – **HEAD**;
- treść dokumentu – **BODY**.

```
<!DOCTYPE html>
<!--deklaracja typu dokumentu HTML5-->
<html>
<!--znacznik początku dokumentu HTML-->
  <head>
<!--nagłówek zawierający parametry dotyczące wyświetlania strony
oraz dołączania skryptów i plików zewnętrznych-->
    <meta charset="utf-8">
<!--sposób kodowania polskich znaków-->
    <link rel="stylesheet" href="plik.css">
<!--sposób dołączania zewnętrznego arkusza stylów do formatowania strony-->
    <title> Tytuł strony </title>
<!--tytuł strony wyświetlany w zakładce przeglądarki-->
  </head>
<!--zamknięcie nagłówka strony-->
  <body>
<!--znacznik początku treści strony-->
Treść strony internetowej wyswietlany w oknie głównym przeglądarki
  </body>
<!--zamknięcie treści strony-->
</html>
<!--zamknięcie dokumentu HTML-->

```

Rys. 2.1. Struktura dokumentu HTML5

Deklaracja typu dokumentu jest definiowana przez określenie standardu **DTD** (ang. *Document Type Definition*), w jakim tworzymy naszą stronę internetową. Specyfikacja języka HTML4.01 zawiera trzy typy dokumentów: **Strict, Transitional** i **Frameset**.

PRZYKŁAD 2.3.

Wersja **Strict** definicji typu dokumentu (DTD) jest wersją wykluczającą wszelkie elementy prezentacyjne, które specyfikacja HTML4.01 uznaje za zdeprecjonowane, czyli przestarzałe. Występuje ona często z dodanym adresem sieciowym wersji DTD, pomagającym w pobraniu definicji przez przeglądarkę.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.
w3.org/TR/html4/strict.dtd">
```

Wersja **Transitional** (przejściowa lub luźna) – zawiera wszystko, co wersja Strict, plus elementy i atrybuty zdeprecjonowane oraz element **<iframe>**.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Wersja **Frameset** (obsługująca ramki) zawiera wszystko, co wersja Transitional, plus elementy odnoszące się do ramek: <frame>, <frameset> oraz <noframes>.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
```

W specyfikacji HTML5 zapis typu dokumentu jest znacznie uproszczony.

```
<!DOCTYPE html>
```

Natomiast w XHTML nie ma żadnego typu deklaracji.

Definicja dokumentu określa język, w jakim jest napisana strona. Najczęściej używanym znacznikiem jest **<html>**. Jednak możliwe jest zastosowanie dodatkowych parametrów do określenia typu języka lub ustawień regionalnych.

PRZYKŁAD 2.4.

```
<html lang="pl-PL">
```

Nagłówek dokumentu zawiera informacje o stronie, które nie są wyświetlane w oknie przeglądarki. Najczęściej stosowanymi znacznikami w nagłówku są:

- **<title>** – określający tytuł strony;
- **<style>** – definiujący wewnętrzne arkusze stylów;
- **<link>** – odsyłacz do innego dokumentu hipertekstowego;
- **<meta>** – zawierający szczegółowe informacje o stronie, np. kodowanie, słowa kluczowe, dane autora itp.

PRZYKŁAD 2.5.

Znacznik **<meta>** pozwala umieścić w nagłówku wiele istotnych informacji. Korzystamy w tym znaczniku z atrybutów: **http-equiv**, **name** oraz **content**. Poniżej przedstawiono przykłady wykorzystania tych atrybutów.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
<!--sposób kodowania polskich znaków-->
    <meta http-equiv="Content-Language" content="pl">
<!--sposób definiowania języka dla zawartości strony-->
    <meta http-equiv="reply-to" content="bb@pl.pl">
<!--sposób podania adresu e-mail autora strony-->
    <meta http-equiv="refresh" content="30">
<!--sposób zdefiniowania odświeżania strony np. co 30 sek.-->
    <meta http-equiv="creation-date" content="2017-09-14T12:15:38Z">
<!--sposób zdefiniowania daty i czasu utworzenia strony-->
    <meta http-equiv="default-style" content="preferowane stylesheet">
<!--sposób określenia preferowanych stylów dla strony-->
    <meta name="description" content="przykładowa strona WWW">
<!--sposób opisu treści strony-->
    <meta name="keywords" content="html, strona, informatyka">
<!--sposób określania słów kluczowych-->
    <meta name="author" content="Bartilini Babacki">
<!--sposób na podanie autora strony-->
    <meta name="generator" content="Notepad++">
<!--sposób na okreslenie edytora kodu-->
    <meta name="robots" content="index">
<!--sposób na indeksowanie strony-->
    <meta name="copyright" content="(c)2005 Jan Nowak">
<!--sposób na zdefiniowanie praw autorskich-->
  </head>
  <body>
Treść strony internetowej wyswietlany w oknie głównym przeglądarki
  </body>
</html>
```

Rys. 2.2. Przykład stosowania znacznika **<meta>**

Kodowanie polskich znaków. Podczas tworzenia dokumentu w języku HTML należy pamiętać, aby treść wyświetlała się w przeglądarce w odpowiedni sposób. Dlatego też istotnym etapem tworzenia strony jest deklaracja języka dla strony oraz określenie sposobu kodowania liter ze znakami diakrytycznymi (np. *ż*, *ą*, *ź* itp.). Wykonuje się to w znaczniku **<meta>** z atrybutem **charset**. Najczęściej używane kodowania polskich znaków to:

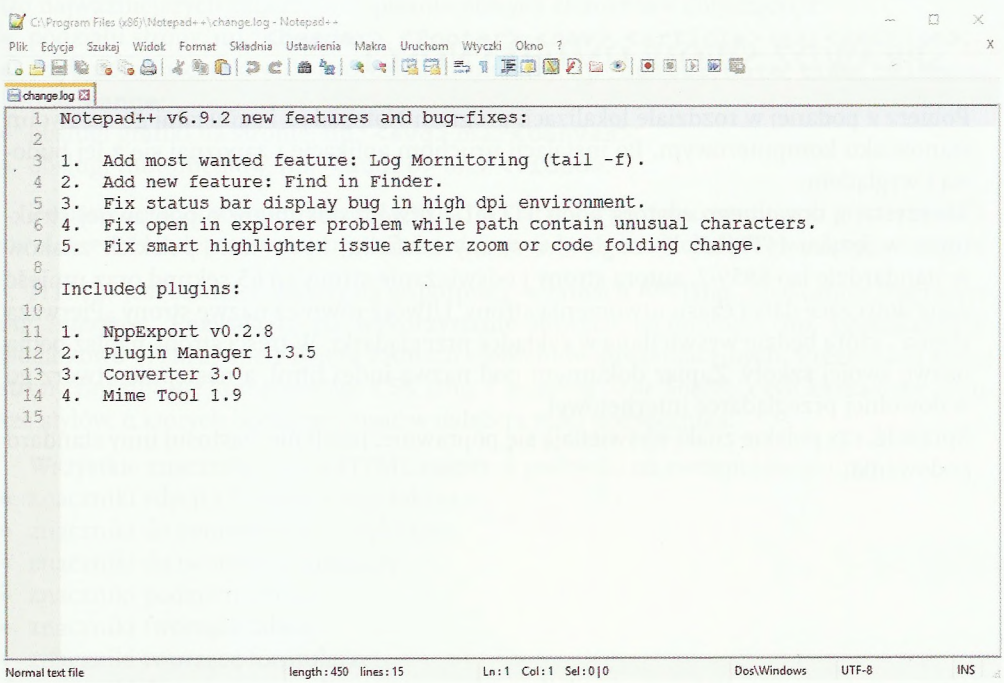
- iso-8859-2,
- utf-8,
- windows-1250.

Treść dokumentu to obszar strony znajdujący się pomiędzy znacznikami **<body>** i **</body>**. Zawiera treść dokumentu w postaci tekstu, obrazów, multimediów itp., wyświetlaną w obszarze roboczym przeglądarki internetowej.

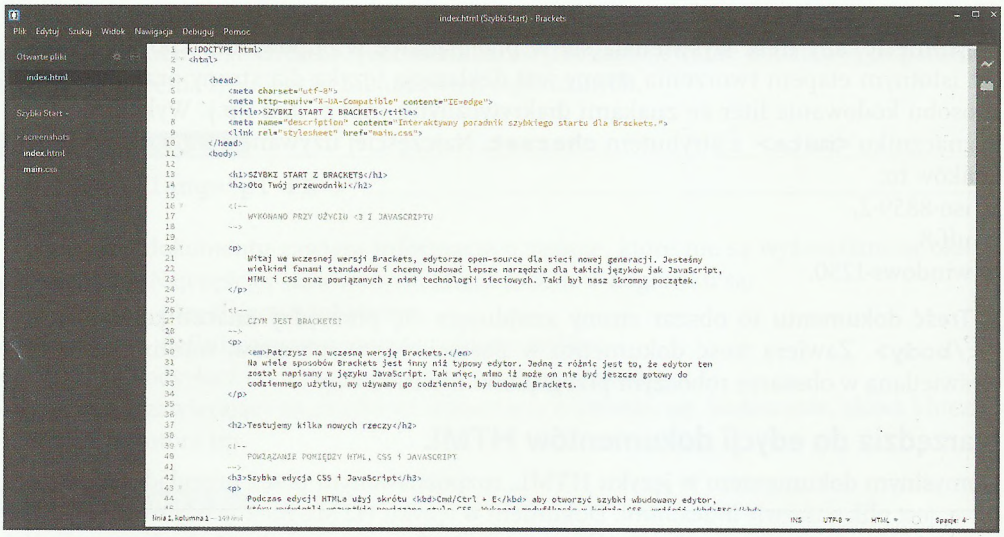
Narzędzia do edycji dokumentów HTML

Domyślnym dokumentem w języku HTML, rozpoznawanym przez przeglądarki internetowe, jest plik o nazwie **index.html**. Dokument w języku HTML można edytować w dowolnym edytorze, począwszy od najprostszego notatnika systemowego. Jednak programiści korzystają z przeznaczonych do tego celu aplikacji, np. Core Editor, Brackets lub Notepad++ oraz edytorów spełniających wymagania WYSIWYG.

W podręczniku będziemy korzystać z edytorów Notepad++ oraz Brackets. Pliki instalacyjne programów można pobrać ze stron o adresie **https://notepad-plus-plus.org/** i **http://brackets.io/index.html**.



Rys. 2.3. Okno programu Notepad++



Rys. 2.4. Okno programu Brackets

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co oznacza skrót DTD?
2. Scharakteryzuj znane ci edytory języka HTML.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Pobierz z podanej w rozdziale lokalizacji program Notepad++ i zainstaluj go na swoim stanowisku komputerowym. Po instalacji uruchom aplikację i zapoznaj się z jej budową i wyglądem.
2. Skorzystaj z dowolnego edytora kodu HTML i utwórz dokument o poprawnej strukturze w języku HTML5. W nagłówku strony zdefiniuj kodowanie polskich znaków w standardzie iso-8859-2, autora strony i odświeżanie strony co 65 sekund oraz umieść dane dotyczące daty i czasu utworzenia strony. Utwórz również nazwę strony „Pierwsza strona", która będzie wyświetlana w zakładce przeglądarki. W treści strony napisz pełną nazwę swojej szkoły. Zapisz dokument pod nazwą index.html, a następnie otwórz go w dowolnej przeglądarce internetowej.
   Sprawdź, czy polskie znaki wyświetlają się poprawnie. Jeżeli nie, zastosuj inny standard kodowania.

2.2 Podstawowe znaczniki i atrybuty w języku HTML

ZAGADNIENIA

- **Specyfikacja HTML5**
- **Elementy liniowe i blokowe**
- **Znaczniki formatowania tekstu**
- **Określanie barwy w kodzie HTML**

Jak wspomniano w poprzednim rozdziale, elementy języka HTML nazywamy znacznikami lub tagami (ang. *tag*). Opis wszystkich znaczników, atrybutów oraz ich wartości jest dostępny na stronach internetowych W3C. W tym rozdziale opiszemy najczęściej wykorzystywane znaczniki dostępne w specyfikacji HTML5.

Co nowego w HTML5?

W najnowszej specyfikacji języka HTML5 nastąpiły zmiany w stosunku do HTML4.01. Do najważniejszych zaliczamy dopisanie nowych elementów dotyczących:

- podziału strony, np.: **<header>**, **<footer>**, **<nav>**, **<article>** oraz **<section>**;
- atrybutów znaczników do tworzenia formularzy, np.: **number**, **date**, **time**, **calendar** oraz **range**;
- obsługi grafiki na stronie, np.: **<svg>** oraz **<canvas>**;
- obsługi multimediów, np.: **<audio>** oraz **<video>**.

Zmiany dotyczą również składni języka, deklaracji typu dokumentu oraz poszerzenia listy elementów zdeprecjonowanych (ang. *deprecated*), czyli takich, które wychodzą z użycia.

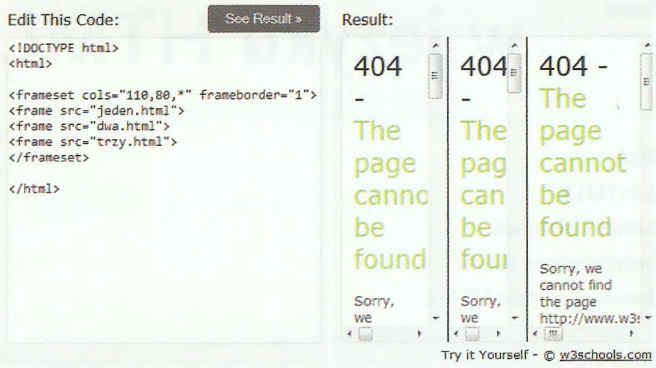
HTML5 jest bardziej otwarty na współpracę w ramach **API** (ang. *Application Programming Interface*). Pozwala to na wykorzystanie nowych technologii, np. geolokalizacji. Dodatkowo HTML5 nie zawiera żadnych elementów prezentacyjnych. Prezentacją treści strony zajmują się kompleksowo **CSS** (ang. *Cascading Style Sheets*), czyli kaskadowe arkusze stylów, o których będziemy pisać w dalszej części podręcznika.

Wszystkie znaczniki języka HTML możemy podzielić na następujące grupy:

- znaczniki edycji i formatowania tekstu;
- znaczniki do tworzenia list i wykazów;
- znaczniki do tworzenia odsyłaczy;
- znaczniki podziału strony;
- znaczniki tworzące tabele;
- znaczniki tworzące formularze;
- znaczniki do wstawiania elementów multimedialnych oraz API.

PRZYKŁAD 2.6.

W HTML5 wychodzą z użycia znaczniki tworzące ramki: **<frame>**, **<frameset>**, **<noframes>**. W związku z tym nie możemy obecnie wykonywać podziału strony według poniższego przykładu.



Rys. 2.5. Niepoprawne użycie znaczników **<frameset>** oraz **<frame>** w HTML5

Elementy liniowe i blokowe

Znaczniki języka HTML można podzielić na **liniowe** i **blokowe**. Elementy blokowe występują niezależnie od siebie. Natomiast elementy liniowe mogą występować w jednej linii, zazwyczaj wewnątrz elementów blokowych.

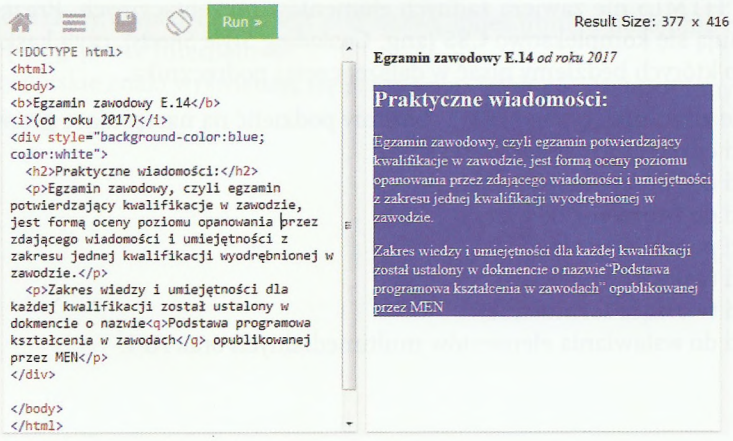
PRZYKŁAD 2.7 .

Element typu liniowego:

```
<q>przykład krótkiego cytatu w tekście </q>
```

Element typu blokowego:

```
<p>
to jest treść pierwszego
akapitu tekstu strony.
</p>
```



Atrybuty znaczników HTML

Atrybut to jeden z podstawowych elementów składni języka HTML. Wszystkie znaczniki języka HTML mogą posiadać atrybuty. Atrybut definiuje dodatkowe informacje dotyczące obiektu opisywanego przez znacznik. Atrybuty umieszczamy zawsze w znaczniku otwierającym. Dodatkowo każdy atrybut posiada określoną wartość, zapisywaną w następujący sposób:

```
<znacznik nazwa atrybutu="wartość atrybutu">
```

W tabeli poniżej opisano najczęściej stosowane atrybuty znaczników języka HTML, o których pisaliśmy wcześniej.

Tabela 2.1. Wybrane atrybuty HTML

Atrybut	Znacznik, z którym jest stosowany	Opis funkcji
Charset	`<meta>`	Określa typ kodowania znaków
Content	`<meta>`	Służy do określania wartości związanej z atrybutami name i http-equiv
Download	`<a>`	Określa, że rozpocznie się pobieranie zasobu po naciśnięciu przez użytkownika tego linku
Href	`<a>`	Opisuje adres URL dla danego odnośnika
Lang	atrybut globalny	Określa język dla danego elementu
Name	`<meta>` i inne	Określa nazwę dla danego elementu strony
Rel	`<a>`, `<link>`	Definiuje powiązanie z określonym zasobem lokalnym lub sieciowym

Listę wszystkich atrybutów dla znaczników HTML można znaleźć na stronie W3C: `https://www.w3schools.com/tags/ref_attributes.asp`

Formatowanie tekstu

Znaczniki do wyświetlania i formatowania tekstu są używane najczęściej w sekcji `<body>`. Pomimo że większość znaczników prezentacji treści została wycofana ze specyfikacji HTML5, nadal możemy korzystać z elementów określających proste formatowanie tekstu w oknie przeglądarki internetowej. W tabeli 2.2 przedstawiono podstawowe znaczniki do formatowania tekstu.

Tabela 2.2. Znaczniki do formatowania tekstu

Znaczniki	Opis
`<b>tekst</b>`	pogrubienie
`<i>tekst</i>`	kursywa
`<u>tekst</u>`	podkreślenie
`<s>tekst</s>`	przekreślenie

Znaczniki	Opis
`<mark>tekst</mark>`	podświetlenie
`<font>tekst</font>`	definiowanie czcionki
`<em>tekst</em>`	wyróżnienie tekstu
`<sub>tekst</sub>`	indeks dolny
`<sup>tekst</sup>`	indeks górny
`<strong>tekst</strong>`	czcionka mocno pogrubiona
`<dfn>tekst</dfn>`	znacznik używany do oznaczenia definicji

UWAGA

Jak widać, przedstawione znaczniki pozwalają na proste formatowanie wyświetlania tekstu na stronie internetowej. Wraz z rozwojem języka HTML liczba tych znaczników systematycznie się zmniejsza. Jest to spowodowane założeniem, że za prezentację treści strony powinny być odpowiedzialne kaskadowe arkusze stylów.

PRZYKŁAD 2.8.

Poniżej przedstawiono działanie poszczególnych znaczników do formatowania tekstu.

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<body>
Znaczniki formatowania tekstu:<br />
<b>pogrubienie</b><br />
<i>pochylenie</i><br />
<u>podkreslenie</u><br />
<s>przekreślenie</s><br />
<font face="Verdana" size="4" color="red">
formatowanie czcionki</font><br />
<em>wyróżnienie tekstu</em><br />
50<sub>indeks dolny</sub><br />
50<sup>indeks górny</sup><br />
<strong>czcionka bardzo pogrubiona</strong><br />
czcionka o jeden <small>mniejsza</small><br />
czcionka o jeden <big>większa</big><br />
linia pozioma:<br /> <hr />
</body>
</html>
```

Result:

Znaczniki formatowania tekstu:
pogrubienie
pochylenie
podkreslenie
~~przekreślenie~~
formatowanie czcionki
wyróżnienie tekstu
$50_{\text{indeks dolny}}$
$50^{\text{indeks górny}}$
czcionka bardzo pogrubiona
czcionka o jeden mniejsza
czcionka o jeden większa
linia pozioma:

Rys. 2.6. Zastosowanie znaczników do formatowania tekstu

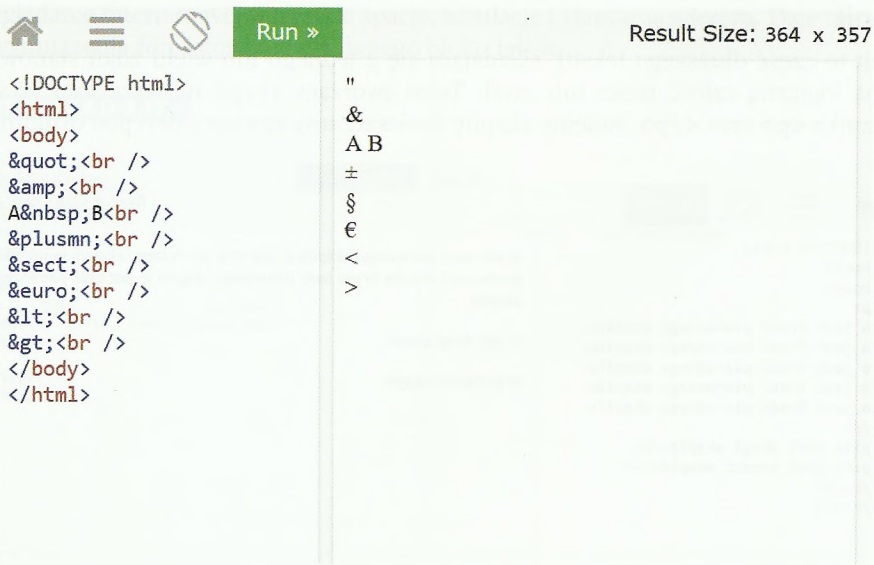
Znaki specjalne

Składnia języka HTML wymaga czasem wprowadzenia dodatkowych znaków, zwanych znakami specjalnymi. Znaki specjalne możemy wprowadzać do dokumentu hipertekstowego w postaci kodu dziesiętnego, szesnastkowego oraz nazwy własnej.

Tabela 2.3. Znaczniki edycji znaków specjalnych

Nazwa	Opis	Kod dziesiętny	Kod szesnastkowy	Przykład
&quot;	cudzysłów (ang. *quote*)	&#34;	&#x22;	"
&amp;	ampersand	&#38;	&#x26;	&
	spacja niełamliwa (twarda) (ang. *non-breaking space*)			
&plusmn;	plus-minus	&#177;	&#xB1;	±
&sect;	paragraf	&#167;	&#xA7;	§
&euro;	znak euro	&#8364;	&#x20AC;	€
&lt;	mniejszy niż	&#60;	&#x3C;	<
&gt;	większy niż	&#62;	&#x3E;	>

Poniżej zaprezentowano przykład użycia poszczególnych znaczników do edycji znaków specjalnych.



Rys. 2.7. Znaczniki do edycji znaków specjalnych

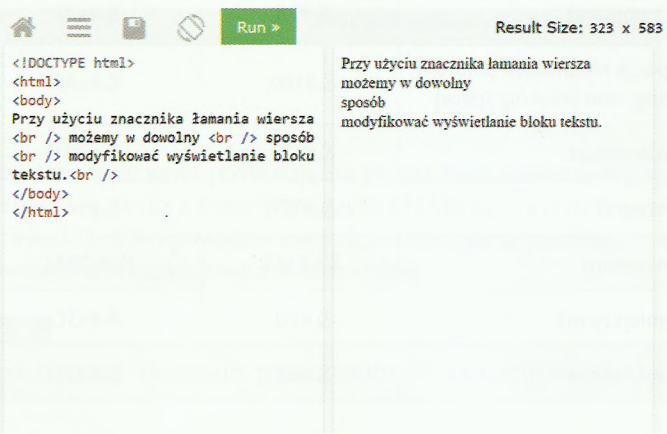
Formatowanie tekstu w bloki

Jeśli tworzymy stronę internetową, chcemy, aby jej zawartość była czytelna. Po zaprojektowaniu odpowiedniego szablonu strony należy podzielić treść na określone **bloki tekstu**. Służy do tego grupa znaczników, które przedstawimy poniżej.

Przełamanie wiersza

Znacznik przełamania wiersza **
** spełnia prostą funkcję przeniesienia wyświetlania tekstu do nowej linii. Zastosowanie go umożliwia kontrolowanie szerokości wiersza tekstu.

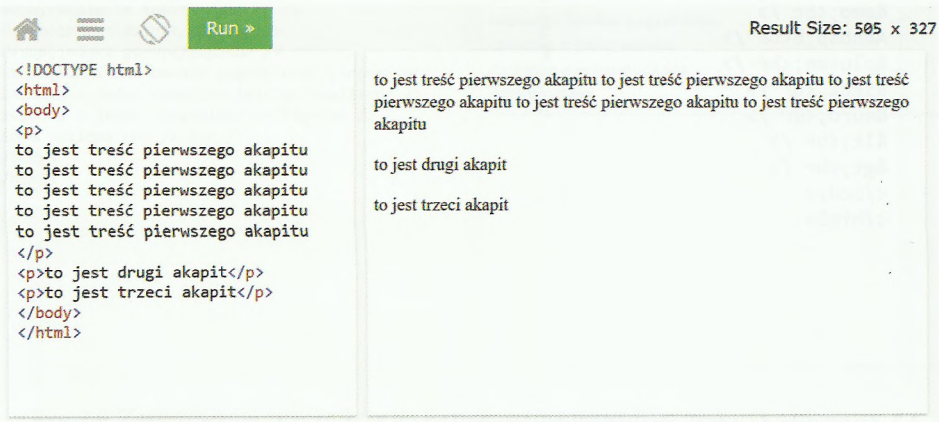
PRZYKŁAD 2.9.



Rys. 2.8. Przykład użycia znacznika **
**

Akapit

Akapit to część dłuższego tekstu, składająca się z jednego lub wielu zdań stanowiących pewną logiczną całość treści lub myśl. Tekst tworzący akapit umieszczamy wewnątrz znacznika ** ** oraz ** **. Kolejne akapity umieszczamy zawsze jeden pod drugim.

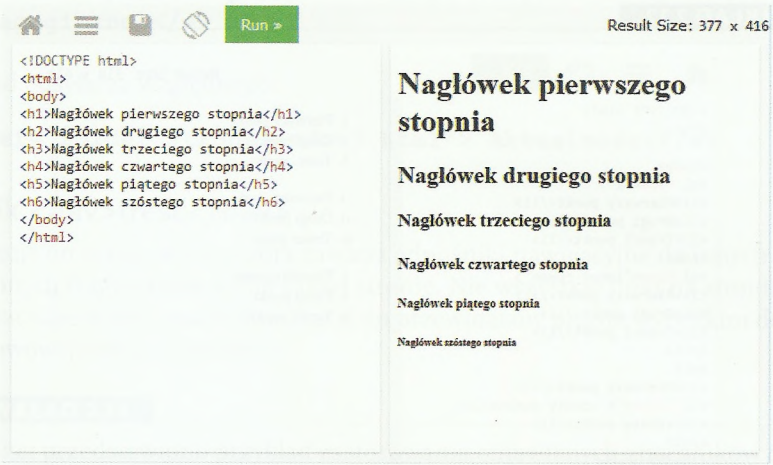


Rys. 2.9. Przykład tworzenia akapitu

Nagłówki tekstu

Nagłówki dzielą treść strony na logicznie wyodrębnione części, nadają im określone formatowanie, zależne od użytego znacznika. Znacznik nagłówka ma sześć stopni, od **<h1>** do **<h6>**. Należy jednak pamiętać, że nagłówek pierwszego stopnia definiuje największą czcionkę, a nagłówek szóstego stopnia – najmniejszą.

PRZYKŁAD 2.10.

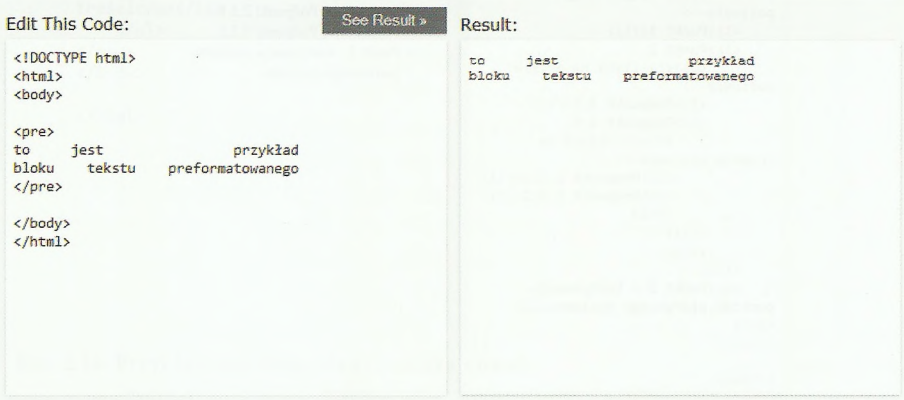


Rys. 2.10. Nagłówki w dokumencie HTML

Bloki preformatowane

Tekst wpisany wewnątrz znaczników **<pre>** i **</pre>** zachowuje podczas wyświetlania w przeglądarce internetowej wszystkie spacje, tabulacje i złamania wiersza. Daje to możliwość wymuszenia formatowania dla danego bloku tekstu.

PRZYKŁAD 2.11.



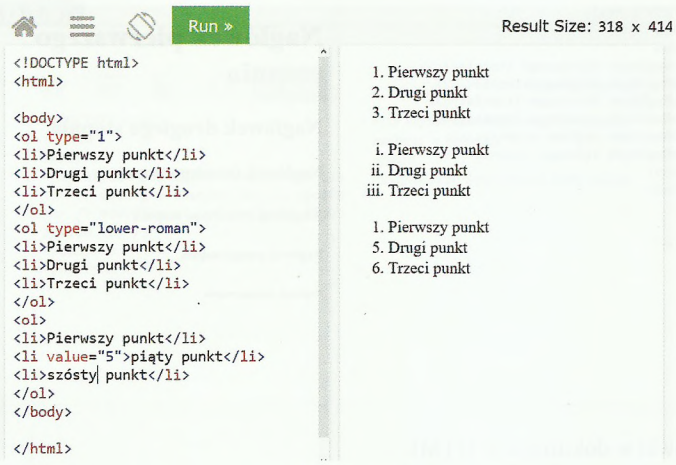
Rys. 2.11. Przykład tekstu preformatowanego

Wykazy

Wykaz daje możliwość tworzenia usystematyzowanych grup informacji. Mamy możliwość tworzenia **wykazów nieuporządkowanych** (wypunktowanych) oraz **uporządkowanych** (numerowanych).

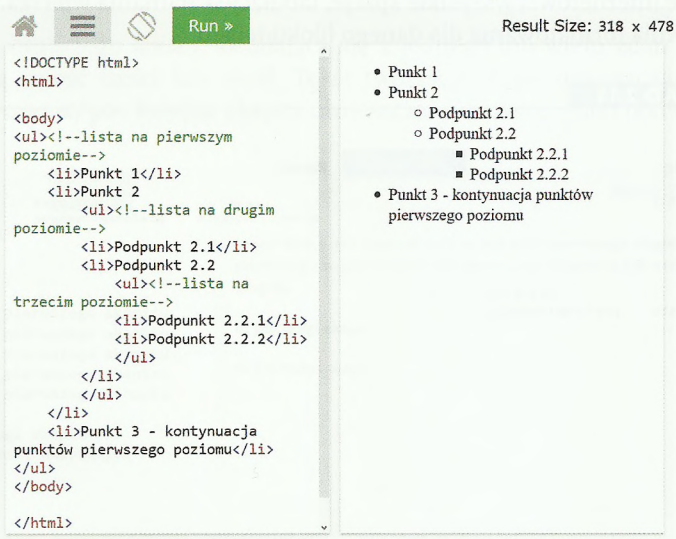
Dzięki możliwości zagnieżdżania wykazów można stworzyć listę punktów wraz z podpunktami. Podpunkty będą przesunięte bardziej w prawo w stosunku do punktów nadrzędnych, dzięki czemu wykaz będzie czytelny i zgodny z zasadami edycji tekstu.

PRZYKŁAD 2.12.



Rys. 2.12. Przykład zastosowania wykazu uporządkowanego

PRZYKŁAD 2.13.



Rys. 2.13. Przykład zastosowania wykazu nieuporządkowanego zagnieżdżonego (lub z podpunktami)

Odsyłacze

Zadaniem odsyłaczy jest utworzenie połączenia umożliwiającego przejście do innej strony internetowej, podstrony lub pliku. Rozróżniamy **odsyłacze absolutne** – do adresu internetowego w postaci formatu URL, oraz **odsyłacze względne** – do zasobu określonego ścieżką dostępu. Znacznikiem odsyłacza jest **<a>**.

Przykład odsyłacza absolutnego:

```
<a href="http:// http://www.wsip.pl"> Wydawnictwa Szkolne
I Pedagogiczne </a>
```

Przykład odsyłacza względnego:

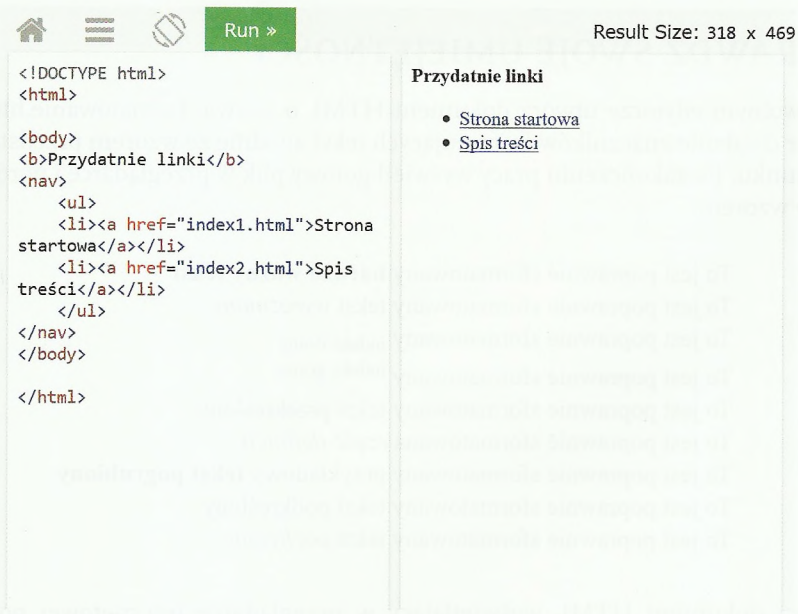
```
<a href="folder strony\index3.html"> Aktualności</a>
```

Znacznik <nav>treść</nav>

Reprezentuje on sekcję strony, która zawiera odnośniki nawigacyjne do innych stron albo do określonych fragmentów na tej samej stronie. Nie wszystkie linki na stronie powinny być umieszczane w tym znaczniku. Został on przewidziany przede wszystkim do oznaczania podstawowej nawigacji witryny.

PRZYKŁAD 2.14.

Poniżej przedstawiono przykład zastosowania omawianych znaczników do tworzenia wykazów i odsyłaczy.

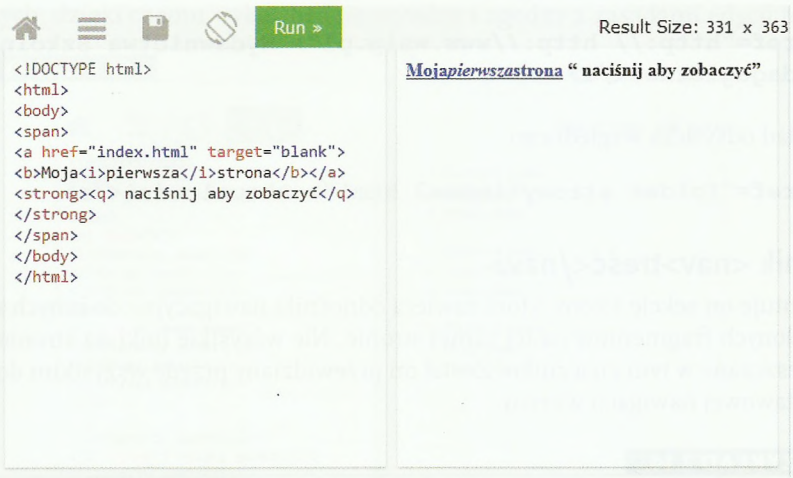


Rys. 2.14. Przykład zastosowania znacznika **<nav>**

Grupowanie elementów liniowych

Znacznik **** służy do zgrupowania elementów liniowych, takich jak tekst czy grafika. Grupowanie elementów ułatwia przypisanie do nich określonego stylu.

PRZYKŁAD 2.15.



Rys. 2.15. Zastosowanie grupowania elementów liniowych

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. W dowolnym edytorze utwórz dokument HTML o nazwie formatowanie.html, który pokaże działanie znaczników formatujących tekst zgodnie ze wzorem przedstawionym na rysunku. Po zakończeniu pracy wyświetl gotowy plik w przeglądarce i porównaj wynik ze wzorem.

To jest poprawnie sformatowany **bardzo ważny tekst**
To jest poprawnie sformatowany tekst *wyróżniony*
To jest poprawnie sformatowany indeks dolny
To jest poprawnie sformatowany indeks górny
To jest poprawnie sformatowany tekst ~~przekreślony~~
To jest poprawnie sformatowana *część definicji*
To jest poprawnie sformatowany przykładowy **tekst pogrubiony**
To jest poprawnie sformatowany tekst podkreślony
To jest poprawnie sformatowany tekst *pochylony*

2. Utwórz dokument HTML wyświetlający w przeglądarce internetowej pogrubiony i pochylony tekst:

Zegarek męski "A.Lange&Söhne" © 2017 Model Lange1 cal.L901.0™ Cena ±31000 €

3. Utwórz dokument HTML wyświetlający wykaz w postaci spisu treści według wzoru.

I. Podzespoły systemu komputerowego
1. Płyty główne
a. Architektura płyt głównych
- Układy Chipset
- Złącza I/O
- Magistrale FSB

b. Standardy płyt głównych
2. Pamięci masowe
a. Dyski twarde HDD
b. Napędy optyczne
c. Macierze dyskowe
3. Urządzenia peryferyjne

Uzupełnij podany wykaz w punkcie 3 o linki do czterech stron producentów urządzeń peryferyjnych.

2.3 Barwa na stronie internetowej

ZAGADNIENIA
- Model RGB
- Model HSL
- Model CMYK
- Kodowanie barw w HTML5 i CSS3

Barwy są wykorzystywane na prawie każdym etapie tworzenia dokumentu hipertekstowego oraz cyfrowego przetwarzania obrazów. Obecnie prezentację treści strony, a co za tym idzie, barwy różnych jej elementów, definiuje się wewnątrz arkuszy stylów (CSS). Jednak nie jest zabronione używanie wartości barwy w atrybutach znaczników języka HTML. W tym rozdziale pokażemy, w jaki sposób z tego korzystać.

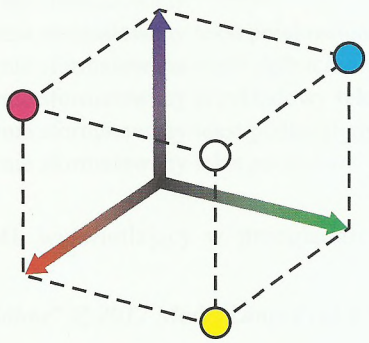
Barwa jest wrażeniem wzrokowym powstałym na skutek oddziaływania fal elektromagnetycznych z pasma częstotliwości światła widzialnego (między 700 nm a 400 nm).

Modele barw

Aby jednoznacznie określić daną barwę na stronach internetowych, stosuje się określone modele barw oraz metody ich zapisu. **Modele barw** opisują określone palety kolorów. Wyróżniamy cztery podstawowe modele barw stosowanych w grafice komputerowej oraz tworzeniu stron internetowych.

a) Model RGB

Barwa jest tworzona poprzez złożenie z trzech podstawowych kolorów: R – czerwonego (*red*), G – zielonego (*green*) oraz B – niebieskiego (*blue*). Model ten jest stosowany do prezentacji na ekranie monitora czy obrazu z projektora. Wykorzystuje się go zarówno do zapisu barwy w kodzie HTML5, jak i CSS3. W specyfikacji CSS3 dopuszczono do stosowania również pokrewny model **RGBA** z dodatkowym współczynnikiem kanału alfa. Opiszemy go dokładnie w rozdziale dotyczącym CSS3.



Rys. 2.16. Sześcian w przestrzeni parametrów R, G, B

b) Model HSL

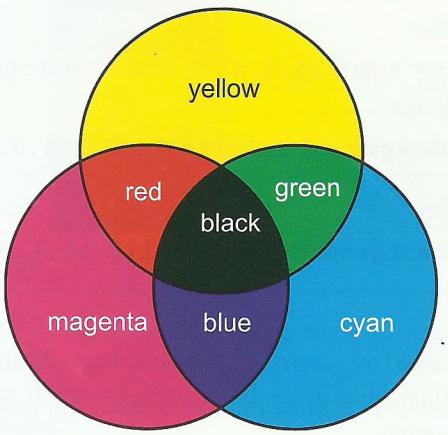
Każdą barwę w tym modelu można opisać za pomocą trzech atrybutów:

- **odcienia (H)**, powiązanego z określoną długością fali;
- **nasycenia (S)**, określającego moc barwy;
- **jasności (L)**, czyli intensywności światła w danej barwie.

Model ten powstał na potrzeby telewizji. Używany jest również model **HSLA**, który jest rozwinięciem omawianego nieco wcześniej modelu barw HSL, dodatkowo uzupełnionym o obsługę przezroczystości. Modele te są akceptowane przez arkusze CSS3.

c) Model CMYK

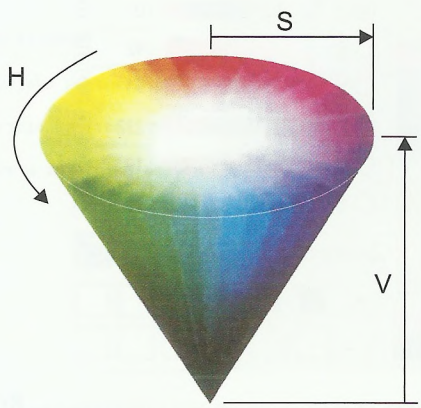
Barwa jest tworzona z czterech kolorów: C – cyjanu, M – magenty, Y – żółtego (*yellow*) oraz K – czarnego (*black*). Model ten jest stosowany do przygotowania wydruków.



Rys. 2.17. Model CMYK

d) Model HSV

Oparty jest na trzech parametrach: H – odcieniu (*hue*), S – nasyceniu (*saturation*), V – wartości (*value*). Model jest rozpatrywany jako stożek, którego podstawą jest koło barw.



Rys. 2.18. Stożek przestrzeni barw HSV

Kodowanie barw w HTML5 i CSS3

Zapis kodów kolorów w dokumentach HTML5 i CSS3 przeprowadza się według określonych notacji. W języku HTML są stosowane nazwy barw oraz wartości zapisane kodem szesnastkowym lub paletą kolorów RGB. W arkuszach stylów CSS3 jest dopuszczone stosowanie dodatkowo zapisów w notacji palety RGBA, HSL oraz HSLA.

PRZYKŁAD 2.16.

W języku HTML dopuszczamy zapis barwy w następujący sposób:

1. zapis słowny

```
<body bgcolor="red"> treść strony </body>
```

2. zapis barwy w kodzie szesnastkowym

```
<body bgcolor="#FF0000"> treść strony </body>
```

3. zapis w kodzie RGB

```
<body bgcolor="rgb(255,0,0)"> treść strony </body>
```

4. zapis w kodzie RGBA

```
<div style="background-color:rgba(0,255,0,0.1)> treść
bloku</div>
```

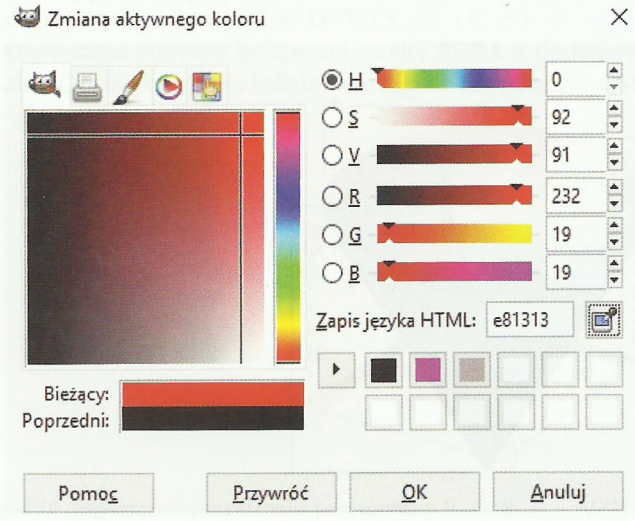
5. zapis w kodzie HSL

```
<div style="background-color:hsl(0,100%, 50%)>treść bloku</
div>
```

UWAGA

Zapis w notacji szesnastkowej czasami można skrócić, np. przez podanie zamiast wartości #00FF00 – wartości #0F0.

Do sprawdzenia wartości kodowej określonej barwy można użyć dowolnego edytora graficznego. Poniżej przedstawiono okno edytora koloru programu GIMP, w którym można odczytać kod wybranej barwy.



Rys. 2.18. Okno edycji barw w edytorze GIMP

Specyfikacja HTML4.01 zawiera zestawienie 16 barw według nazw i kodu szesnastkowego. Poniżej przedstawiono tabelę zawierającą możliwe notacje tych barw.

Kolor	Nazwa po angielsku	Zapis szesnastkowy	Kolor	Nazwa po angielsku	Zapis szesnastkowy
cyjan	aqua	#00FFFF	granatowy	navy	#000080
czarny	black	#000000	oliwkowy	olive	#808000
niebieski	blue	#0000FF	fioletowy	purple	#800080
fuksja	fuchsia	#FF00FF	czerwony	red	#FF0000
zielony	green	#008000	srebrny	silver	#C0C0C0
szary	grey	#808080	morski	teal	#008080
limonka	lime	#00FF00	biały	white	#FFFFFF
wiśniowy	maroon	#800000	żółty	yellow	#FFFF00

Rys. 2.19. Barwy w HTML4.01

W specyfikacji HTML5 dostępność barw jest o wiele większa. Na stronie organizacji W3C (**https://www.w3schools.com/html/html_colors.asp**) dostępny jest mikser barw RGB pozwalający utworzyć własny kod, który najlepiej będzie pasował do danego elementu strony internetowej.

W dokumencie HTML definiujemy barwę przez wpisanie jej nazwy lub kodu jako wartości określonego atrybutu znacznika.

PRZYKŁAD 2.17.

Run »

Result Size: 410 x 327

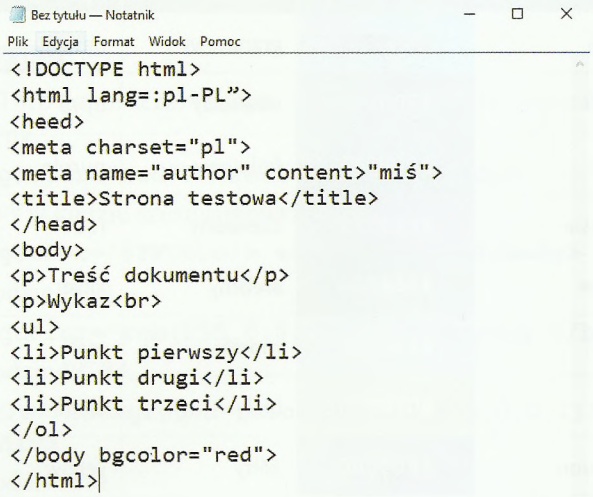
```
<!DOCTYPE html>
<html>
<body bgcolor="silver" text="#3300FF"
link="black" vlink="blue">
Przykład wykorzystania barwy jako wartości
atrybutów znaczników:<br />
<font color="red">czerwona czcionka</font><br
/>
tekst w kolorze zdefiniowanym w znaczniku body.
</body>

</html>
```

Przykład wykorzystania barwy jako wartości atrybutów znaczników:
czerwona czcionka
tekst w kolorze zdefiniowanym w znaczniku body.

Rys. 2.20. Przykład zastosowania kodowania barw w dokumencie HTML4.01

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przeanalizuj załączony kod i wskaż ewentualne błędy w zakresie składni języka oraz poprawności użycia znaczników.



2. Poszukaj w dostępnych źródłach (np. https://www.w3schools.com) informacji na temat poprawności stosowania atrybutów: color, bgcolor, align w składni języka HTML5. Zapisz w zeszycie wnioski.
3. Wypełnij tabelę – podaj nazwy modeli barw zapisane za pomocą kodu HTML.

Zapis barwy w kodzie	Nazwa modelu barw
(0,100%, 50%)	
#FF00FF	
(255,0,0)	
(0,255,0,0.1)	

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym charakteryzuje się odsyłacz względny?
2. Co to są znaczniki zdeprecjonowane?

2.4 Podział strony i obsługa danych w języku HTML

ZAGADNIENIA

- **Podział strony na bloki i sekcje**
- **Tabele w dokumentach hipertekstowych**
- **Obsługa danych za pomocą formularzy**

Istotnym etapem tworzenia strony internetowej jest utworzenie jej struktury z wydzieleniem poszczególnych bloków, kontenerów lub części spełniających określoną funkcję. W obecnych czasach, kiedy dużą część witryn projektuje się z wykorzystaniem filozofii zarządzania **CMS** (ang. *Content Management System*), stworzenie jednolitej struktury, z której korzystają wszystkie strony danej witryny, jest kluczowe dla sukcesu całego projektu.

W starszych standardach HTML do podziału strony najczęściej używano ramek. Jednak z czasem stały się one przestarzałe i obecnie znaczniki tworzenia ramek znalazły się w grupie elementów zdeprecjonowanych.

Bloki

W HTML4 do podziału strony na części funkcjonalne stosowano powszechnie elementy zwane blokami lub kontenerami. Elementy wpisane między znacznikami ** ** i ** ** łatwo można wypełnić treścią, załadować określoną podstronę oraz zastosować wewnętrzne lub zewnętrzne formatowanie za pomocą stylów (tworzymy wtedy tzw. kontenery CSS).

Do najważniejszych atrybutów znacznika ** ** zaliczamy:

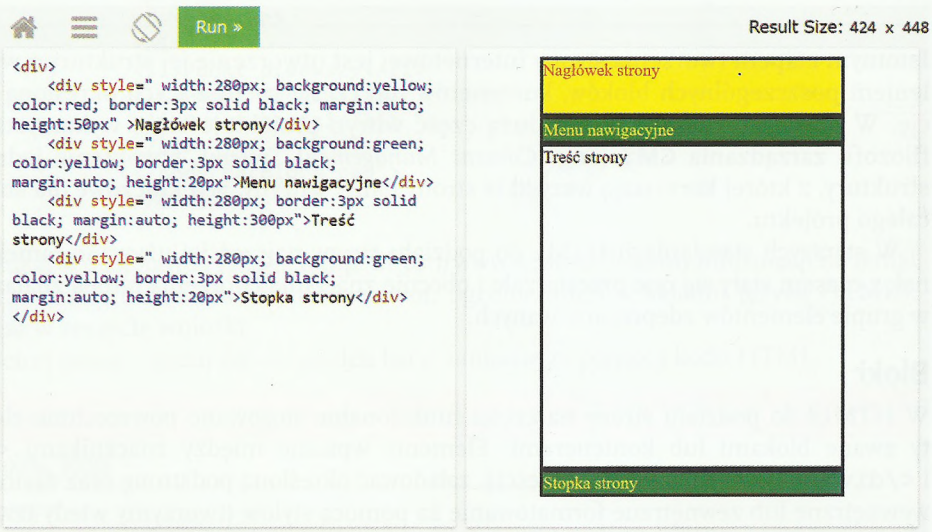
Atrybuty	Opis / znacznik
Class	klasa CSS
Id	identyfikator elementu
Style	styl elementu
Title	tekst do wyświetlenia po najechaniu myszką
Align	wyrównanie zawartości elementów blokowych w poziomie; nie jest wspierany przez standard HTML5

PRZYKŁAD 2.18.

Naszym zadaniem jest utworzenie strony internetowej podzielonej na cztery bloki: nagłówek strony, menu nawigacyjne, treść strony oraz stopka strony. Wszystkie bloki powinny mieć szerokość 280 px, obramowanie czarne (oprócz głównego kontenera), pogrubione o szerokości 3 px oraz domyślny margines zewnętrzny. Ponadto poszczególne bloki powinny być formatowane w następujący sposób:

- nagłówek strony – tło żółte, kolor czcionki czerwony oraz wysokość bloku 50 px;
- menu nawigacyjne oraz stopka strony – tło zielone, kolor czcionki żółty oraz wysokość bloku 20 px;
- treść strony – wysokość bloku 300 px.

Po zastosowaniu znacznika ** ** z atrybutem style i określonymi w treści zadania wartościami uzyskamy efekt pokazany na rysunku 2.21.



Rys. 2.21. Przykład podziału strony za pomocą znacznika ** **

W praktyce bardziej poprawne jest formatowanie za pomocą zewnętrznych arkuszy stylów, o czym będziemy pisać w dalszej części podręcznika. W takim przypadku każdy znacznik ** ** jest traktowany jako kontener (pojemnik) stylu CSS i powinien mieć swoją nazwę. Do nadania nazwy używa się atrybutu class lub id. Różnica między nimi polega na tym, że atrybutu id o takiej samej wartości (w tym wypadku nazwie) można użyć tylko jeden raz w obrębie jednej strony. Natomiast atrybut class o danej nazwie można zastosować wielokrotnie w obrębie jednej strony internetowej.

PRZYKŁAD 2.19.

Poniżej pokazano przykład zastosowania identyfikatora elementu **<id>**, pozwalającego na formatowanie poszczególnych kontenerów odpowiednio zdefiniowanym zewnętrznym arkuszem stylów.

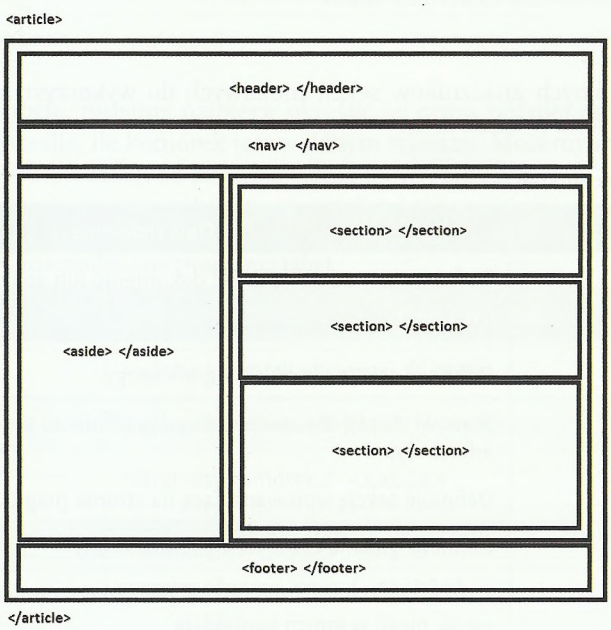
```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="plik_styl.css" type="text/css">
</head>
<body>
<div id="Sekcja glowna strony">
    <div id="naglowek">Nagłówek strony</div>
    <div id="menu">Menu nawigacyjne</div>
    <div id="tresc">Treść strony</div>
    <div id="stopka">Stopka strony</div>
</div>
</body>
</html>
```

Rys. 2.22. Zastosowanie nazw atrybutu id znacznika ** **

Sekcje

Wraz z opublikowaniem specyfikacji HTML5 wprowadzono nowe elementy blokowe, które mają zastąpić opisywany wcześniej znacznik ** ** poprzez precyzowanie roli określonych fragmentów dokumentu hipertekstowego. Tego typu elementy języka HTML nazywamy znacznikami semantycznymi. Takie znaczniki w sposób jednoznaczny są interpretowane przez przeglądarkę i nie wymagają przy tym użycia dodatkowych atrybutów definiujących ich rolę.

Do najczęściej wykorzystywanych znaczników sekcji zaliczamy: **<article>**, **<header>**, **<nav>**, **<section>**, **<aside>** czy **<footer>**. Rysunek 2.23 ilustruje funkcję poszczególnych znaczników sekcji w strukturze strony.



Rys. 2.23. Rola poszczególnych znaczników sekcji

PRZYKŁAD 2.20.

Definiowanie stopki za pomocą znacznika :

```
<div id="stopka"> Stopka strony </div>
```

Definiowanie stopki za pomocą znacznika <footer>:

```
<footer> Stopka strony </footer>
```

PRZYKŁAD 2.21.

Na rysunku przedstawiono przykład zastosowania znacznika <nav>.

Rys. 2.24. Zastosowanie znacznika **<nav>**

Opis poszczególnych znaczników sekcji możliwych do wykorzystania w standardzie HTML5 przedstwiono w tabeli 2.4.

Tabela 2.4. Opis znaczników sekcji

Znacznik	Opis znaczenia
<article>	Definiuje samodzielną część dokumentu lub aplikacji
<aside>	Definiuje sekcję na stronie
<figure>	Definiuje sekcję dla obiektu graficznego
<footer>	Stanowi stopkę dla zawierającego ją elementu sekcji i zawiera dane informacyjne
<header>	Definiuje sekcję wprowadzającą na stronie (nagłówek)
<main>	Definiuje główną część nawigacyjną strony
<mark>	Zaznacza lub uwydatnia tekst w jednym dokumencie w celu odwołania się do niego w innym kontekście

Znacznik	Opis znaczenia
`<nav>`	Reprezentuje sekcję strony, która zawiera odnośniki nawigacyjne do innych stron albo do określonych fragmentów na tej samej stronie
`<section>`	Definiuje tematyczną grupę treści, zwykle zawierającą nagłówek
`<details>`	Definiuje interaktywne sekcje widoczne np. po naciśnięciu na daną pozycję menu
`<summary>`	Definiuje widoczny element dla sekcji `<details>`
`<time>`	Definiuje blok daty i czasu

Używanie dodatkowo formatowania i pozycjonowania bloków za pomocą kaskadowych arkuszy stylów CSS daje możliwość dowolnego zmieniania ich położenia i wyglądu.

Tabele

Tabele pozwalają na łatwą i czytelną prezentację danych, obrazów, tekstu i innych elementów na stronach internetowych. Do tworzenia tabel używamy znaczników z tabeli 2.5.

Tabela 2.5. Znaczniki tworzenia tabel

Znaczniki	Opis
`<table> </table>`	znaczniki określające ramy tabeli
`<tr> </tr>`	znaczniki wiersza tabeli
`<td> </td>`	znaczniki komórki tabeli
`<th> </th>`	komórka nagłówkowa w tabeli; zawiera się wewnątrz wierszy
`<caption> </caption>`	tytuł tabeli

Aby stworzyć tabelę, musimy najpierw określić jej ramy, później podzielić ją na wiersze, a na końcu określić, ile komórek jest w danym wierszu. Możemy również skorzystać z określonych atrybutów tabeli do sformatowania jej wyglądu.

Najważniejsze atrybuty znaczników tworzących tabele przedstwiono w tabeli 2.6.

Tabela 2.6. Atrybuty znaczników do tworzenia tabel

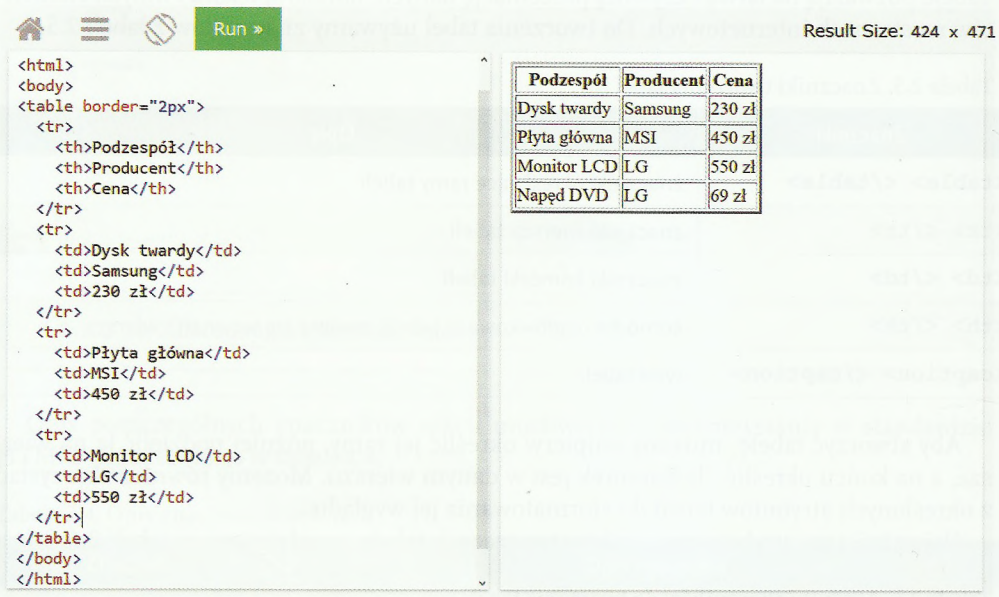
Atrybuty	Opis / znacznik
`Colspan`	scalenie komórek w wierszu / `<td>`
`Rowspan`	scalenie komórek w kolumnie / `<td>`
`cellpadding`	marginesy komórek / `<table>`
`cellspacing`	rozmiar odstępów między komórkami / `<table>`
`Border`	rozmiar obramowania / `<table>`
`Width`	szerokość elementu tabeli / `<table>`, `<td>`

Atrybuty	Opis / znacznik
`Height`	wysokość elementu tabeli / `<table>`, `<td>`
`Valign`	wyrównanie tekstu w pionie w komórkach w wierszu
`Align`	wyrównanie tekstu w komórkach w wierszu
`Bgcolor`	kolor wypełnienia

Wygląd tabeli również możemy określać za pomocą stylów wewnętrznych lub zewnętrznych, co opiszemy w następnych rozdziałach.

PRZYKŁAD 2.22.

Poniżej przedstawiono przykład tabeli prostej złożonej z 3 kolumn i 5 wierszy, z których jeden jest wierszem nagłówkowym.



Rys. 2.25. Przykład tabeli w HTML

Za pomocą atrybutów opisanych w tabeli 2.6 możemy formatować wygląd komórek tabeli oraz ich wypełnienia, co ilustruje rysunek 2.26.

Run »

Result Size: 424 x 448

```
<!DOCTYPE html>
<html>
<body>
<table border height=300>
<tr>
<td align="center" bgcolor="#FF0000">wyrównanie -
center</td>
<td align="left" bgcolor="#FF3F00">wyrównanie -
left </td>
<td align="right" bgcolor="#088000">wyrównanie -
right</td>
</tr>
<tr>
<td bgcolor="#FF00FF" valign="top">wyrównanie w
pionie - top</td>
<td bgcolor="#00F2FF" valign="bottom">wyrównanie
w pionie - bottom</td>
<td bgcolor="#000AAF" valign="MIDDLE">wyrównanie
w pionie - middle</td>
</tr>
</table>
</body>
</html>
```

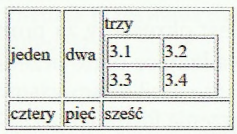
Rys. 2.26. Formatowanie wyglądu tabeli

Możliwe jest zagnieżdżanie tabel w komórkach tabeli nadrzędnej. W tym celu należy zdefiniować tabelę w wybranej komórce tabeli nadrzędnej. Na przykład:

Run »

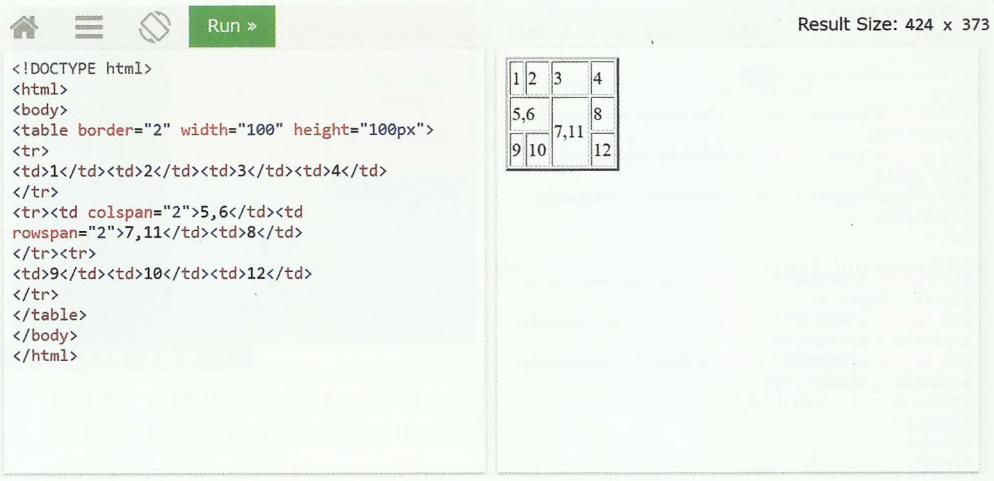
Result Size: 424 x 448

```
<!DOCTYPE html>
<html>
<body>
<table border width=200>
<tr>
<td>jeden</td><td>dwa</td><td>trzy
<table border width=100>
<tr><td>3.1</td><td>3.2</td></tr>
<tr><td>3.3</td><td>3.4</td></tr>
</table>
</td>
</tr>
<tr><td>cztery</td><td>pięć</td>
<td>sześć</td></tr>
</table>
</body>
</html>
```



Rys. 2.27. Zagnieżdżanie tabel

Możemy również stosować łączenie zarówno kolumn, jak i wierszy. Na przykład:



Rys. 2.28. Łączenie kolumn i wierszy

Formularze

Jednym z elementów interaktywnych w dokumentach HTML jest **formularz**. Na stronach internetowych służy on do zbierania i wysyłania określonych danych w założonym przez autora celu. Wprowadzone dane możemy przesłać na adres poczty mailowej lub wykorzystać w uruchomionym skrypcie zarówno po stronie użytkownika (klienta), jak i zdalnego serwera.

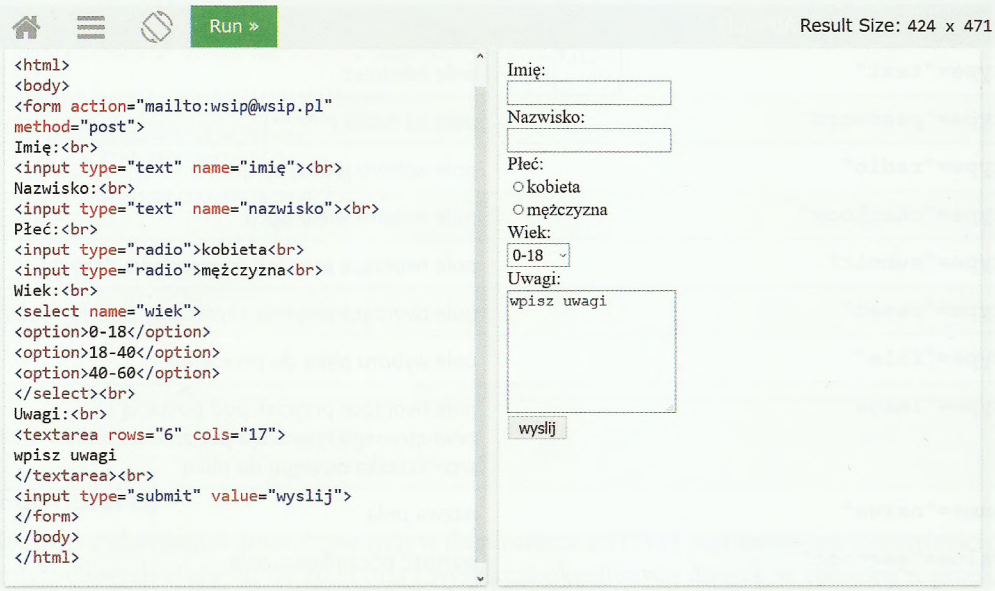
Obszar formularza jest określany za pomocą znaczników **<form>** i **</form>**. Podstawowe znaczniki wprowadzające dane przedstawia tabela 2.7.

Tabela 2.7. Znaczniki tworzenia formularza

Znaczniki	Opis
<input>	Tworzy pola formularza
<select>	Tworzy listy rozwijane
<textarea>	Tworzy pola opisowe
<fieldset>	Obramowanie pól formularza
<legend>	Opis pola formularza
<label>	Tworzy etykietę pola dla znacznika <input>

PRZYKŁAD 2.23.

Na rysunku poniżej przedstawiono przykład prostego formularza z opcją przesłania danych na adres e-mail podany w kodzie.



Rys. 2.29. Formularz w HTML

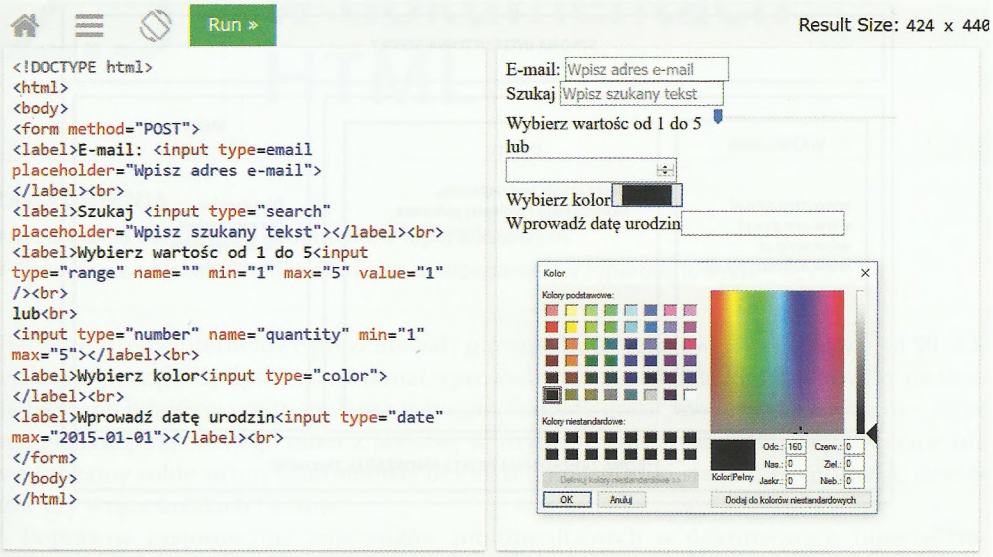
Jak widać na rysunku, do przesłania danych tworzy się przycisk za pomocą typu submit z etykietą **wyślij**. Natomiast metodę i miejsce docelowe przesyłania danych z formularza definiujemy atrybutami **action** i **method** w znaczniku **<form>** tworzącym blok formularza.

Dla funkcjonalności formularza jest istotne określenie typu i innych atrybutów znacznika **<input>**. W kolejnych specyfikacjach języka HTML poszerzano grupę typów danych wprowadzanych do pól formularzy. W specyfikacji HTML5 mamy do dyspozycji dodatkowo następujące typy:

- color,
- date,
- datetime-local,
- email,
- month,
- number,
- range,
- search,
- tel,
- time,
- url,
- week.

PRZYKŁAD 2.24.

Poniżej przedstawiono formularze wykorzystujące nowe typy znacznika <input> w HTML5



Rys. 2.30. Przykład typów znacznika **<input>** w HTML5

Znacznikami niesemantycznymi są np. ** ** lub ****, ponieważ nic nie mówią nam o roli, jaką mają pełnić na stronie.

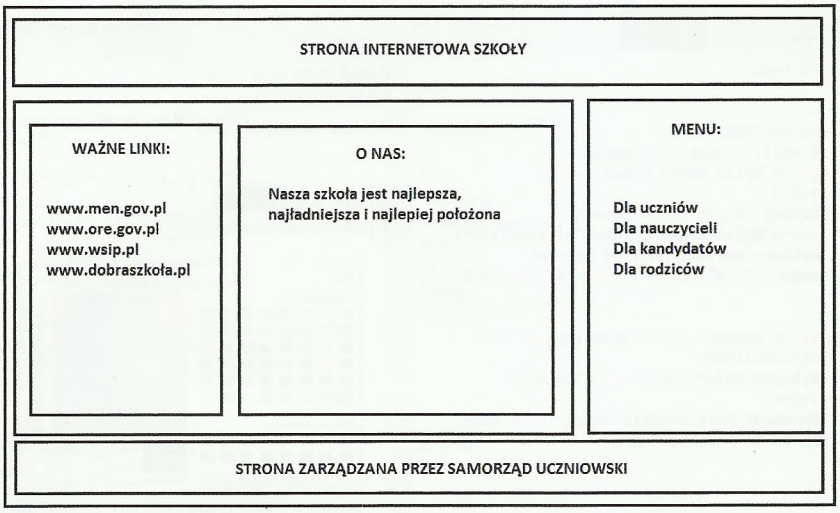
Do znaczników semantycznych zaliczamy takie znaczniki, jak **<table>** czy **<header>**, ponieważ zarówno twórca strony, jak i przeglądarka wiedzą, jaką rolę pełnią one w strukturze strony.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wypisz w zeszycie pięć znaczników semantycznych i pięć niesemantycznych.
2. Utwórz dokument hipertekstowy, w którym zdefiniujesz tabelę zgodną ze wzorem podanym poniżej. Pamiętaj, że kolor obramowania tabeli ma być szary, a szerokość obramowania powinna wynosić 2 px. Gotowy plik zapisz pod nazwą tabela.html i otwórz w dowolnej przeglądarce internetowej.

Rozkład zajęć dodatkowych z informatyki				
Rok szkolny	Miesiąc	Dzień	Godzina	Sala
2017/2018	wrzesień–styczeń	wtorek	8.00	116
	luty–kwiecień	czwartek	14.00	
	maj–czerwiec	wtorek	16.00	

3. Utwórz dokument hipertekstowy, który zdefiniuje podział strony internetowej według podanego wzoru. Do wykonania zadania wykorzystaj znaczniki zgodne ze specyfikacją HTML5. Gotowy plik zapisz pod nazwą szablon.html.



SPRAWDŹ SWOJĄ WIEDZĘ

1. Omów zmiany w zakresie możliwości tworzenia formularzy wprowadzono w specyfikacji języka HTML5 w stosunku do poprzednich wersji.
2. Jaką rolę pełnią znaczniki <nav>, <input>, <td> ?

2.5 Grafika i multimedia w dokumentach HTML

ZAGADNIENIA

- **Wstawianie obrazów do dokumentów hipertekstowych**
- **Wstawianie obiektów multimedialnych do dokumentów hipertekstowych**

Pierwsze strony internetowe prezentowały głównie treści tekstowe. Na początku lat 90. XX wieku rozpoczęła się dyskusja na temat wprowadzenia znacznika, który wstawiłby na stronę bitmapę lub pixmapę (popularne wówczas formaty graficzne) poprzez pobranie przez przeglądarkę określonego pliku z podanej w atrybucie znacznika lokalizacji. Obecnie nie wyobrażamy sobie strony internetowej bez elementów grafiki statycznej, animacji, dźwięków czy wręcz krótkich filmów.

Poprawne zastosowanie znaczników multimedialnych w dokumentach hipertekstowych jest uzależnione od rodzaju i wersji przeglądarki internetowej. Dlatego przed użyciem konkretnych technologii konieczne jest ustalenie przeglądarek i urządzeń, na których pracować ma dana strona internetowa.

Grafika statyczna

Do osadzania obrazów statycznych (zdjęć, rysunków, symboli lub wykresów) służy znacznik **<img>**. Jednak w dokumencie hipertekstowym nie wklejamy gotowego obrazu, tylko wskazujemy ścieżkę URL do pliku, który otworzy się w oknie przeglądarki po uprzednim jego pobraniu. Dlatego też poza samym znacznikiem jest niezbędne zastosowanie atrybutu **src="adres URL do pliku"**.

```
<img src="folder\obraz.jpg">
```

Z czasem możliwość osadzania różnych formatów plików graficznych została znacznie poszerzona, a stare bitmapy (bmp) coraz rzadziej wykorzystujemy w technologii tworzenia stron internetowych.

Do rozszerzeń plików używanych obecnie najczęściej dla statycznych obrazów cyfrowych należą: **jpg**, **jpeg**, **png** oraz **gif**.

Jaki format jest najlepszy do stosowania na stronach internetowych? Aby odpowiedzieć na to pytanie, musimy poznać różnice między poszczególnymi formatami.

Format **JPG** (**JPEG**) stwarza możliwość kompresji obrazów z płynnymi przejściami tonalnymi. Nadaje się więc idealnie do osadzania na stronie wielobarwnych zdjęć z płynnymi przejściami między kolorami.

Format **GIF** jest dobry do prezentacji tworzonych rysunków, banerów czy przycisków, korzystających z palety 256 kolorów. Warto pamiętać, że format ten obsługuje przezroczystość i animację.

Format **PNG** jest przeznaczony do zastosowań internetowych, a powstał jako następca formatu GIF. PNG nie ma limitu 256 kolorów i dysponuje lepszą kompresją. Wspiera

on także przezroczystość kanału alfa, czyli może być częściowo przezroczysty, całkowicie przezroczysty lub zupełnie nieprzezroczysty.

Znacznika **<img>** używamy również z atrybutami. Jednym z nich jest atrybut **alt**, który pozwala na dodanie opisu, etykiety obrazu. Etykieta ta może się pojawić w miejscu obrazu, jeśli np. źle określono adres obrazu lub został on usunięty. Dodatkowo, stosując atrybuty **width** (szerokość) oraz **height** (wysokość), można wykonać skalowanie obrazu do wymaganego na stronie rozmiaru.

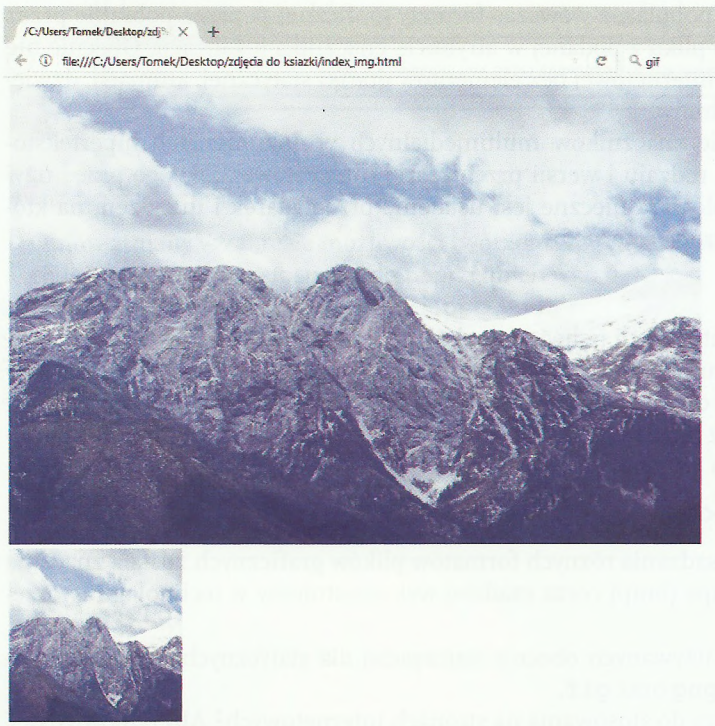
PRZYKŁAD 2.25.

Poniżej przedstawiono dwa przykłady osadzania obrazu w kodzie HTML. Wynik i różnice w prezentacji obrazu pokazuje rysunek 2.31.

Fragment kodu:

```
<img src="giewont.jpg" alt="Giewont">
<img src="giewont.jpg" alt="Giewont" width="200" height="200">
```

Wynik działania kodu:



Rys. 2.31. Zastosowanie znacznika **<img>**

Mapa obrazu

Z umieszczonej na stronie grafiki możemy utworzyć mapę odsyłaczy. Dzielimy wtedy obszar obrazu na określone części i przypisujemy im określony odsyłacz. Po najechaniu kursorem myszy na dany obszar możemy uruchomić zdefiniowany w kodzie odsyłacz. Bazujemy tutaj na znaczniku **<img>** i definiujemy nazwę mapy atrybutem **usemap="#nazwa"**.

Znacznik **<map>** służy do stworzenia mapy obrazu po stronie przeglądarki internetowej. Natomiast znacznik **<area>** jest zawsze umieszczony wewnątrz znacznika **<map>** i służy on do tworzenia obszarów na mapie obrazu. Następnie za pomocą atrybutów **shape** (wartości: default, rect, circle, poly) określamy typ obszaru, a **coords** – współrzędne obszaru (**rect** przyjmuje cztery wartości **coords=**"od lewej, od góry, szerokość, wysokość"; **circle** przyjmuje trzy wartości **coords=**"od lewej, od góry, promień okręgu"; **poly** przyjmuje wartości, które są współrzędnymi wierzchołków danego wielokąta).

PRZYKŁAD 2.26.

Na rysunku 2.32 przedstawiono kod HTML tworzący podział rysunku kwadrat.jpg na cztery obszary, dla których zdefiniowano odnośniki.

```
<!DOCTYPE html>
<html>
  <body>
<img src="kwadrat.jpg" alt="4kwadraty" usemap="#kw" width="200px" height="200px">

<map name="kw">
  <area shape="rect" coords="0,0,100,100" alt="górny lewy" href="1.htm">
  <area shape="rect" coords="0,100,100,200" alt="dolny lewy" href="2.htm">
  <area shape="rect" coords="100,0,200,100" alt="górny prawy" href="3.htm">
  <area shape="rect" coords="100,100,200,200" alt="dolny prawy" href="4.htm">
</map>
  </body>
</html>

```

Rys. 2.32. Przykład tworzenia mapy obrazu

Kontener grafiki SVG

Specyfikacja języka HTML5 pozwala na zastosowanie znaczników, które nie tylko wstawiają pliki formatu svg na stronę internetową, ale także umożliwiają dynamiczne rysowanie w oknie przeglądarki obrazu zdefiniowanego w kodzie za pomocą znacznika **<svg>**. Skalowalna Grafika Wektorowa SVG (ang. *Scalable Vector Graphics*) jest językiem na bazie XML, służącym do opisu dwuwymiarowej grafiki wektorowej.

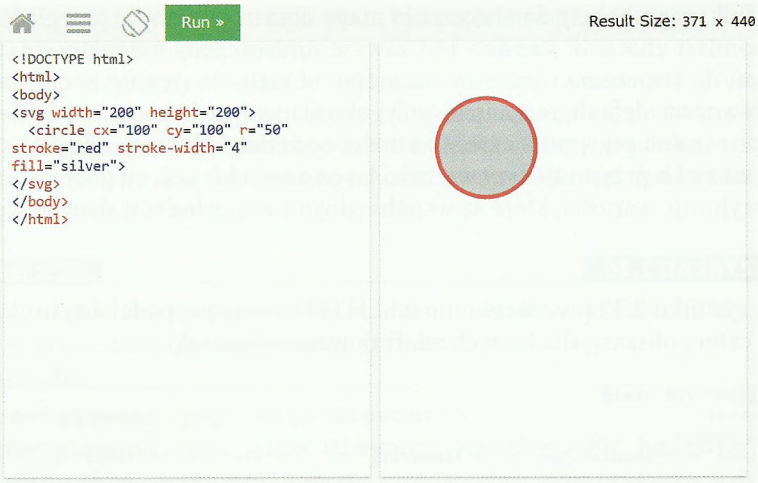
Do dyspozycji mamy różne rodzaje kształtów:

- **<rect>** – prostokąt;
- **<circle>** – okrąg;
- **<ellipse>** – elipsa;
- **<line>** – linia;
- **<polyline>** – wielokąt;
- **<polygon>** – wielobok;
- **<path>** – ścieżka.

Znacznik **<svg>** ma wiele atrybutów, które mogą być przypisane określonym kształtom.

PRZYKŁAD 2.27.

Na rysunku 6.3 pokazano przykład zastosowania znacznika **<svg>**. W oknie przeglądarki zdefiniowano kontener svg o wymiarach 200 na 200 px, w którym został narysowany okrąg o podanych parametrach cx, cy i r oraz nadanym kolorze wypełnienia i krawędzi ustalonej za pomocą atrybutu stroke. Wszystkie atrybuty znacznika **<svg>** możemy znaleźć na stronach W3C.



Rys. 2.33. Przykład zastosowania znacznika **<svg>**

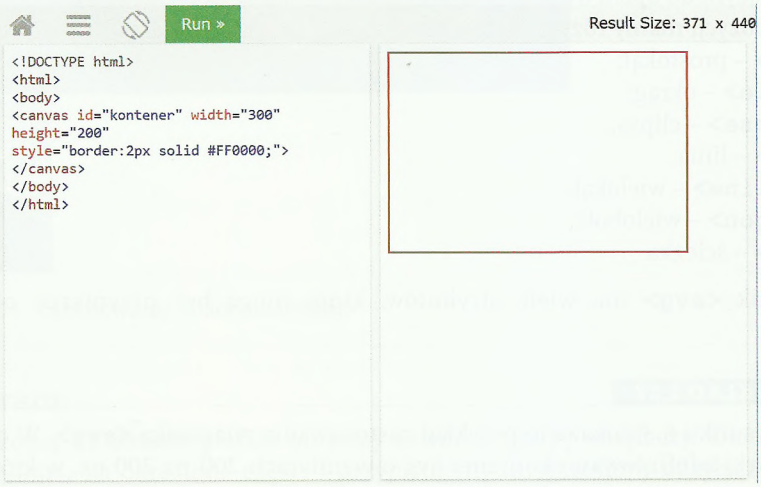
Element Canvas

Innym elementem umożliwiającym tworzenie grafiki bezpośrednio w oknie przeglądarki jest znacznik **<canvas>**. Znacznik definiuje kontener, który możemy wypełnić kolorem, tekstem, figurami geometrycznymi lub innym obrazem.

Działanie znacznika **<canvas>** jest oparte na dynamicznym, skryptowym renderowaniu kształtów i obrazów bitmapowych. Dlatego musimy pamiętać, że znacznik **<canvas>** definiuje tylko kontener do rysowania grafiki. Rysowaniem zajmuje się skrypt.

PRZYKŁAD 2.28.

Ponieważ językiem skryptowym zajmiemy się w dalszej części podręcznika, na rysunku 6.4 pokazano tylko definiowanie kontenera do tworzenia grafiki.



Rys. 2.34. Zastosowanie znacznika **<canvas>**

Multimedia na stronach internetowych

Multimedia reprezentują grupę plików zawierających dźwięk oraz wideo. Obsługują one najczęściej następujące formaty:

1. pliki dźwiękowe – wav, mid, midi, mp3, ogg;
2. pliki wideo i animacji – avi, mpeg, mp4, asf oraz swf.

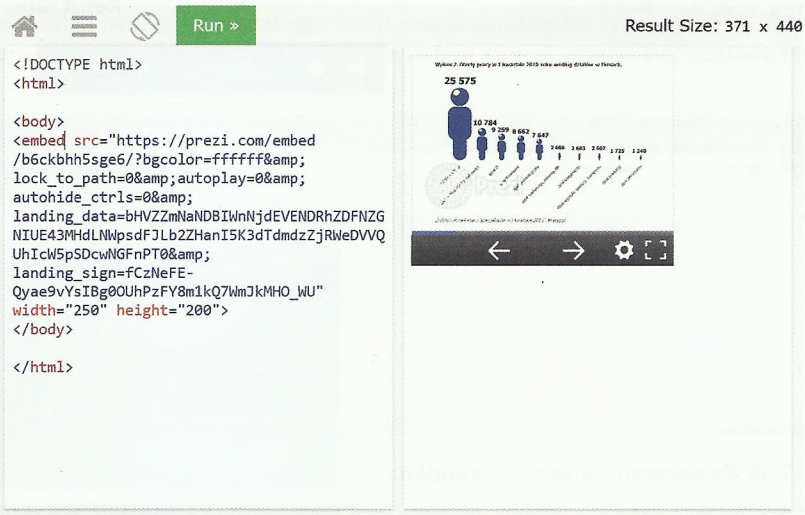
Niektóre z nich wymagają zainstalowania w przeglądarce specjalnego rozszerzenia zwanego plug-in lub wtyczką.

Do umieszczania na stronie internetowej plików multimedialnych służy znacznik **<embed>**. Niestety, znacznik ten nie jest obsługiwany przez wszystkie przeglądarki internetowe. Jedynym wymaganym przez znacznik **<embed>** atrybutem jest **src**, który definiuje ścieżkę dostępu do pliku multimedialnego. Dodatkowo możemy wykorzystać standardowe atrybuty określające obszar wyświetlania.

PRZYKŁAD 2.29.

```
<embed src="animacja.swf" width="100px" height="50px">
```

Na rysunku 2.35 przedstawiono zastosowanie znacznika **<embed>** z zastosowaniem sieciowego pliku multimedialnego do wyświetlenia na stronie.



Rys. 2.35. Zastosowanie znacznika **<embed>**

Jeżeli przeglądarka nie obsługuje znacznika **<embed>**, można zastosować znacznik **<object>**. Znacznik ten pozwala na osadzanie w dokumentach hipertekstowych plików audio, wideo, apletów Javy, animacji Flash a nawet dokumentów PDF. Przy zastosowaniu znacznika **<object>** należy pamiętać, że większość jego atrybutów nie jest obsługiwana przez HTM5. W najnowszej specyfikacji języka HTML stosowane są atrybuty:

- **data** – definiujący ścieżkę URL do pliku;
- **form** – definiujący jedną lub więcej form należących do obiektu;
- **name** – określający nazwę obiektu;
- **width** oraz **height** – określające wymiary okna wyświetlania obiektu.

PRZYKŁAD 2.30.

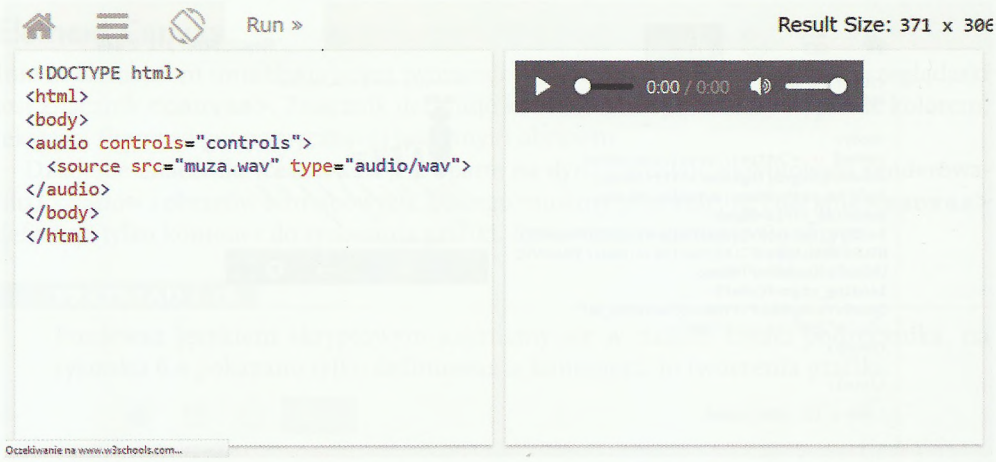
```
<object width="100px" height="50px" data="animacja.swf">
</object>
```

W języku HTML5 występują dodatkowe znaczniki, pozwalające na osadzanie w kodzie dokumentu hipertekstowego plików multimedialnych. Jednak nie wszystkie przeglądarki obsługują wszystkie formaty plików audio i wideo. Należy o tym pamiętać podczas projektowania witryny.

Do umieszczania plików audio służy znacznik **<audio>**. Może dotyczyć jednego źródła dźwięku lub kilku źródeł reprezentowanych poprzez użycie atrybutu **src** lub elementu **<source>**. Określona przeglądarka wybierze wówczas najbardziej odpowiedni dla niej zapis. Dzięki użyciu MediaStream znacznik może służyć również jako destynacja dla mediów strumieniowych.

PRZYKŁAD 2.31.

Na rysunku 2.36 przedstawiono przykład zastosowania znacznika **<audio>**. W oknie przeglądarki widać panel sterowania odtwarzaniem pliku dźwiękowego.



Rys. 2.36. Zastosowanie znacznika **<audio>**

Do umieszczania plików wideo służy znacznik **<video>**. Do standardowych atrybutów tego znacznika zaliczamy między innymi:

- **src** – definiujący ścieżkę URL do pliku;
- **poster** – definiujący ścieżkę do obrazka, który będzie wyświetlany w trakcie ładowania filmu;
- **autoplay** – automatyczne odtwarzanie podczas ładowania filmu;
- **controls** – wyświetlanie panelu sterowania;
- **loop** – pętlę odtwarzania filmu.

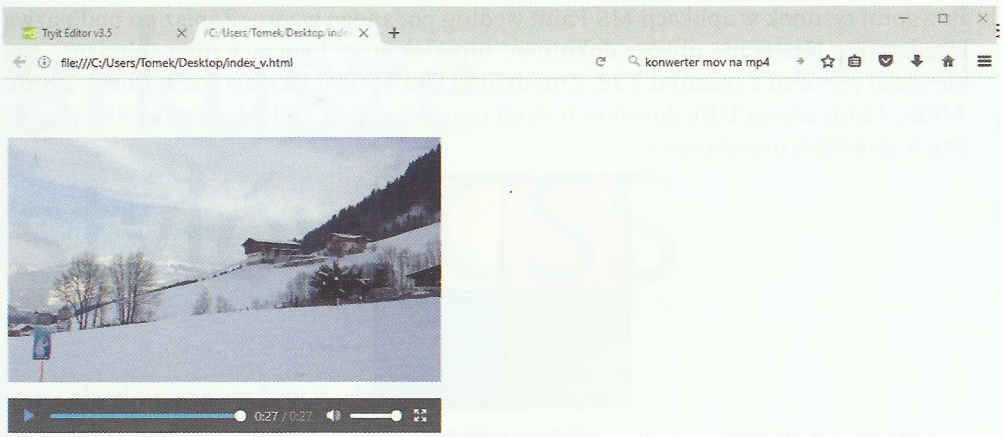
PRZYKŁAD 2.32.

Na rysunku 2.37 pokazano kod strony wykorzystującej publikację pliku wideo za pomocą znacznika **<video>**. Zastosowano tutaj definicję dwóch plików z różnymi rozszerzeniami, do wyboru przez przeglądarkę obsługiwanego rodzaju multimediów.

```
<!DOCTYPE html>
<html>
  <body>
 <video width=500" height="400" controls>
   <source src="animacja.mp4" type="video/mp4">
   <source src="animacja.mov" type="video/mov">
</video>
  </body>
</html>
```

Rys. 2.37. Zastosowanie znacznika **<video>** w kodzie HTML

Efektem załadowania omawianego kodu jest widoczny na rysunku 2.38 obszar z wyświetlanym plikiem multimedialnym oraz panelem sterowania wyświetlaniem.



Rys. 2.38. Efekt działania kodu ze znacznikiem **<video>**

Do odtwarzania filmów wideo przez przeglądarkę internetową często niezbędne są kodeki lub wtyczki. Pozwalają one na sprawne przeprowadzenie procesu kodowania lub dekodowania plików multimedialnych. Do popularnych kodeków zaliczają się: Real Player i Adobe Flash Player.

Zainstalowanie odpowiednich wtyczek w przeglądarkach klientów umożliwia poprawne odtwarzanie materiałów wideo z popularnych portali, takich jak np. YouTube. Wykorzystujemy wówczas znacznik **<iframe>**. W ścieżkę URL wystarczy wpisać identyfikator filmu (np. XGSy3_Czz8k) oraz atrybuty parametrów wyświetlania obrazu. Przykład zastosowania tego elementu pokazano na rysunku 2.39.

Run »

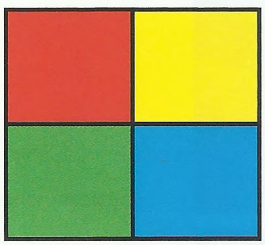
Result Size: 371 x 367

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

<iframe width="420" height="345"
src="https://www.youtube.com/embed
/O5kHei3MHzM">
</iframe>

</body>
</html>
```

Rys. 2.39. Przykład osadzania filmu w dokumencie hipertekstowym

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przygotuj rysunek w aplikacji MS Paint według podanego wzoru. Zapisz go pod nazwą kwadrat.jpg. Następnie utwórz dokument hipertekstowy tworzący mapę obrazu – wykorzystaj przykład z rysunku 2.32. Zmodyfikuj kod – podaj zamiast nazw 1.htm, 2.htm, 3.htm, 4.htm adresy URL dowolnych stron internetowych. Efekt działania pliku przetestuj w dowolnej przeglądarce.



2. Skorzystaj z dowolnej metody osadzania plików multimedialnych i utwórz dokument hipertekstowy publikujący wybrany film z portalu internetowego YouTube. Efekt swojej pracy zapisz pod nazwą wideo.html. Otwórz swój plik w trzech przeglądarkach: Opera, IE, Firefox. Zwróć uwagę na poprawność i sposób wyświetlania filmu. Wnioski z wykonanego zadania omów na forum klasy z nauczycielem.
3. Sprawdź, które z wymienionych niżej formatów plików dźwiękowych są obsługiwane w przeglądarce internetowej Internet Explorer. Informacji szukaj w dokumentacji przeglądarki i specyfikacji języka HTML5.
   Formaty plików dźwiękowych: mp3, wav, ogg.
4. Do czego służy znacznik **<object>**?
5. Zmodyfikuj kod źródłowy z rysunku 2.37 w taki sposób, aby w trakcie ładowania filmu wyświetlał się obraz o nazwie rys.jpg, który znajduje się w katalogu głównym strony internetowej.

3. Kaskadowe arkusze stylów CSS

- Kaskadowość stylów w dokumentach HTML
- Selektory, właściwości oraz wartości w CSS
- Model blokowy w CSS
- Pozycjonowanie elementów w CSS
- Praktyczne zastosowania CSS i HTML

3.1 Kaskadowość stylów w dokumentach HTML

ZAGADNIENIA

- **Definicja CSS**
- **Co to jest kaskadowość stylów?**
- **Struktura arkusza stylów**

Wraz z rozwojem budowy stron internetowych pojawiła się potrzeba rozdzielenia części kodu obejmującej strukturę i treść strony oraz części odpowiedzialnej za prezentację tej treści. Praktycznie, wraz z powstaniem specyfikacji HTML4 zaczęto rozdzielać kodowanie treści i jej prezentację. Rolę prezentacji treści przejęły kaskadowe arkusze stylów **CSS** (ang. *Cascading Style Sheets*).

Struktura arkusza stylów

Style CSS składają się z listy reguł (dyrektyw) określających, w jaki sposób zostanie wyświetlony wybrany element dokumentu HTML. Definiowanie stylów CSS podlega zasadom określonym w specyfikacji W3C. Styl składa się z **reguł** (sekcji). Każda reguła jest złożona z **selektora** i **deklaracji**. Deklaracja składa się z **właściwości**, którym przypisujemy określone **wartości**.

PRZYKŁAD 3.1.

Poniżej przedstawiono składnię pojedynczej reguły kaskadowego arkusza stylów.

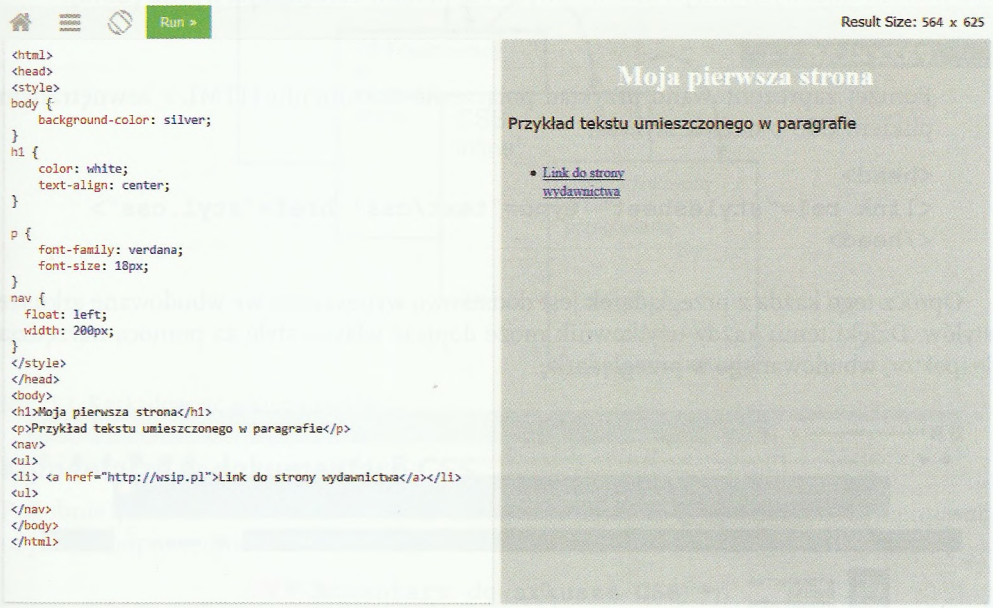
```
selektor {
właściwość1: wartość1;
właściwość2: wartość2;
właściwość3: wartość3;
}
```

Selektorem może być wybrany element strony lub grupa elementów, które chcemy sformatować. Wewnątrz nawiasów klamrowych deklarujemy cechy danego elementu w postaci elementarnych właściwości. Każdej właściwości przyporządkowujemy wartość – wpisujemy ją po dwukropku. Zapis każdej deklaracji właściwości kończymy średnikiem.

PRZYKŁAD 3.2.

Na przykładzie widać styl wstawiony w nagłówku strony internetowej. Definiuje on cztery elementy strony: body, h1, p i nav odnoszące się do znaczników języka HTML o tych samych nazwach. Oznacza to, że jeżeli w obrębie strony użyjemy znacznika `<h1>`, to formatowanie tego elementu będzie pobierane ze stylu CSS, a dokładniej – z deklaracji właściwości selektora h1. Efektem działania tego

selektora będzie wyświetlenie tekstu w nagłówku pierwszego stopnia, w kolorze białym, z wyrównaniem do środka strony.



Rys. 3.1. Przykład zastosowania CSS

Do najważniejszych zalet kaskadowych arkuszy stylów zaliczamy:

- rozdzielenie kodu treści dokumentu HTML od kodu jej prezentacji w oknie przeglądarki;
- możliwość formatowania wielu dokumentów HTML jednakowo;
- łatwość modyfikacji struktury strony.

Metody dołączania stylów do dokumentu HTML

Istnieją trzy metody dołączania kaskadowych arkuszy stylów do dokumentów HTML.

1. **Metoda inline** (styl wpisany, lokalny). W tej metodzie stosujemy atrybut style dla wybranego znacznika języka HTML.

PRZYKŁAD 3.3.

```
<p style=" color: blue; font-size: 12px"> Przykładowy tekst
</p>
```

2. **Metoda osadzenia w nagłówku** `<head>` (styl wewnętrzny). Taki arkusz stylów definiujemy w nagłówku dokumentu HTML, wewnątrz znaczników `<style>` oraz `</style>`. Przykład wykorzystania takiej metody pokazano na rysunku 3.1.
3. **Metoda łączenia z plikiem zewnętrznym** (styl zewnętrzny). Metoda ta polega na utworzeniu oddzielnego pliku, który zawiera listę reguł. Plik zapisujemy z rozszerzeniem `.css`, a następnie definiujemy połączenie z nim za pomocą znacznika `<link>` w nagłówku dokumentu HTML.

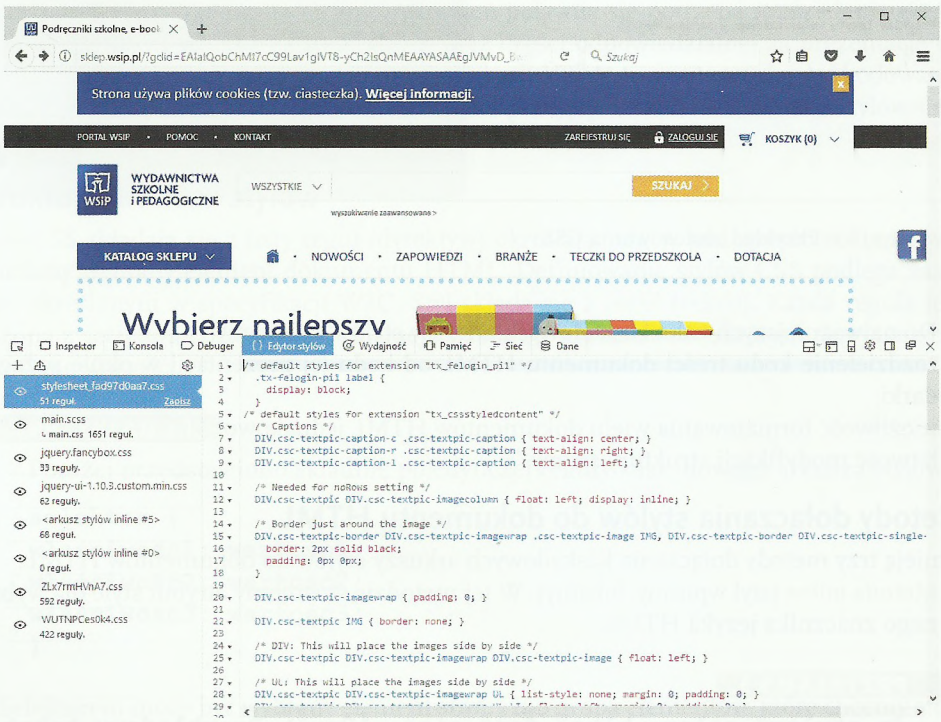
Ta metoda pozwala na definiowanie arkuszy stylów dla wielu dokumentów HTML i jest obecnie najpopularniejszym i najbardziej efektywnym sposobem zbiorczego formatowania wielostronicowych witryn internetowych i systemów zarządzania treścią (CMS).

PRZYKŁAD 3.4.

Poniżej zaprezentowano przykład połączenia dokumentu HTML z zewnętrznym plikiem CSS o nazwie styl.css.

```
<head>
<link rel="stylesheet" type="text/css" href="styl.css">
</head>
```

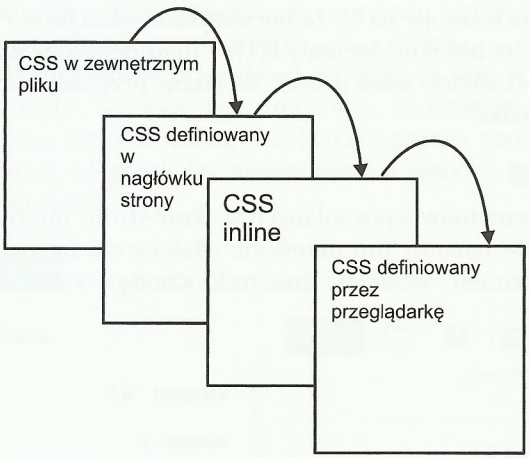
Oprócz tego każda z przeglądarek jest dodatkowo wyposażona we wbudowane arkusze stylów. Dzięki temu każdy użytkownik może dopisać własne style za pomocą narzędzia **inspektor**, wbudowanego w przeglądarkę.



Rys. 3.2. Widok narzędzia inspektor w przeglądarce Firefox

Kaskadowość arkuszy stylów

Kaskadowość arkuszy stylów określa kolejność oddziaływania stylu w zależności od metody dołączenia jego kodu do formatowanego dokumentu HTML. Dlatego style o wyższym priorytecie mają pierwszeństwo w modyfikowaniu elementów dokumentu HTML. Style o niższym priorytecie mogą zmienić formatowanie jakiegoś elementu, tylko jeżeli style o wyższym priorytecie nie definiują formatowania tego elementu. Dzięki temu możemy pracować z różnymi metodami tworzenia arkuszy stylów bez obawy, że będzie to powodować błędy formatowania elementów na stronie.



Rys. 3.3. Kaskadowość arkuszy stylów

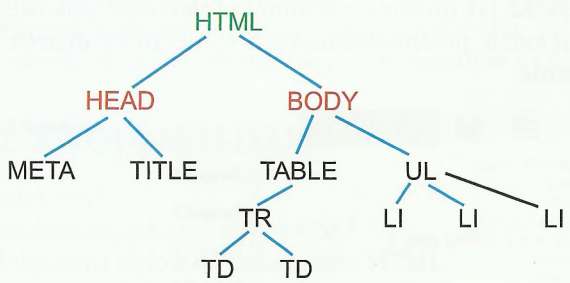
Komentarze w dokumentach CSS

Podobnie jak w dokumentach HTML, kaskadowe arkusze stylów można komentować, używając odpowiedniej składni kodu:

```
/* komentarz do arkusza CSS */
```

Dziedziczenie

Struktura każdego dokumentu hipertekstowego opiera się na hierarchicznej budowie **drzewa dokumentu**. Można to porównać do zależności występujących w drzewie genealogicznym typowej rodziny. Każdy bowiem element w drzewie dokumentu ma dokładnie **jednego rodzica**, oprócz elementu głównego, czyli korzenia (ang. *root*) drzewa, którym jest zawsze element **<HTML>** (rys. 3.4).



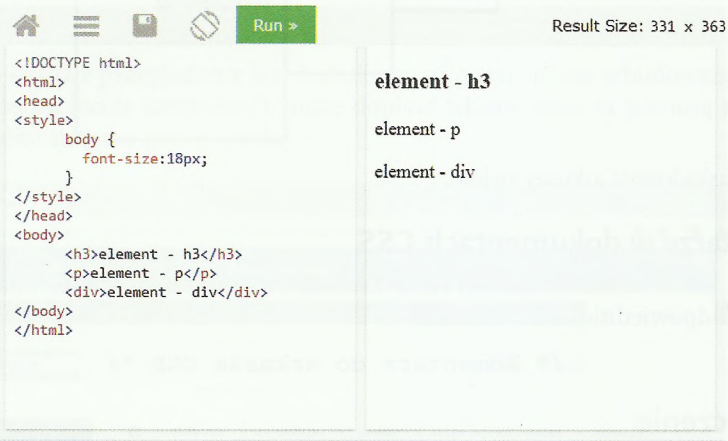
Rys. 3.4. Drzewo dokumentu HTML

Jak przedstawiono to na rys. 3.4, mamy w takiej strukturze przodków (rodziców, dziadków), braci i potomków (dzieci). Pozycja w danej hierarchii decyduje również o **dziedziczeniu** pewnych cech (np. wyglądu) elementów znajdujących się wyżej oraz o przekazywaniu cech elementom znajdującym się niżej w hierarchii. Hierarchię ustalamy zawsze na podstawie miejsca, w którym znajduje się dany element dokumentu. Ponieważ za prezentację dokumentu hipertekstowego odpowiadają kaskadowe arkusze stylów, elementy HTML mogą dziedziczyć również właściwości CSS przodka lub przekazywać je swoim potomkom.

Należy jednak zwrócić uwagę na to, że nie wszystkie właściwości CSS są dziedziczone automatycznie. Niektóre bowiem elementy HTML mają domyślne właściwości (np. **<h4>**) lub nie obsługują wszystkich właściwości. W takim przypadku nie będą przyjmować formatowania od przodka.

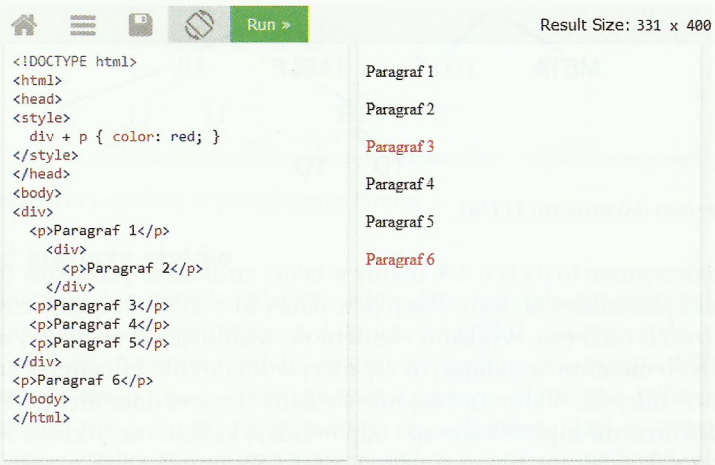
PRZYKŁAD 3.5.

Ponieważ dla elementów wyświetlanych w oknie strony internetowej rodzicem jest element **<body>**, nadamy mu określone właściwości formatowania i zobaczymy czy wszystkie elementy wewnątrz znacznika **<body>** odziedziczą te same cechy.



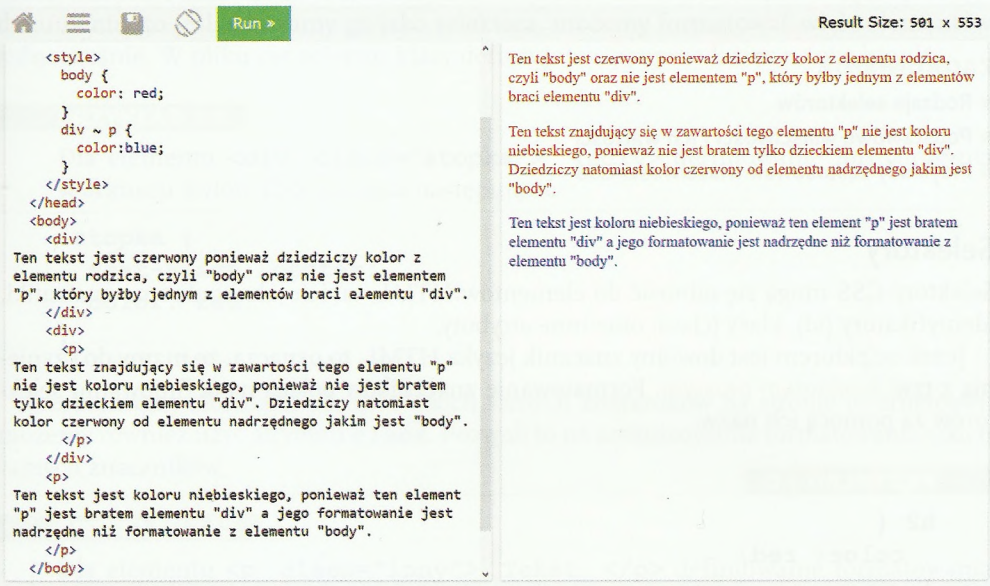
Rys. 3.5. Przykład dziedziczenia stylów CSS

Widać, że element **<h3>** nie odziedziczył formatu z rodzica (elementu **<body>**), ponieważ ma swój nadrzędny styl formatowania.
Dziedzicznie można łatwo zaobserwować w strukturze blokowej. Na rysunku 3.6 pokazano przykład, jak można programować elementy blokowe, opierając się na tej przydatniej metodzie przejmowania cech w hierarchii drzewa dokumentu, jakim jest dziedziczenie.



Rys. 3.6. Przykład zastosowania dziedziczenia stylów CSS przez elementy blokowe

Tego typu selektorów często używamy w tworzeniu stylów opartych na zależnościach związanych z dziedziczeniem i hierarchią drzewa.
Innym przykładem selektora braci (braci – ponieważ leżą w tym samym rzędzie hierarchii) jest tylda. Selektor ten umożliwia nadanie określonych atrybutów wszystkim braciom, których nazwa w deklaracji została podana jako druga. Zwykły tekst znajdujący się pomiędzy braćmi nie ma wpływu na działanie selektora, tzn. jest ignorowany (rys. 3.7).



Rys. 3.7. Przykład zastosowania selektora "~"

Przy tworzeniu tego stylu użyto selektora „+". Spowodował on, że formatowane będą tylko elementy blokowe znajdujące się bezpośrednio po elemencie ** **. Dlatego kolor czerwony został ograniczony do paragrafów 3 i 6.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Do czego służą selektory?
2. Jaką rolę pełnią kaskadowe arkusze stylów CSS?
3. Omów metody dołączania stylów do dokumentu HTML.
4. W jaki sposób komentuje się kaskadowe arkusze stylów?

3.2 Selektory, właściwości oraz wartości w CSS

ZAGADNIENIA

- **Rodzaje selektorów**
- **Podstawowe właściwości i ich wartości**
- **Przykłady zastosowań właściwości CSS**

Selektory

Selektory CSS mogą się odnosić do elementów HTML, takich jak: konkretne znaczniki, identyfikatory (id), klasy (class) oraz inne atrybuty.

Jeżeli selektorem jest dowolny znacznik języka HTML, to oznacza, że mamy do czynienia z tzw. **selektorem prostym**. Formatowanie znaczników polega na definiowaniu selektorów za pomocą ich nazw.

PRZYKŁAD 3.6.

```
h2 {
  color: red;
}
```

Możemy również grupować selektory znaczników, które mają być identycznie formatowane.

PRZYKŁAD 3.7.

```
h1, p, i {
  color: red;
}
```

Do identyfikacji poszczególnych elementów używamy w języku HTML identyfikatorów, zapisywanych skrótem id. Szczególnie często stosujemy je w definiowaniu elementów blokowych `<div>`. Każdy identyfikator ma unikatową nazwę w obszarze jednego dokumentu. Nazwy tej można użyć jako selektora do formatowania elementu, który ona identyfikuje. Należy pamiętać, że zapis selektora identyfikatora jest poprzedzony zawsze znakiem #.

PRZYKŁAD 3.8.

Dla elementu `<div id="stopka"> </div>` definiowanie formatowania w arkuszu stylów CSS wygląda następująco:

```
#stopka {
```

```
    text-align: center;
    color: green;
}
```

Inną metodą definiowania selektora jest użycie klasy, która również służy do rozróżniania elementów języka HTML. Klasy definiujemy za pomocą atrybutu **class**. Selektor klasy odszukuje każdy element w dokumencie HTML, który ma ten atrybut o takiej samej nazwie. Ponieważ atrybutu klasy możemy używać wielokrotnie w obrębie jednego dokumentu, to jeśli używamy go jako selektora, możemy formatować wiele elementów jednocześnie. W pliku css selektor klasy definiujemy, poprzedzając nazwę kropką.

PRZYKŁAD 3.9.

Dla elementu `<div class="stopka"> </div>` definiowanie formatowania w arkuszu stylów CSS wygląda następująco:

```
.stopka {
    text-align: center;
    color: red;
}
```

Jeżeli chcemy sformatować część takich samych znaczników na stronie internetowej, możemy również użyć atrybutu **class**. Pozwoli to na zróżnicowanie formatowania takich samych znaczników.

PRZYKŁAD 3.10.

Dla elementu `<p class="inny"> Tekst </p>` definiowanie formatowania w arkuszu stylów CSS wygląda następująco:

```
p.inny {
    color: red;
}
```

W CSS stosujemy również **selektory uniwersalne**, oznaczone znakiem *. Mają one taką samą ważność jak definiowany styl elementu najwyższego poziomu, czyli **body**.

Hierarchiczny układ elementów HTML pozwala również na stosowanie selektorów zależnych od elementów umieszczonych wyżej w hierarchii, na tym samym poziomie lub niżej. Mówimy wtedy o **selektorach potomka** (ang. *descendant selector*), **selektorach dziecka** (ang. *child selector*) lub **selektorach braci** (ang. *adjacent sibling selector*).

PRZYKŁAD 3.11.

```
ul ul li {color:red; font-size:20px;}
```

Tego typu zapis wykorzystuje selektor potomka. Oznacza to, że elementy wykazu na drugim poziomie zagnieżdżenia będą zapisane kolorem czerwonym oraz czcionką o wielkości 20 px, podczas gdy elementy wykazu pierwszego poziomu będą miały domyślny kolor i rozmiar czcionki.

Właściwości

Podczas doboru właściwości CSS pierwszą rzeczą, na jaką powinniśmy zwrócić uwagę, jest to, czy dana właściwość jest poprawnie interpretowana przez przeglądarki, na których będzie wyświetlana nasza strona. Może się zdarzyć, że właściwości ze specyfikacji CSS3 nie będą poprawnie interpretowane przez niektóre przeglądarki. Jest to skutkiem ciągłego rozwijania tej wersji specyfikacji.

Wszystkie właściwości CSS możemy podzielić na kilka kategorii. Każda z kategorii jest odpowiedzialna za zmianę wyglądu określonego elementu HTML. Możemy wyróżnić następujące kategorie właściwości CSS:

- tekst – właściwości odpowiedzialne za formatowanie cech wyglądu tekstu;
- czcionka – właściwości odpowiedzialne za formatowanie wyglądu czcionki tekstu;
- skalowanie – właściwości odpowiedzialne za formatowanie szerokości i wysokości elementu;
- tło – właściwości odpowiedzialne za formatowanie cech wyglądu tła w danym elemencie;
- obramowanie – właściwości odpowiedzialne za formatowanie cech wyglądu obramowania danego elementu;
- wykazy – właściwości formatujące listy;
- tabela – właściwości formatujące cechy komórek tabeli.

Ze względu na dużą liczbę możliwych do wykorzystania właściwości opiszemy tylko niektóre z nich.

Tekst

Do często używanych właściwości formatujących tekst zaliczamy:

- **`color`** – pozwala na określenie koloru danego elementu;
- **`text-align`** – definiuje wyrównanie tekstu;
- **`text-decoration`** – pozwala na dekorację (wyróżnienie) fragmentów tekstu;
- **`direction`** – określa kierunek wyświetlania tekstu;
- **`text-overflow`** – określa sposób wyświetlenia skróconego tekstu.

PRZYKŁAD 3.12.

Run »

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    color: blue;
}

h2 {
    color: green;
}
a {
    text-decoration: overline;
}
</style>
</head>
<body>
<h2>Sformatowany nagłówek drugiego stopnia</h2>
<p>Tekst w paragrafie sformatowany selektorem body</p>
<p style="direction:rtl">Tekst akapitu idący od prawego do
lewego marginesu</p>
<p>Przykład wyróżnienia tekstu <a
href="https://www.wsip.pl">WSiP</a></p>
</body>
</html>
```

Result Size: 518 x

Sformatowany nagłówek drugiego stopnia

Tekst w paragrafie sformatowany selektorem body

Tekst akapitu idący od prawego do lewego margin

Przykład wyróżnienia tekstu WSiP

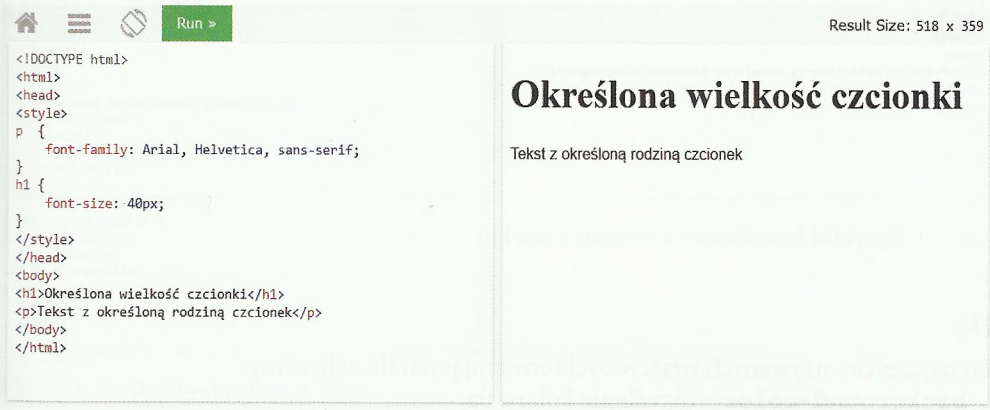
Rys. 3.8. Przykład wykorzystania właściwości CSS odnoszących się do tekstu

Czcionka

Do najczęściej używanych właściwości formatujących tekst zaliczamy:

- **font-style** – określenie stylu czcionki;
- **font-family** – określenie rodziny czcionek;
- **font-size** – zdefiniowanie wielkości czcionek.

PRZYKŁAD 3.13.



Rys. 3.9. Właściwości formatujące czcionki

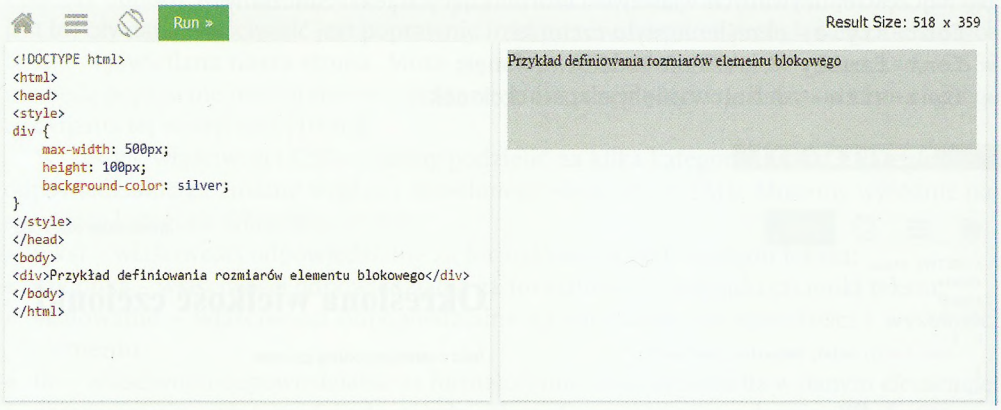
Ze względu na dużą liczbę czcionek dostępnych na rynku oraz brak możliwości sprawdzenia, czy u danego użytkownika zainstalowana jest w systemie dana czcionka, możemy definiować w CSS grupę czcionek (font-family), które będą wykorzystywane w zależności od możliwości przeglądarki klienta.

Formatowanie rozmiaru

Do tej grupy zaliczamy właściwości określające szerokość i wysokość danego elementu. Zaliczamy do nich:

- **width** – określenie szerokości elementu;
- **height** – określenie wysokości elementu;
- **min-width** – określenie minimalnej szerokości;
- **max-width** – określenie maksymalnej szerokości;
- **min-height** – określenie minimalnej wysokości;
- **max-height** – określenie maksymalnej wysokości.

PRZYKŁAD 3.14.



Rys. 3.10. Przykład formatowania rozmiaru obiektu

Tło

Do najczęściej używanych właściwości formatujących tło zaliczamy:

- **background-color** – określenie koloru tła;
- **background-image** – wstawienie obrazu strony jako tło;
- **background-position** – ustawienie grafiki tła w określonej pozycji.

PRZYKŁAD 3.15.

```
body {
  background-image: url("/grafika/tlo.jpg");
}
```

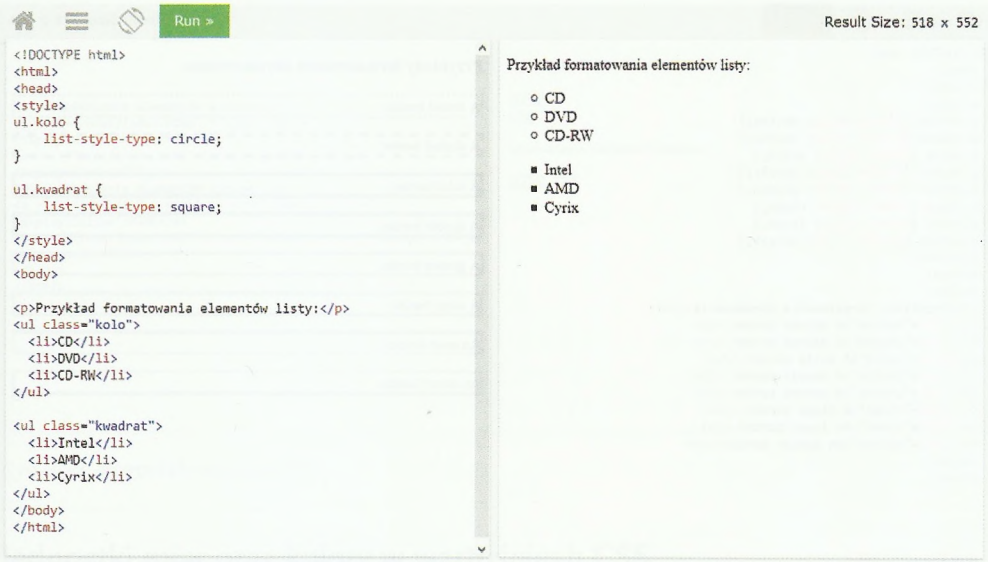
Zastosowanie takiego stylu spowoduje umieszczenie obrazka z pliku tlo.jpg jako tła dla całej strony.

Wykazy (listy)

Do najczęściej używanych właściwości formatujących listy zaliczamy:

- **list-style-type** – określenie graficznych punktorów listy;
- **list-style-image** – zdefiniowanie pliku graficznego jako punktora.

PRZYKŁAD 3.16.



Rys. 3.11. Formatowanie listy w CSS

Obramowanie

Do najczęściej używanych właściwości formatujących obramowanie elementów zaliczamy:

- **border-style** – określa styl obramowania;
- **border-color** – określa kolor obramowania;
- **border-radius** – definiuje zakres zaokrąglenia krawędzi obramowania;
- **border-image** – definiuje graficzny styl obramowania.

PRZYKŁAD 3.17.

Run »

Result Size: 518 x 552

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
</style>
</head>
<body>
<h3>Przykłady formatowania obramowania:</h3>
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
</body>
</html>
```

Przykłady formatowania obramowania:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

Rys. 3.12. Przykłady obramowania w CSS

Tabele

Do najczęściej używanych właściwości formatujących tabele zaliczamy:

- **caption-side** – określa położenie tytułu tabeli;
- **border-spacing** – definiuje odstępy między komórkami.

Pozostałe właściwości

Oprócz wymienionych możemy spotkać właściwości ułatwiające wyświetlanie elementu, formatowanie wypływania treści kontenera, oblewanie tekstem elementu czy właściwości definiujące rodzaju kursora.

Właściwość display reguluje sposób wyświetlania elementów – tworzy z nich, na żądanie autora strony, elementy blokowe, liniowe czy punkty wykazu.

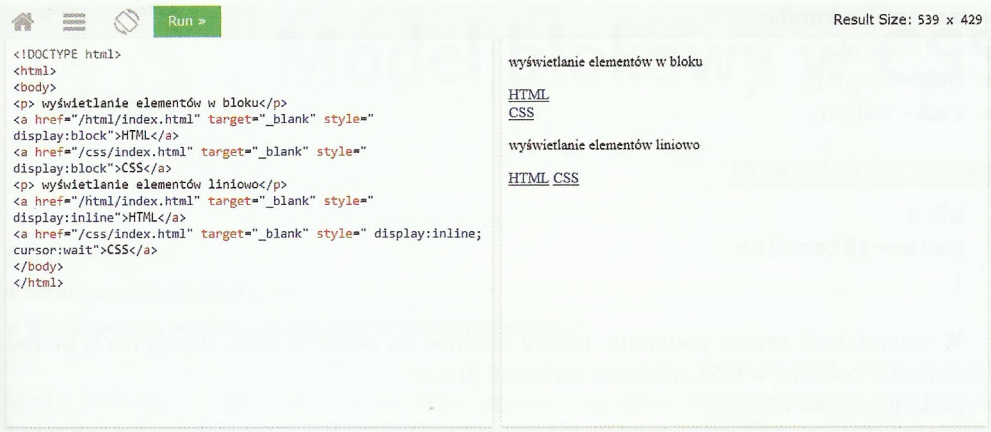
Właściwość overflow pozwala zarządzać przepełnieniem treści bloku czy kontenera.

Float umożliwia zdefiniowanie sposobu otaczania elementów tekstem. Natomiast właściwość cursor określa typ kursora myszki, który ma obowiązywać na danym obszarze.

UWAGA

Zestawienie najpopularniejszych właściwości i ich wartości specyfikacji CSS3 zamieszczono jako załączniki, na końcu podręcznika.

PRZYKŁAD 3.18.



Rys. 3.13. Pozostałe właściwości

Jednostki miar oraz kolory w wartościach CSS

Kaskadowe arkusze stylów określają cechy wielu elementów. Stosowane są przy tym różne jednostki miar. Możemy podzielić je na jednostki względne i bezwzględne.

Do jednostek miar względnych zaliczamy:

- **px** – piksele;
- **em** – proporcje elementu względem wielkości czcionki;
- **ex** – proporcję względem wysokości x danej czcionki;
- **%** – procenty.

PRZYKŁAD 3.19.

```
p{
font-size:10%;
margin-left:20px;
}
```

Jednostki miar bezwzględnych to:

- **in** – cal;
- **pt** – punkt (jednostka wywodząca się z typografii; 72 pt = 1 in);
- **cm** – centymetr;
- **mm** – milimetr;
- **pc** – pica (jednostka wywodząca się z typografii; 1 pc = 12 pt).

PRZYKŁAD 3.20.

```
p{
font-size:12pt;
margin-left:2in;
}
```

Dodatkowo w tworzeniu kaskadowych arkuszy stylów używa się jednostek miary kątów oraz jednostek czasu:

- **ms** – milisekunda;
- **s** – sekunda;
- **deg** – stopień;
- **rad** – radian.

PRZYKŁAD 3.21.

```
h5 {
pause-after:10s;
}
```

W wartościach często podajemy nazwy kolorów za pomocą kilku dostępnych metod. Kodowanie kolorów w CSS możemy wykonać przez:

- podanie nazwy koloru,

```
h3 {color:red;}
```

- zapis za pomocą modelu RGB z wykorzystaniem wartości liczbowych,

```
#stopka {color:rgb(255,0,0);}
```

- zapis za pomocą modelu RGB z wykorzystaniem wartości procentowych,

```
p {color:rgb(100%,0%,0%);}
```

- zapis za pomocą modelu RGBA,

```
#menu {background-color: rgba(255, 0, 0, 0.3);}
```

- zapis za pomocą modelu HLS,

```
.menu {background-color: hsl(120, 100%, 50%);}
```

- zapis za pomocą modelu HSLA,

```
p.stopka {background-color: hsla(120, 100%, 25%, 0.3);}
```

- zapis korzystający z systemu szesnastkowego,

```
p {color:#FF0000;}
```

Dodatkowo możemy określić stopień przeźroczystości właściwością opacity – stosujemy wartości od 0.0 (pełna przeźroczystość) do 1.0 (brak przeźroczystości).

```
p {background-color:rgb(255,0,0);opacity:0.7;}
```

SPRAWDŹ SWOJĄ WIEDZĘ

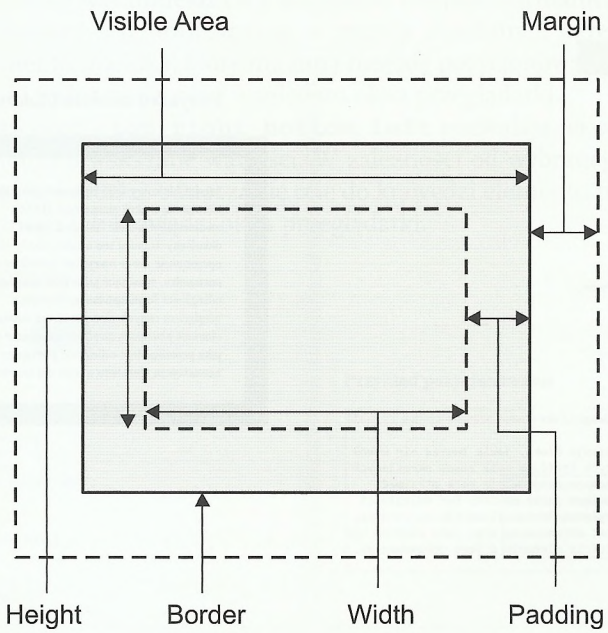
1. Scharakteryzuj poszczególne rodzaje selektorów.
2. Wymień najczęściej używane właściwości CSS.
3. W jaki sposób można przeprowadzić kodowanie kolorów w CSS?

3.3 Model blokowy w CSS

ZAGADNIENIA

- **Struktura układu blokowego**
- **Właściwości i wartości stosowane w modelu blokowym**

Model blokowy w CSS opisuje bloki, które są tworzone dla elementów HTML. Model blokowy dokładnie definiuje ustawienie marginesów zewnętrznego i wewnętrznego oraz obramowania, a także zawartość. Strukturę układu blokowego elementów HTML pokazuje rysunek 3.14.



Rys. 3.14. Przykład modelu blokowego

Aby wytłumaczyć zasadę działania modelu blokowego w kaskadowych arkuszach stylów, posłużymy się przykładem formatowania prostego dokumentu hipertekstowego, zawierającego jeden nagłówek drugiego stopnia i jeden blok `<div>`.

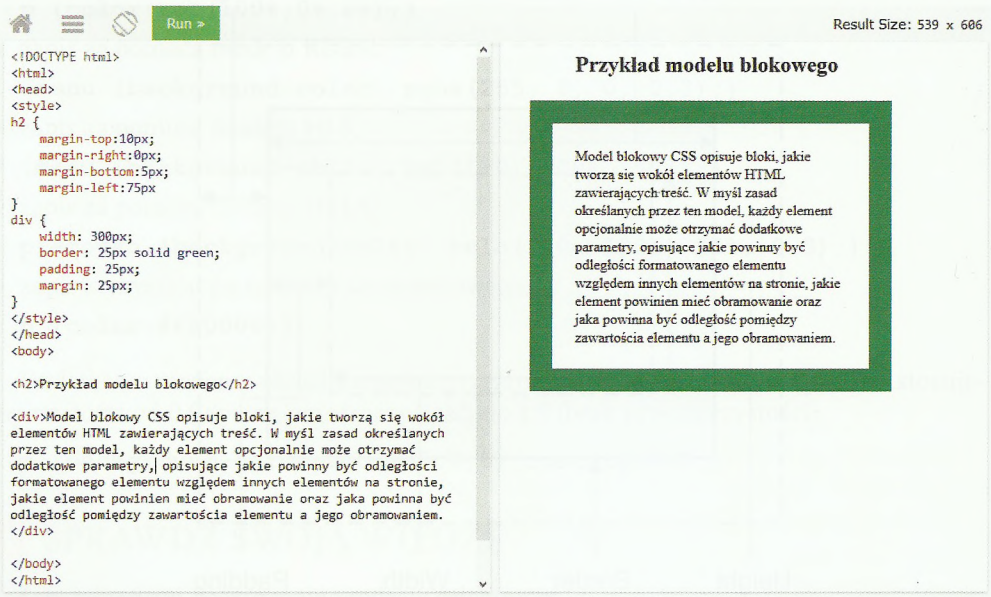
```
<!DOCTYPE html>
<html>

<body>

<h2>Przykład modelu blokowego</h2>

<div>Model blokowy CSS opisuje bloki, jakie tworzą się wokół
elementów HTML zawierających treść. W myśl zasad określanych przez
ten model, każdy element opcjonalnie może otrzymać dodatkowe
parametry, opisujące jakie powinny być odległości formatowanego
elementu względem innych elementów na stronie, jakie element powinien
mieć obramowanie oraz jaka powinna być odległość pomiędzy zawartościa
elementu a jego obramowaniem.</div>
</body>
</html>
```

Rys. 3.15. Dokument HTML przed formatowaniem stylem CSS

Mamy tutaj dwa elementy języka HTML, które sformatujemy z wykorzystaniem modelu blokowego. Dla nagłówka zdefiniujemy wartości marginesów zewnętrznych **margin**, które określą jego położenie względem kontenera strony.

Dla bloku ** ** zdefiniujemy margines zewnętrzny **margin** i wewnętrzny **padding** oraz określimy szerokość i kolor obramowania **border** oraz szerokość bloku **width**.

Po zapisaniu definicji stylu wewnętrznego w sekcji **<head>** otrzymamy efekt widoczny na rysunku 3.16.



Rys. 3.16. Dokument po formatowaniu według modelu blokowego

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jaką funkcję pełni model blokowy w CSS?
2. Czym charakteryzuje się dokument HTML po sformatowaniu przy użyciu stylu CSS?

3.4 Pozycjonowanie elementów w CSS

ZAGADNIENIA

- **Rola pozycjonowania elementów**
- **Właściwości i wartości dotyczące pozycjonowania**

Właściwości pozycjonowania pozwalają na precyzyjne określenie miejsca, w którym dany element ma być wyświetlany na stronie internetowej. Elementy mogą być rozmieszczane na kilka sposobów:

- **`static`** – element jest umieszczany tak, jak wynika to z układu w dokumencie hipertekstowym;
- **`relative`** – element jest umieszczany względem swojego normalnego położenia;
- **`absolute`** – element jest umieszczany w pozycji absolutnej, określanej względem pierwszego elementu przodka, który ma inną metodę pozycjonowania niż static;
- **`fixed`** – element jest umieszczany względem okna przeglądarki.

Dodatkowo właściwości **`top`**, **`right`**, **`bottom`**, **`left`** pozwalają na określenie przesunięcia bloku elementu o wskazane wartości. W zależności od wybranej metody pozycjonowania oraz rodzaju wartości, odnoszą się one do krawędzi elementu pozycjonowanego, elementu nadrzędnego lub krawędzi okna przeglądarki.

PRZYKŁAD 3.22.

Run »

Result Size: 539 x 606

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #6666FF;
}
</style>
</head>
<body>
<h2>Przykład pozycjonowania</h2>
<div class="relative">Element z pozycjonowaniem normalnym
<div class="absolute">Element z pozycjonowaniem absolutnym
względem poprzedniego elementu</div>
</div>
</body>
</html>
```

Przykład pozycjonowania

Element z pozycjonowaniem normalnym

Element z pozycjonowaniem absolutnym względem poprzedniego elementu

Rys. 3.17. Przykład pozycjonowania elementów w CSS

Korzystanie z pozycjonowania i przesunięć elementów względem układu normalnego może skutkować koniecznością określenia sposobu lub kolejności nakładania się jednego elementu na drugi. Właściwością, która pozwoli na wymuszenie kolejności nakładania się elementów, jest **z-index**.

PRZYKŁAD 3.23.

Tworzymy dokument hipertekstowy z wewnętrznym arkuszem stylów formatującym tekst i grafikę. Gotowy listing jest pokazany na rysunku 3.18.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
p {
    color:blue;
}
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
}
</style>
</head>
<body>

<h1>Przykład nagłówka z pozycjonowaniem grafiki</h1>
<img src="logo.jpg" width="150" height="100">
<p>Zastosowanie właściwości z-index z wartością -1 spowoduje, że
grafika osadzona zostanie za tekstem.</p>

</body>
</html>
```

Rys. 3.18. Dokument HTML z przykładem pozycjonowania

W stylach zastosowano właściwość **z-index** z wartością „-1". Pozwoli to na umieszczenie rysunku pod tekstem umieszczonym w nagłówku i akapicie. Efekt jest widoczny na rysunku 3.19.

Przykład nagłówka z pozycjonowaniem grafiki

Zastosowanie właściwości z-index z wartością -1 spowoduje, że grafika osadzona zostanie za tekstem.

Rys. 3.19. Przykład zastosowania właściwości **z-index**

3.5 Praktyczne zastosowania CSS i HTML

ZAGADNIENIA

- **Przykład formatowania formularza**
- **Przykład podziału strony na bloki**

Znamy już podstawowe znaczniki języka HTML oraz zasady wykorzystania kaskadowych arkuszy stylów do formatowania dokumentów hipertekstowych. W tym rozdziale opiszemy kilka przykładów praktycznego wykorzystania tych umiejętności.

Przykład formatowania formularzy

Naszym zadaniem jest przygotowanie formularza, który będzie wysyłać na adres e-mail wynik prostej ankiety dla uczniów. W tytule e-maila musi pojawić się tekst „Temat ankiety". Do wysłania e-maila należy zastosować najbezpieczniejszą z dostępnych metod. W ankiecie uczeń powinien podać imię i nazwisko oraz wybrać z listy jeden z czterech dostępnych tematów kursów: HTML, CSS, CMS lub SQL. Dodatkowo dwa pierwsze pola mają wymusić podanie danych. W tytule formularza należy wstawić tekst „Ankieta dotycząca wyboru tematu kursu" w formacie nagłówka pierwszego stopnia. Pod formularzem należy zdefiniować dwa przyciski, jeden do wysłania danych – o nazwie Submit, a drugi do czyszczenia zawartości pól – o nazwie Reset.

W dokumencie hipertekstowym piszemy kod spełniający warunki zadania. Zaczynamy od napisania struktury strony. W przykładzie pozostawiono pustą sekcję **<head>**, ale śmiało można ją uzupełnić według przykładów z poprzednich rozdziałów, np. deklaracją kodowania polskich znaków **<meta charset="utf-8">**. W sekcji **<body>** wstawiono nagłówek formularza zgodnie z wytycznymi, a poniżej zdefiniowano poszczególne elementy formularza. Aby wymusić wysłanie e-maila z określonym tematem, korzystamy z atrybutu action w postaci:

```
<form action="mailto:wsip@wsip.pl?subject=Temat ankiety" method
="post">
```

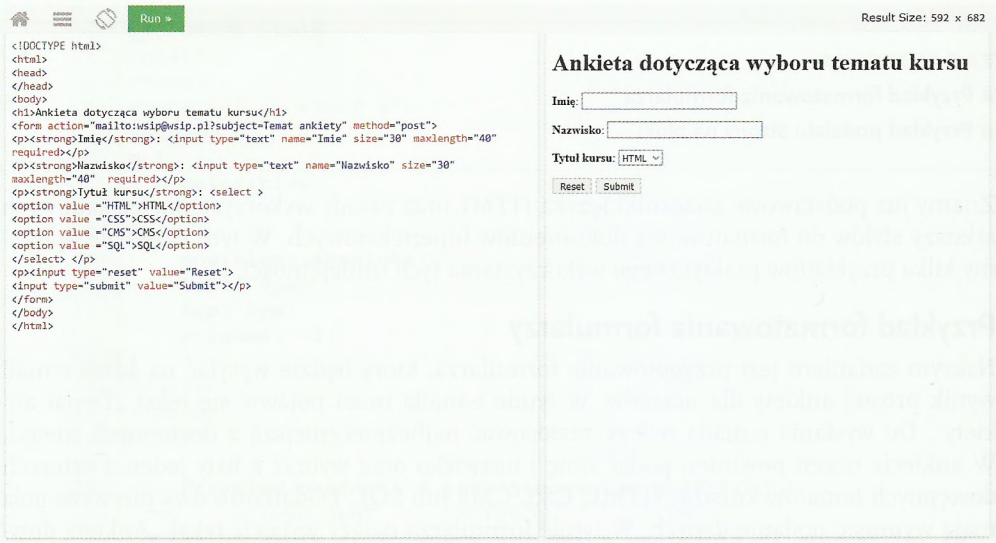
W kodzie stosujemy metodę POST, która jest bezpieczniejsza niż druga metoda – GET – ze względu na to, że parametrów transmisji nie widać w pasku przeglądarki.

Do pól „imię" i „nazwisko" zastosowano typ tekstowy o określonym rozmiarze, natomiast do wyboru tematu kursu wykorzystano pole **select**, które służy do tworzenia rozwijalnych list z opcjami, spośród których użytkownik wybiera interesujące go pozycje. Jako opcje podano nazwy kursów: HTML, CSS, CMS i SQL.

Aby wymusić wypełnienie pól imieniem i nazwiskiem, posłużono się atrybutem **required**, który w sytuacji, gdy użytkownik nie wpisze danych do pola formularza, wyświetli odpowiedni komunikat.

Pod polami formularza, w oddzielnym bloku wstawiono obok siebie dwa przyciski typu **reset** oraz **submit**.

Po zapisaniu dokumentu i otwarciu go w oknie przeglądarki otrzymano efekt pokazany na rysunku 3.20.



Rys. 3.20. Formularz utworzony w języku HTML

Formularz jest gotowy, ale wyraźnie brakuje mu walorów estetycznych oraz właściwego pozycjonowania. Aby nadać mu odpowiednie formatowanie, należy użyć kaskadowych arkuszy stylów. Dla czytelności zmian użyjemy stylów wewnętrznych i inline.

Styl wewnętrzny wstawiamy w sekcję **<head>**. Sformatuje on elementy: h1 oraz dwa przyciski typu **reset** i **submit**. Dla elementu h1 określamy kolor czcionki, rodzaj obramowania, efekt wyświetlania tekstu oraz jego wyśrodkowanie. Za pomocą grupowania selektorów formatujemy przyciski. Definiujemy dla nich kolejno: kolor tła, obramowanie, kolor tekstu, marginesy (tym samym wielkość przycisku), rodzaj kursora oraz zaokrąglenie obramowania.

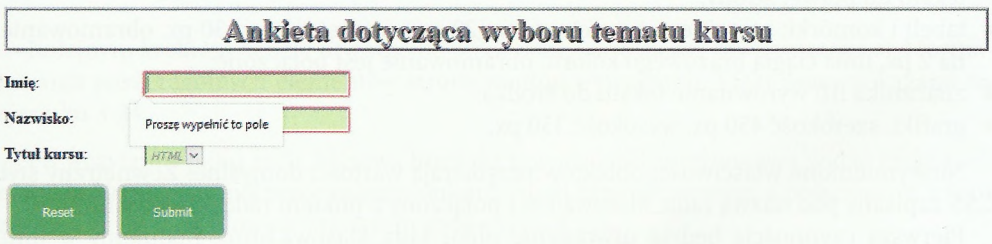
Styl inline ma formatować wygląd pól formularza. Definiujemy w nim pozycjonowanie, kolor tła oraz kolor i styl wyświetlania tekstu.

Gotowy listing przedstawiono na rysunku 3.21.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    color:grey;
    border-style: ridge;
    text-shadow: 2px 2px #000;
    text-align: center;
}
input[type=submit], input[type=reset] {
    background-color: #4CAF50;
    border: 4px solid #ccc;
    color: white;
    padding: 16px 32px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 8px;
}
</style>
</head>
<body>
<h1>Ankieta dotycząca wyboru tematu kursu</h1>
<form action="mailto:wsip@wsip.pl?subject=Temat ankiety" method="post">
<p><strong>Imię</strong>: <input type="text" name="Imie" size="30" maxlength="40" style="position: absolute; left: 150px;
background-color:#CCFF99; font-style: italic; color: gray" required></p>
<p><strong>Nazwisko</strong>: <input type="text" name="Nazwisko" size="30" maxlength="40" style="position: absolute;
left:150px; background-color:#CCFF99; font-style: italic; color: gray " required></p>
<p><strong>Tytuł kursu</strong>: <select style="position: absolute; left:150px; background-color:#CCFF99; font-style: italic;
color: gray ">
<option value ="HTML">HTML</option>
<option value ="CSS">CSS</option>
<option value ="CMS">CMS</option>
<option value ="SQL">SQL</option>
</select> </p>
<p><input type="reset" value="Reset">
<input type="submit" value="Submit"></p>
</form>
</body>
</html>
```

Rys. 3.21. Listing dokumentu hipertekstowego z formularzem

Po zapisaniu zmian i wyświetleniu formularza w przeglądarce uzyskamy efekt widoczny na rysunku 3.22.



Rys. 3.22. Sformatowany formularz

Przykład podziału strony na bloki

Zadanie polega na utworzeniu strony internetowej rady klasowej według określonych wytycznych.

Cechy strony:

- zastosowany standard kodowania polskich znaków;
- zastosowany zewnętrzny arkusz stylów;
- podział strony na bloki: panel lewy, panel prawy, stopka – zrealizowany za pomocą znaczników sekcji;
- gotowy plik zapisany pod nazwą rada_klasowa.html.

Zawartość panelu lewego:

- nagłówek pierwszego stopnia o treści: „Aktualności";
- nagłówek trzeciego stopnia o treści: „Co robimy?";
- lista numerowana z elementami: Pomagamy w nauce, Informujemy o prawach ucznia, Prowadzimy finanse klasy;
- nagłówek trzeciego stopnia o treści: „Skład rady:",
- tabela zgodna z rysunkiem 2, o wymiarach 3 wiersze na 2 kolumny, przy czym w pierwszej kolumnie wszystkie wiersze są scalone w jedną komórkę. Treść komórki scalonej: Jan Abacki Przewodniczący, przed słowem „przewodniczący" występuje łamanie linii. Treść kolumny 2. kolejno: Anna Babacka, Anna Cabacka, Krzysztof Nowak.

Zawartość panelu prawego:

- nagłówek pierwszego stopnia o treści: „Nasza klasa",
- osadzony rysunek klasa.jpg w oddzielnym akapicie.

W stopce strony umieszczony w akapicie napis o treści: „Autor strony: Jan Abacki"
Styl CSS witryny internetowej powinien zawierać formatowanie dla:

- panelu lewego: szerokość 40%, wysokość 600 px, kolor tła #FF6633;
- panelu prawego: szerokość 60%, wysokość 600 px, kolor tła #FF9933;
- stopki: kolor tła #993300, biały kolor czcionki, wysokość czcionki 150%, wyrównanie tekstu do prawej strony;
- tabeli i komórki: marginesy zewnętrzne na 20 px, wewnętrzne na 30 px, obramowanie na 2 px, linia ciągła brązowego koloru, obramowanie jest połączone;
- znacznika h1: wyrównanie tekstu do środka;
- grafiki: szerokość 450 px, wysokość 350 px.

Niewymienione właściwości obiektów przybierają wartości domyślne. Zewnętrzny styl CSS zapisany pod nazwą rada_klasowa.css i połączony z plikiem rada_klasowa.html.

Pierwszą czynnością będzie utworzenie pliku rada_klasowa.html. Kodujemy w nim strukturę strony, zawierającą kodowanie polskich znaków, połączenie z zewnętrznym plikiem arkusza stylów oraz podział strony na bloki za pomocą znaczników ** **. Kod gotowego pliku jest pokazany na rysunku 3.23.

```
<!DOCTYPE HTML>
<html lang="pl">
<head>
    <meta charset="utf-8"/>
    <title>Rada klasowa klasy 2TA</title>
    <link rel="stylesheet" type="text/css" href="rada_klasowa.css">
</head>
<body>
    <div id="lewy"><h1>Aktualności</h1>
    <h3>Co robimy?</h3>
        <ol>
         <li>Pomagamy w nauce</li>
         <li>Informujemy o prawach ucznia</li>
         <li>Prowadzimy finanse klasy</li>
        </ol>
    <h3>Skład Rady:</h3>
        <table callspecing="0">
         <tr>
            <td rowspan="4">Jan Abacki <br>Przewodniczący</td>
            <td>Anna Babacka</td>
        </tr>
            <tr>
            <td>Anna Cabacka</td>

            <tr>
            </tr>
                <tr>
            <td>Krzysztof Nowak</td>
            </tr>
    </table>
    </div>
    <div id="prawy"><h1>Nasza Klasa</h1>
        <p><img src="klasa.jpg"></p>
        </div>
        <div id="stopka">
            <p>Autor strony: Jan Abacki</p>
        </div>
</body>
</html>
```

Rys. 3.23. Listing pliku rada_klasowa.html

Kolejnym krokiem będzie utworzenie pliku rada_klasowa.css zawierającego styl formatowania poszczególnych elementów strony według wytycznych. Efekt pracy pokazano na rysunku 3.24.

Po wczytaniu pliku rada_klasowa.html do przeglądarki internetowej widać efekt końcowy, pokazujący współpracę między dokumentem hipertekstowym a połączonym z nim zewnętrznym kaskadowym arkuszem stylów (rys. 3.25).

4. Przeanalizuj kod strony internetowej z osadzonym wewnętrznym arkuszem stylów, który pokazano na rysunku 3.26. Zlokalizuj i popraw błędy składni kodu.

```
<!DOCTYPE html>
<html>
<head>
<meta name="utf-8">
<meta http-equiv="Content-Language" content="pl">
<meta name="keywords" content="">
<style>
<!nieodwiedzony link>
a:link {
    color: #FEF;
}

<!odwiedzony link>
a:visited {
    color: bardzojasnozielony;
}

/* najechany kursorem link */
a:hover {
    color: #hotpink;
}

/* wybrany link */
a:active {
    color: blue;
}
</style>
</head>
<body>
<video width=500 height=400 controls>
  <img src="film.mp4" type="video/mp4">
</video>
<p><b><a href="index.php">To jest link do mojej strony</a></b></p>
</body>
</html>
```

Rys. 3.26. Kod strony internetowej

5. Uzupełnij poprawiony kod w taki sposób, aby odsyłacz do strony index.php można było uruchomić przyciskiem, którego wzór pokazano na rysunku. Formatowanie przycisku wykonaj z wykorzystaniem kaskadowych arkuszy stylów.

Rys. 3.27. Wzór przycisku

6. Utwórz dokument hipertekstowy opisujący Góry Stołowe. Dodaj do niego rysunek ilustrujący piękno tego pasma górskiego. Za pomocą zewnętrznego kaskadowego arkusza stylów sformatuj tekst i obraz według następujących wytycznych:
   - tytuł strony „Góry Stołowe" w nagłówku drugiego stopnia;
   - tytuł strony w kolorze zielonym oraz wyśrodkowany;
   - obraz o wymiarach 400 px na 400 px;
   - wyrównanie obrazu do prawej strony.

   Gotowy plik hipertekstowy zapisz pod nazwą index.html, a plik z zewnętrznym arkuszem stylów – pod nazwą style.css. Efekt swojej pracy zaprezentuj nauczycielowi w przeglądarce internetowej.

4. Skrypty po stronie klienta – JavaScript

- Podstawy JavaScript
- Wyświetlanie komunikatorów na ekranie
- Okna dialogowe
- Typy danych
- Zmienne
- Operatory
- Instrukcje sterujące
- Funkcje w JavaScript
- Obiekty w JavaScript
- Zdarzenia w JavaScript
- Obsługa formularzy

4.1 Podstawy JavaScript

ZAGADNIENIA

- **Jakim językiem jest JavaScript?**
- **Różnice między JavaScript i Java**
- **Zastosowanie JavaScript**
- **Sposoby umieszczania skryptu w dokumencie HTML**

JavaScript jest językiem skryptowym dla dokumentów internetowych. Skrypty napisane za pomocą JavaScript mogą być umieszczane bezpośrednio na stronach HTML i wykonywane są za pomocą przeglądarki internetowej działającej na komputerze klienta. Język daje duże możliwości wzbogacania stron o interesujące interaktywne elementy. Można na przykład zdefiniować reakcję na zainicjowane przez czytelnika działania. Niektóre efekty jeszcze niedawno były możliwe jedynie dzięki użyciu CGI, a teraz jesteśmy w stanie zbudować wyrafinowane konstrukcyjnie strony za pomocą JavaScript.

JavaScript i Java nie są tymi samymi językami. Są to dwie odrębne techniki programowania w internecie. Java jest językiem programowania, natomiast JavaScript jest językiem skryptowym. Java to wywodzący się z C++ w pełni obiektowy język programowania, umożliwiający tworzenie aplikacji niezależne od platformy sprzętowej. JavaScript jest interpretowanym językiem, służącym do tworzenia niewielkich programów rozszerzających możliwości HTML w zakresie opisu stron.

Tabela 4.1. Podstawowe różnice pomiędzy językami Java oraz JavaScript

JavaScript	Java
Język jest interpretowany na komputerze klienta.	Język jest kompilowany do tzw. b-kodu, wykonywanego za pomocą wirtualnej maszyny Javy na komputerze klienta.
Obiektowy język skryptowy.	Język jest zorientowany obiektowo z obsługą wszystkich mechanizmów obiektowości.
Kod programu jest zagnieżdżony w kodzie HTML.	Kod programu jest niezależny od kodu HTML i znajduje się w oddzielnych plikach.
Zmienne i typy nie muszą być deklarowane przed użyciem.	Zmienne i ich typ muszą być zadeklarowane przed ich użyciem.
Odwołania do obiektów i funkcji są wykonywane podczas uruchamiania programu.	Wszystkie odwołania do obiektów i funkcji są sprawdzane na etapie kompilacji.

Skrypty tworzone w JavaScript zapewniają interaktywność strony poprzez reagowanie na zdarzenia, tworzenie elementów nawigacji oraz sprawdzenie poprawności formularzy.

Zalety języka JavaScript:

- możliwość modyfikacji wyglądu bieżącego dokumentu;
- wyświetlanie okien dialogowych;
- sprawdzanie poprawności formularza;
- dostęp do daty i czasu.

Skrypty JavaScript mogą być umieszczane wewnątrz kodu HTML, są to tzw. skrypty osadzone. Aby umieścić skrypt w HTML 4.01, należy użyć znacznika <script> w następujący sposób:

```
<script type="text/javascript">
kod skryptu
</script>
```

W przypadku HTML 5.0 parametr **type** nie jest już używany. Skrypty osadzone można umieszczać w nagłówku lub w ciele strony.

Drugą metodą umieszczenia skryptu w kodzie jest zapisanie go w oddzielnym pliku i użycie znacznika **<script>** z parametrem **src**:

```
<script type="text/javascript" src="lokalizacja skryptu">
kod skryptu
</script>
```

Plik ze skryptem ma rozszerzenie **js**.

W jednym dokumencie HTML można naraz umieścić wiele skryptów, zarówno osadzonych, jak i zewnętrznych.

Instrukcje umieszczane w skrypcie mogą być pisane w jednym wierszu lub w kilku. Średnik jest ogranicznikiem instrukcji, ale nie jest wymagany. Jeżeli w jednym wierszu zostanie zapisanych kilka instrukcji, to trzeba je oddzielić średnikiem.

Słowa kluczowe są pisane małymi literami, a nazwy obiektów wbudowanych rozpoczynają się wielką literą. Nazwy obiektów DOM są pisane małymi literami, ale w nazwach metod danych obiektów mogą być małe i wielkie litery. Definiowanym zmiennym, funkcjom czy obiektom można nadawać dowolne nazwy zgodnie z zasadami nadawania nazw (analogicznie jak w C++).

Komentarze mogą być umieszczane w dowolnym miejscu kodu. Jeśli komentarzem jest tekst w jednym wierszu, używamy znaku **//**:

```
//Komentarz do instrukcji
```

Jeżeli komentarz zawiera wiele wierszy, używamy **/*** do rozpoczęcia komentarza i ***/** do zakończenia komentarza.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest JavaScript?
2. Omów sposoby umieszczania skryptu w dokumencie HTML.
3. Czym różni się JavaScript od Javy?
4. Jak wprowadzać komentarze w skrypcie?

4.2 Wyświetlanie komunikatorów na ekranie

ZAGADNIENIA

- Działanie instrukcji document.write
- W jaki sposób wyświetlić tekst na stronie za pomocą skryptu?
- Jak formatować wyświetlany tekst?

Aby wyświetlić dowolny tekst na ekranie przeglądarki, używamy instrukcji **document.write**, np.:

```
document.write(„Komunikat");
```

PRZYKŁAD 4.1.

```
1  <!DOCTYPE HTML>
2  <HTML>
3  <HEAD>
4  <META CHARSET="UTF-8">
5  <TITLE>Tworzenie skryptów w JS</TITLE>
6  </HEAD>
7  <BODY>
8  <SCRIPT>
9  document.write("To mój pierwszy skrypt");
10 </SCRIPT>
11 </BODY>
12 </HTML>
13
14
```

Rys. 4.1. Przykładowy kod ilustrujący użycie instrukcji **document.write**

Powyższy skrypt spowoduje, że na ekranie pojawi się napis „To mój pierwszy skrypt".

Argumenty funkcji **write** są traktowane przez przeglądarkę jak tekst w HTML. Oznacza to, że do wyświetlanych komunikatów można wstawiać dowolne znaczniki formatujące tekst, jak również znaczniki przejścia do nowego wiersza itd.

PRZYKŁAD 4.2.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Tworzenie skryptów w JS</TITLE>
</HEAD>
<BODY>
<SCRIPT>
document.write("<h1>To mój <u>pierwszy skrypt</u></h1>");
document.write("<br>Każdy wyświetlany komunikat można formatować znacznikami HTML.");
</SCRIPT>
</BODY>
</HTML>
```

Rys. 4.2. Przykład formatowania tekstu wewnątrz skryptu

Efekt powyższego jest następujący:

To mój pierwszy skrypt

Każdy wyświetlany komunikat można formatować znacznikami HTML

Rys. 4.3. Efekt formatowania tekstu

SPRAWDŹ SWOJĄ WIEDZĘ

1. Podaj postać instrukcji **`document.write`**.
2. Jak można sformatować tekst wyświetlany przez skrypt?

4.3 Okna dialogowe

ZAGADNIENIA

- Co to jest okno dialogowe?
- Rodzaje okien dialogowych
- Jak wyświetlić okno informacyjne?
- Jak wyświetlić okno decyzyjne?
- Jak wyświetlić okno tekstowe?

Okno dialogowe jest to narzędzie, które pozwala na nawiązanie interakcji z użytkownikiem.

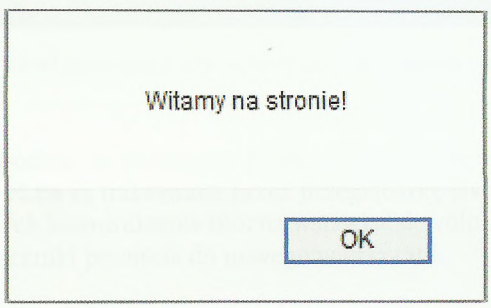
W języku JavaScript istnieją trzy rodzaje okien dialogowych:

a) informacyjne,
b) decyzyjne,
c) tekstowe.

Okna informacyjne służą do przekazania określonej informacji i nie mają wpływu na dalsze działanie skryptu. Do wyświetlenia tego rodzaju okna służy metoda **alert**:

```
alert(„Witamy na stronie!");
```

Powyższa instrukcja spowoduje wyświetlenie okna dialogowego z tekstem „Witamy na stronie!" oraz przyciskiem OK:



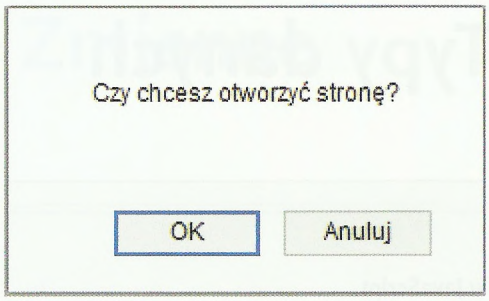
Rys. 4.4. Przykład okna informacyjnego

Okno decyzyjne służy do wyświetlenia treści komunikatu oraz dwóch przycisków – **OK** i **Anuluj**, które po wciśnięciu zwracają wartość logiczną **true** lub **false**.

Postać przykładowej instrukcji:

```
confirm(„Czy chcesz otworzyć stronę?");
```

Spowoduje ona wyświetlenie następującego okna dialogowego:

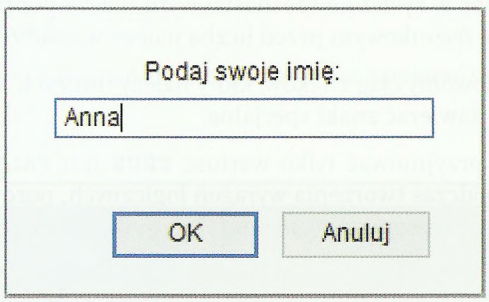


Rys. 4.5. Przykład okna decyzyjnego

Okna tekstowe służą do wyświetlenia komunikatu oraz pola umożliwiającego wprowadzanie danych przez użytkownika. Do utworzenia takiego okna dialogowego służy metoda **`prompt`**, np.:

```
prompt(„Podaj swoje imię: „);
```

Na ekranie przeglądarki pojawi się następujące okno:



Rys. 4.6. Przykład okna tekstowego

SPRAWDŹ SWOJĄ WIEDZĘ

1. Omów rodzaje okien dialogowych w JavaScript.
2. W jaki sposób można wyświetlić okno dialogowe na stronie?

4.4 Typy danych

ZAGADNIENIA

- **Rodzaje typów danych w JavaScript**
- **W jaki sposób określa się typ danych?**
- **Jakiego typu dane liczbowe można wprowadzać w skrypcie?**

W języku JavaScript można stosować kilka typów danych: liczbowy, łańcuchowy, logiczny, obiektowy oraz typy specjalne.

Typ liczbowy obejmuje różnego rodzaju liczby. JavaScript nie uwzględnia podziału na liczby całkowite i zmiennoprzecinkowe. Umożliwia wprowadzanie liczb w postaci dziesiętnej, ósemkowej lub szesnastkowej. Liczby dziesiętne zapisujemy w postaci tradycyjnej, np. 8, 123, –54 itd. Liczby w systemie szesnastkowym należy poprzedzić znakami **`0x`** lub **`0X`**, natomiast w zapisie ósemkowym przed liczbą należy wpisać zero.

Typ łańcuchowy to dowolny ciąg znaków, który należy umieścić w cudzysłowie lub między apostrofami. Może zawierać znaki specjalne.

Typ logiczny może przyjmować tylko wartość **`TRUE`** lub **`FALSE`** (prawda lub fałsz). Typ ten jest używany podczas tworzenia wyrażeń logicznych, porównywania wartości lub w celu sprawdzenia, czy dana operacja zakończyła się sukcesem.

Typ obiektowy to typ danych służący do reprezentacji obiektów.

Typy specjalne to **`NULL`** oraz **`UNDEFINED`**. Pierwszy z nich oznacza wartość pustą, drugi natomiast – wartość niezdefiniowaną.

W języku JavaScript istnieją również złożone typy danych, w których można przechowywać więcej niż jedną wartość. Złożonymi typami danych są:

- typ **`OBJECT`** – służy do reprezentacji obiektów;
- typ **`ARRAY`** – typ tablicowy.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień i omów typy danych.
2. Do czego służą typy specjalne?

4.5 Zmienne

ZAGADNIENIA

- Co to jest zmienna?
- Jak deklarować zmienne?
- Jak przypisać wartość do zmiennej?

Zmienna to element, który pozwala na przechowywanie danych i wyników. Każda zmienna ma swoją unikatową nazwę. Na uwagę zasługuje to, że w JavaScript nie trzeba deklarować typu zmiennej przed jej użyciem. Typ jest przypisany do zmiennej po nadaniu jej wartości. Ponadto typ zmiennej może się zmienić podczas działania skryptu.

Zmienne deklaruje się za pomocą słowa kluczowego **var**, poprzedzającego nazwę zmiennej. Aby zadeklarować zmienną, należy użyć zapisu:

```
var nazwa_zmiennej;
```

Nazwa zmiennej musi zaczynać się literą i nie może zawierać spacji (można je zastąpić znakiem podkreślenia (_).

PRZYKŁAD 4.3.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
 <META CHARSET="UTF-8">
 <TITLE>Tworzenie skryptów w JS</TITLE>
 </HEAD>
<BODY>
<SCRIPT>
 var tekst="Twoja liczba wynosi";
 var liczba=17;
 document.write("Zadeklarowałeś dwie zmienne. Jedna jest tekstem a druga liczbą.<br>");
 document.write(tekst+" "+liczba);

 </SCRIPT>
 </BODY>
 </HTML>
```

Rys. 4.7. Przykład deklaracji zmiennej liczbowej i łańcuchowej

W skrypcie zostały zadeklarowane dwie zmienne: **liczba** (przechowuje liczbę 17) i **tekst** (przechowuje łańcuch znaków). Aby w wyświetlanym komunikacie połączyć ciąg znaków, przechowywany w zmiennej **tekst** z liczbą przechowywaną w zmiennej **liczba**, należy użyć znaku „+”.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czy typ zmiennej może się zmienić podczas działania skryptu? Odpowiedź zilustruj przykładem.
2. W jaki sposób określa się typ zmiennej?

4.6 Operatory

ZAGADNIENIA

- **Rodzaje operatorów w JavaScript**
- **Co to jest konkatenacja?**
- **Jak wykonać przypisanie do zmiennej?**
- **Działania operatorów arytmetycznych, logicznych i bitowych**
- **Działania operatora inkrementacji i dekrementacji**

W JavaScript mogą być stosowane operatory:
a) arytmetyczne,
b) logiczne,
c) bitowe,
d) przypisania,
e) porównania,
f) łańcuchowy.

Operatory arytmetyczne

Operatory arytmetyczne to standardowe operatory służące do wykonywania działań matematycznych na zmiennych:

- operator dodawania +
- operator odejmowania –
- operator mnożenia *
- operator dzielenia /
- operator dzielenia modulo (wyznaczanie reszty z dzielenia) %.

PRZYKŁAD 4.4.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Tworzenie skryptów w JS</TITLE>
</HEAD>
<BODY>
<SCRIPT>
var liczba1=7;
var liczba2=2;
document.write(liczba1+" + "+liczba2+" = "+(liczba1+liczba2));
document.write("<br>"+liczba1+" - "+liczba2+" = "+(liczba1-liczba2));
document.write("<br>"+liczba1+" * "+liczba2+" = "+(liczba1*liczba2));
document.write("<br>"+liczba1+" / "+liczba2+" = "+(liczba1/liczba2));
document.write("<br>"+liczba1+" % "+liczba2+" = "+(liczba1%liczba2));
</SCRIPT>
</BODY>
</HTML>
```

Rys. 4.8. Zastosowanie operatorów arytmetycznych

Wynik działania tego skryptu pokazany jest na rysunku 4.9.

7 + 2 = 9
7 – 2 = 5
7 * 2 = 14
7 / 2 = 3.5
7 % 2 = 1

Rys. 4.9. Wynik działania skryptu ilustrujcego użycie operatorów arytmetycznych

Wyrażenia **liczba1+liczba2, liczba1-liczba2** itd. muszą być zapisane w nawiasach. Znak „+" z lewej strony wyrażenia służy do połączenia tekstu z wartością zmiennej.

Operatory logiczne

Argumentami operatorów logicznych muszą być wartości wyrażenia lub zmienne, a ich wynik posiada wartość logiczną **TRUE** lub **FALSE**. Wyróżniamy następujące operatory logiczne:

Tabela 4.2. Operatory logiczne w JavaScript

Symbol	Nazwa	Składnia	Działanie
&&	Koniunkcja	a&&b	Operator zwraca wartość **TRUE**, jeśli zmienne **a** i **b** są prawdziwe.
||	Alternatywa	a||b	Operator zwraca wartość **TRUE**, jeśli jedna ze zmiennych jest prawdziwa.
!	Negacja	!a	Operator neguje wyrażenie **a**.

PRZYKŁAD 4.5.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Tworzenie skryptów w JS</TITLE>
</HEAD>
<BODY>
<SCRIPT>
var liczba1=7;
var liczba2=-2;
if (liczba1>0 && liczba2<0)
{
document.write("Koniunkcja jest spełniona.");
}
else
{
document.write("Koniunkcja nie jest spełniona.");
}
</SCRIPT>
</BODY>
</HTML>
```

Rys. 4.10. Zastosowanie operatorów logicznych

Instrukcja warunkowa sprawdza, czy pierwsza liczba jest dodatnia, a druga liczba – ujemna. Jak widać, oba warunki są spełnione, zatem zostanie wyświetlony komunikat pierwszy. Wystarczy zmiennej **liczba2** przypisać wartość większą od zera, a warunek w instrukcji warunkowej nie zostanie spełniony i zostanie wyświetlony drugi komunikat.

Operatory bitowe

Operatory bitowe służą do wykonywania operacji na bitach w arytmetyce binarnej. W JavaScript istnieją następujące operatory:

Tabela 4.3. Operatory bitowe w JavaScript

Symbol	Składnia	Nazwa	Działanie
&	a&b	operator bitowy AND	Wyświetla 1, jeśli obie zmienne są jedynkami.
^	x^y	operator bitowy XOR	Wyświetla 1, jeśli jedna ze zmiennych (ale nie obie jednocześnie) jest jedynką.
|	x|y	operator bitowy OR	Wyświetla 1, jeśli jedna ze zmiennych jest jedynką.
<<	x<<y	przesunięcie w lewo	Przesunięcie bitów w lewo o określoną liczbę miejsc.
>>	x>>y	przesunięcie w prawo	Przesunięcie bitów w prawo o określoną liczbę miejsc.

PRZYKŁAD 4.6.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Operatory bitowe</TITLE>
</HEAD>
<BODY>
<SCRIPT>
var liczba1=8;
var liczba2=15;
document.write("Liczba "+liczba1+"= 1000");
document.write("<br>Liczba "+liczba2+"= 1111");
document.write("<br>8 & 15 = " +(liczba1&liczba2)+", czyli 1000");
document.write("<br>8 ^ 15 = " +(liczba1^liczba2)+", czyli 0111");
document.write("<br>8 | 15 = " +(liczba1|liczba2)+", czyli 1111");
document.write("<br>8 << 2 =  " +(liczba1<<2)+", czyli 100000");
</SCRIPT>
</BODY>
</HTML>
```

Rys. 4.11. Zastosowanie operatorów bitowych

Argumenty są konwertowane do 32-bitowej liczby całkowitej i wyrażone binarnie. Każdy bit w pierwszym argumencie jest dopasowany do odpowiedniego bitu w drugim argumencie, czyli pierwszy bit do pierwszego bitu, drugi bit to drugiego bitu itd. Operator działa na każdej parze bitów i wynik jest bitowy. Efekt działania skryptu jest następujący:

Liczba 8 = 1000
Liczba 15 = 1111
8 & 15 = 8, czyli 1000
8 ^ 15 = 7, czyli 0111
8 | 15 = 15, czyli 1111
8 << 2 = 32, czyli 100000

Rys. 4.12. Efekt działania skryptu z rys. 4.11

Operatory przypisania

Podstawowym operatorem przypisania jest znak równości. Instrukcja przypisania ma postać:

```
zmienna = wartość_lub_wyrażenie;
```

PRZYKŁAD 4.7.

```
<SCRIPT>
var liczba1=7;          //przypisanie zmiennej liczba1 wartości 7
var liczba2=-3;         //przypisanie zmiennej liczba2 wartości -3
var liczba3=(liczba1+liczba2)/2;    //przypisanie zmiennej liczba3 średniej arytmetycznej dwóch poprzednich

document.write("Średnia arytmetyczna liczb: "+liczba1+" i "+liczba2+" wynosi: " +liczba3);
</SCRIPT>
```

Rys. 4.13. Zastosowanie operatora przypisania

Aplikacja przypisuje dwóm zmiennym wartości liczbowe, a trzecia zmienna jest równa średniej arytmetycznej dwóch poprzednich zmiennych.

Ponadto istnieją jeszcze dodatkowe operatory przypisania (tabela 4.4):

Tabela 4.4. Operatory przypisania w JavaScript

Symbol	Składnia	Działanie
+=	a+=b	Wykonuje działanie a=a+b
-=	a-=b	Wykonuje działanie a=a-b
*=	a*=b	Wykonuje działanie a=a*b
/=	a/=b	Wykonuje działanie a=a/b
%=	a%=b	Wykonuje działanie a=a%b
<=	a<=b	Wykonuje działanie a=a<b
^=	a^=b	Wykonuje działanie a=a^b
|=	a|=b	Wykonuje działanie a=a|b
<<=	a<<=b	Wykonuje działanie a=a<<b
>>=	a>>=b	Wykonuje działanie a=a>>b

PRZYKŁAD 4.8.

```
<SCRIPT>
var a=8;
var b=15;
var c=10;
document.write("<b>Przed działaniem operatorów przypisania: </b>");
document.write("<br>a= "+a+" b= "+b+" c= "+c);
a+=b;
c*=b;
document.write("<br><b>Po działaniu operatorów przypisania: </b>");
document.write("<br>a= "+a+" b= "+b+" c= "+c);
</SCRIPT>
```

Rys. 4.14. Zastosowanie innych operatorów przypisania

Skrypt wykona dwa działania: suma liczb 8 i 15 zostanie przypisana do zmiennej **a**, iloczyn liczb 10 i 15 zostanie przypisany do zmiennej **c**:

Przed działaniem operatorów przypisania:
a= 8 b= 15 c= 10
Po działaniu operatorów przypisania:
a= 23 b= 15 c= 150

Rys. 4.15. Wynik działania skryptu z rys. 4.14

Często stosowanym operatorem jest operator *inkrementacji*, którego zadaniem jest zwiększenie wartości zmiennej o jeden. Operator ten jest zapisywany za pomocą dwóch plusów (**++**).

Drugim operatorem równie często stosowanym jest operator *dekrementacji*, który zmniejsza wartość zmiennej o jeden. Operator ten jest zapisywany za pomocą dwóch minusów (**--**).

PRZYKŁAD 4.9.

```
<SCRIPT>
var a=8;
var b=15;
document.write("<br><b>Przed działaniem operatorów: </b>");
document.write("<br>a= "+a+" b= "+b);
a++;
b--;
document.write("<br><b>Po działaniu operatorów: </b>");
document.write("<br>a= "+a+" b= "+b);
</SCRIPT>
```

Rys. 4.16. Zastosowanie operatora dekrementacji i inkrementacji

Powyższy skrypt zwiększy wartość **a** o jeden, natomiast wartość **b** zmniejszy o jeden:

Przed działaniem operatorów:
a= 8 b= 15
Po działaniu operatorów:
a= 9 b= 14

Rys. 4.17. Wynik działania skryptu z rys. 4.16

Operatory porównania

Operatory porównania służą do porównywania dwóch argumentów. Wynikiem jest wartość **TRUE** lub **FALSE**.

Tabela 4.5. Operatory porównania w JavaScript

Symbol	Składnia	Działanie
!=	a!=b	Wynikiem jest wartość **TRUE**, jeśli zmienne nie są równe.
==	a==b	Wynikiem jest wartość **TRUE**, jeśli zmienne są równe.
>	a>b	Wynikiem jest wartość **TRUE**, jeśli zmienna **a** jest większa od zmiennej **b**.
>=	a>=b	Wynikiem jest wartość **TRUE**, jeśli zmienna **a** jest większa od zmiennej **b** lub równa zmiennej **b**.
<	a<b	Wynikiem jest wartość **TRUE**, jeśli zmienna **a** jest mniejsza od zmiennej **b**.
<=	a<=b	Wynikiem jest wartość **TRUE**, jeśli zmienna **a** jest mniejsza od zmiennej **b** lub jej równa.

Operator łańcuchowy

Operator łańcuchowy umożliwia łączenie łańcuchów znakowych, czyli *konkatenację*. Operatorem tym jest znak **+**.

PRZYKŁAD 4.10.

```
document.write(„Mam „+35+" lat");
```

Jak wynika z powyższego przykładu, oprócz zmiennych tekstowych można również łączyć zmienne liczbowe. Na ekranie przeglądarki wyświetli się napis: „Mam 35 lat".

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień i opisz operatory arytmetyczne.
2. Jakie znasz operatory bitowe?
3. Czym są inkrementacja i dekrementacja?
4. Do czego służy operator %?

4.7 Instrukcje sterujące

ZAGADNIENIA

- **Postać i zastosowanie instrukcji warunkowej**
- **Co to jest instrukcja przetwarzania warunkowego?**
- **Do czego służą pętle?**
- **Jakie pętle są dostępne w JavaScript?**
- **Czym różni się instrukcja warunkowa od instrukcji switch?**

W bardziej złożonych skryptach jest konieczne stosowanie instrukcji sterujących, takich jak instrukcje warunkowe i pętle. Ten rozdział zawiera informacje na temat:
a) instrukcji warunkowych: **`if...else`** oraz **`switch`**,
b) pętli: **`for`**, **`while`**, **`do...while`**.

Instrukcje warunkowe

Instrukcje warunkowe służą do wykonywania określonych instrukcji w zależności od spełnienia warunków. Tak jak w przypadku języka C++, istnieją instrukcje warunkowe niepełna oraz pełna.

Postać niepełnej instrukcji warunkowej: instrukcje w poniższym kodzie zostaną wykonane tylko w przypadku, jeśli warunek przyjmuje wartość **TRUE**.

Postać instrukcji:

```
if (warunek)
{
instrukcje;
}
```

PRZYKŁAD 4.11.

```
<SCRIPT>
 var a=7;
 if (a>0)
 {
 document.write("Liczba "+a+" jest dodatnia.");
 }
</SCRIPT>
```

Rys. 4.18. Zastosowanie niepełnej instrukcji warunkowej

Skrypt sprawdza, czy zmienna **a** jest wartością dodatnią. Warunek jest spełniony, zatem w oknie przeglądarki zostanie wypisany komunikat: „**`Liczba 7 jest dodatnia`**".

Pełna instrukcja warunkowa pozwala na sprawdzenie warunku, a następnie wykonywanie instrukcji1 w przypadku, gdy warunek jest spełniony lub instrukcji2, jeśli warunek nie jest spełniony.

Postać pełnej instrukcji warunkowej:

```
if (warunek)
{
instrukcje1;
}
else
{
instrukcje2;
}
```

PRZYKŁAD 4.12.

```
<SCRIPT>
 var a=-7;
 if (a>0)
 {
 document.write("Liczba "+a+" jest dodatnia.");
 }
 else
 {
 document.write("Liczba "+a+" jest ujemna.");
 }
</SCRIPT>
```

Rys. 4.19. Zastosowanie pełnej instrukcji warunkowej

Poprzedni skrypt został rozszerzony o klauzulę **else**, po której wpisujemy instrukcje wykonujące się, jeśli warunek ma wartość **FALSE**. Zmienna **a=-7**, zatem na ekranie przeglądarki zostanie wyświetlony komunikat: „**Liczba -7 jest ujemna**".

W języku JavaScript istnieje jeszcze instrukcja przetwarzania warunkowego, która jest skróconą wersją instrukcji **if...else**. Jeśli warunek ma wartość **TRUE**, to zostanie wykonana pierwsza instrukcja. Jeśli natomiast warunek ma wartość **FALSE**, to zostanie wykonana druga instrukcja.

Postać instrukcji:

```
(warunek)? {instrukcja1;}:{instrukcja2;}
```

PRZYKŁAD 4.13.

```
<SCRIPT>
var liczba=prompt("Podaj liczbę: ");
 var wynik=(liczba%2==0)? "parzysta":"nieparzysta";

 document.write("Liczba "+liczba+" jest "+wynik);

</SCRIPT>
```

Rys. 4.20. Zastosowanie instrukcji przetwarzania warunkowego

Za pomocą okna dialogowego użytkownik zostanie poproszony o podanie liczby. Zmienna **wynik** zwraca wartość „**parzysta**" w przypadku, gdy podana liczba jest podzielna przez 2.

Zmienna **wynik** zwraca wartość „**nieparzysta**" w przypadku, gdy podana liczba nie jest podzielna przez 2.

Istnieje również możliwość dokonania kilku kolejnych sprawdzeń z wykorzystaniem klauzuli **else if**:

```
if (warunek1)
{
//kod wykonywany w przypadku, gdy warunek1 jest prawdziwy
}
else if (warunek2)
{
//kod wykonywany w przypadku, gdy warunek1 jest fałszywy,
a warunek2 prawdziwy
}
else if (warunek3)
{
//kod wykonywany w przypadku, gdy warunek1 i warunek2 są fałszywe,
a warunek3 prawdziwy
}
else
{
//kod wykonywany w przypadku, gdy wszystkie warunki są fałszywe
}
```

PRZYKŁAD 4.14.

```
<SCRIPT>
var liczba=prompt("Podaj liczbę: ");
if (liczba > 0) {
 alert("Liczba jest dodatnia");
} else if (liczba < 0) {
 alert("Liczba jest ujemna");
} else {
 alert("Liczba równa 0");
}
</SCRIPT>
```

Rys. 4.21. Zastosowanie instrukcji warunkowej z klauzulą else if

Skrypt sprawdza, czy podana liczba jest dodatnia, ujemna czy równa zero, i wyświetla odpowiedni komunikat.

Instrukcja SWITCH

Instrukcja wyboru **switch** pozwala w wygodny sposób sprawdzić ciąg warunków i wykonać różne instrukcje w zależności od wyników porównywania.

Postać instrukcji:

```
switch(wyrażenie){
  case wartość1:
    instrukcje1;
    break;
  case wartość2:
    instrukcje2;
    break;
  case wartość3:
    instrukcje3;
    break;
  default:
    instrukcje4;
}
```

Instrukcja **switch** sprawdza wartość wyrażenia i w zależności od wyniku są wykonywane różne instrukcje. Jeśli wynikiem jest **wartość1**, to wykonywane są **intrukcje1** i następuje przerwanie wykonywania bloku **switch** (przerwanie jest wykonywane przez instrukcję **break**). Jeśli wynikiem jest **wartość2**, to wykonywane są **intrukcje2** i następuje przerwanie wykonywania bloku **switch** itd. Jeśli nie zachodzi żaden z wymienionych przypadków, wykonywane są **instrukcje4** i następuje zakończenie instrukcji **switch**. Blok **default** jest jednak opcjonalny i może zostać pominięty.

Łatwo zauważyć, że jest to odpowiednik złożonej instrukcji **if...else if**.

PRZYKŁAD 4.15.

```
<SCRIPT>
var a=prompt("Podaj liczbę od 1 do 7:");
switch (a)
{
case "1": document.write("Dziś jest poniedziałek");
          break;

case "2": document.write("Dziś jest wtorek.");
          break;
case "3": document.write("Dziś jest środa.");
          break;
case "4": document.write("Dziś jest czwartek.");
          break;
case "5": document.write("Dziś jest piątek.");
          break;
case "6": document.write("Dziś jest sobota.");
          break;
case "7": document.write("Dziś jest niedziela.");
          break;
default: document.write("Wybrałeś złą liczbę.");
}
</SCRIPT>
```

Rys. 4.22. Zastosowanie instrukcji switch

Argumentem instrukcji **switch** jest zmienna **a**, której wartość jest wpisana za pomocą okna dialogowego. Wartość zmiennej **a** jest porównywana do wartości występujących po słowie **case**. Jeśli zgodność zostanie stwierdzona, zostaną wykonane instrukcje występujące w danym bloku **case**. Jeśli nie uda się dopasować wartości wyrażenia do żadnej z wartości występujących po słowach **case**, jest wykonywany blok **default**.

Uwaga: Przypadkowe pominięcie instrukcji **break** może doprowadzić do nieoczekiwanych wyników i błędów w skrypcie.

Pętla FOR

Zadaniem pętli **FOR** jest powtarzanie instrukcji określoną liczbę razy. Składnia tej instrukcji jest identyczna jak w języku C++:

```
for (wyr_początkowe; wyr_końcowe; wyr_modyfikujące)
{
instrukcje;
}
```

Wyrażenie początkowe określa wartość początkową zmiennej, która jest licznikiem ilości wykonywania pętli. Wyrażenie końcowe to warunek zakończenia działania pętli. Wyrażenie modyfikujące określa sposób modyfikacji licznika.

PRZYKŁAD 4.16.

```
<SCRIPT>
for (var i=1;i<=5;i++)
{
document.write("Oto mój skrypt!<br>");
}
```

Rys. 4.23. Zastosowanie instrukcji FOR

Efektem działania skryptu jest pięciokrotne wyświetlenie w oknie przeglądarki tekstu „**Oto mój skrypt**". Licznikiem pętli jest zmienna **i**, której wartość początkowa wynosi 1. Wartość końcowa licznika wynosi 5, a ostatni argument pętli zwiększa licznik o 1.

Oznacza to, że dopóki wartość zmiennej **i** jest mniejsza niż 5 bądź równa 5, należy zwiększać **i** o 1 oraz wykonywać instrukcję **document.write**.

Pętla WHILE

Pętla ma następującą składnię:

```
while (wyr_warunkowe)
{
instrukcje;
}
```

Jeśli warunek w pętli jest spełniony, to zostaną wykonane instrukcje wewnątrz pętli. Jeśli warunek przyjmuje wartość **TRUE**, to pętla będzie wykonywana do czasu, aż warunek osiągnie wartość **FALSE**. Może zatem się zdarzyć, że instrukcje nie zostaną wykonane ani razu, jeśli za pierwszym razem warunek przyjmie wartość **FALSE**.

PRZYKŁAD 4.17.

```
<SCRIPT>
var i=1;
while (i<=5)
{
document.write("Oto mój skrypt!<br>");
i++;
}
</SCRIPT>
```

Rys. 4.24. Zastosowanie instrukcji WHILE

Zadaniem skryptu jest pięciokrotne wyświetlenie tekstu za pomocą pętli **WHILE**. Przed wejściem do pętli wartość zmiennej i wynosi 1, zatem warunek pętli jest spełniony i zostaną wykonane instrukcje wewnętrzne.

Pętla DO...WHILE

Pętla **do...while** pozwala na wykonanie instrukcji wewnętrznych przynajmniej raz, zanim zostanie sprawdzony warunek logiczny.

Składnia pętli jest następująca:

```
do
{
instrukcje;
}
while(warunek)
```

PRZYKŁAD 4.18.

```
<SCRIPT>
var i=1;
do
{
document.write("Oto mój skrypt!<br>");
i++;
}
while (i<=5)
</SCRIPT>
```

Rys. 4.25. Zastosowanie instrukcji **DO...WHILE**

Zadaniem skryptu jest pięciokrotne wyświetlenie tekstu „**Oto mój skrypt!**". Przed wejściem do pętli wartość zmiennej i wynosi 1. Najpierw zostanie wyświetlony tekst i wartość licznika będzie zwiększona o 1, a dopiero potem nastąpi sprawdzenie warunku pętli.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jaka jest różnica między instrukcją warunkową a instrukcją przetwarzania warunkowego?
2. Podaj postać wszystkich instrukcji warunkowych.
3. Wymień wszystkie znane ci pętle i podaj różnice między nimi.

4.8 Funkcje w JavaScript

ZAGADNIENIA
- Co to jest funkcja?
- Argumenty funkcji
- Definiowanie funkcji
- Wywoływanie funkcji

Funkcja jest to grupa instrukcji służących do wykonania określonych, powtarzających się zadań.

Definicja funkcji zawiera:
a) słowo kluczowe **function**,
b) nazwę funkcji,
c) opcjonalną listę argumentów,
d) implementację funkcji.

Nazwa funkcji powinna zaczynać się od litery, a kolejnymi jej znakami mogą być litery, cyfry lub znak podkreślenia. Lista argumentów musi być zawarta między nawiasami okrągłymi, poszczególne argumenty rozdziela się przecinkami. Implementacja jest to ciąg instrukcji zawartych między nawiasami klamrowymi i wykonywanych przez funkcję. W ciele funkcji może znajdować się dowolna ilość instrukcji.

Ogólna definicja funkcji:

```
function nazwa(arg1, arg2, ..., argN)
{
instrukcje;
}
```

PRZYKŁAD 4.19.

```
function srednia(a,b)
{
document.write("Średnia liczb "+a+" oraz "+b+ " wynosi "+((a+b)/2)+".");
}
```

Rys. 4.26. Definicja funkcji

Funkcja w oknie **srednia** ma dwa argumenty, a jej zadaniem jest obliczenie i wyświetlenie na ekranie przeglądarki średniej dwóch liczb.

Aby wywołać zdefiniowaną funkcję, wystarczy podać jej nazwę i w nawiasach okrągłych podać wartości argumentów, na których funkcja ma wykonać obliczenia.

Zapis:

```
srednia(7,3);
```

spowoduje, że zostanie uruchomiona funkcja **srednia**, której pierwszym argumentem jest wartość 7, a drugim – wartość 3. Efektem będzie wyświetlenie na ekranie przeglądarki następującego komunikatu:

Średnia liczb 7 oraz 3 wynosi 5

Rys. 4.27. Wynik działania funkcji

Oprócz funkcji tworzonych przez użytkownika w JavaScript istnieją także funkcje wbudowane, np. **prompt**, **alert**, **confirm** (zostały omówione w rozdziale o oknach dialogowych).

Inne funkcje wbudowane to:

- **parseInt()** (pobiera argument typu tekstowego i konwertuje go na liczbę);
- **parseFloat()** (działa podobnie jak **parseInt()**, ale działa także na ułamkach);
- **Number()** (przekonwertuje argument na obiekt numeryczny);
- **isNaN()** (sprawdza, czy wartość podana jako parametr nie jest liczbą).

PRZYKŁAD 4.20.

```
<SCRIPT>
function suma(a,b)
{
var s=a+b;
return s;
}
var a =Number(prompt("Podaj liczbę: "));
var b =Number(prompt("Podaj liczbę: "));
var c= suma(a,b);
document.write("Suma liczb wynosi "+c);
</SCRIPT>
```

Rys. 4.28. Definicja i wywołanie funkcji

Funkcja o nazwie **suma** ma dwa argumenty i zwraca wartość będącą sumą tych dwóch argumentów. Skrypt prosi o podanie dwóch liczb za pomocą okna dialogowego, a następnie wartości te są przekazywane do funkcji **suma**.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień znane ci funkcje wbudowane.
2. W jaki sposób definiuje się funkcje?
3. W jaki sposób wywołuje się funkcję z argumentami?

4.9 Obiekty w JavaScript

ZAGADNIENIA

- **Co to jest obiekt?**
- **Sposób tworzenia własnych obiektów**
- **Rodzaje obiektów wbudowanych**
- **Wywoływanie metod dla obiektów wbudowanych**

Obiekty to konstrukcje programistyczne o właściwościach, którymi mogą być zmienne lub inne obiekty. Z obiektami są powiązane funkcje wykonujące operacje na właściwościach obiektu, które nazywamy *metodami*. Obiekt tworzymy za pomocą operatora **new**, wywołującego funkcję zwaną konstruktorem.

Aby dostać się do właściwości danego obiektu, stosujemy zapis:

```
nazwa_obiektu.nazwa_właściwości;
```

Oprócz możliwości tworzenia własnych obiektów JavaScript zawiera zbiór obiektów wbudowanych, mających własne właściwości i metody.

Obiekt WINDOW

Obiekt **WINDOW** znajduje się na samym szczycie hierarchii obiektów w JavaScript. Jest to obiekt domyślny, zatem do większości metod i właściwości można odwołać się z pominięciem jego nazwy. Obiekt ten opisuje bieżące okno przeglądarki. Do metod obiektu **WINDOW** można odwołać się poprzez następujące zapisy:

```
window.metoda();
metoda();
```

Tabela 4.6. Właściwości obiektu **WINDOW**

Właściwość	Opis
`Frames`	tablica ramek zawartych w bieżącym oknie
`Length`	liczba zdefiniowanych ramek
`Self`	bieżące okno
`Status`	tekst wyświetlany na pasku stanu okna przeglądarki
`defaultStatus`	tekst domyślnie pojawiający się w pasku stanu okna przeglądarki

Tabela 4.7. Metody obiektu **WINDOW**

Metoda	Opis
`alert()`	Wyświetla okno dialogowe informacyjne.
`close()`	Zamyka okno z dokumentem.
`confirm()`	Wyświetla okno dialogowe decyzyjne.
`open()`	Otwiera okno przeglądarki.
`prompt()`	Wyświetla okno tekstowe umożliwiające wprowadzenie wartości przez użytkownika.

Obiekt LOCATION

Obiekt **LOCATION** zawiera informacje dotyczące bieżącego adresu URL dokumentu oraz metody pozwalające na operowanie tym adresem.

Tabela 4.8. Właściwości obiektu **LOCATION**

Właściwość	Opis
`Href`	tekst (łańcuch) zawierający cały adres URL dokumentu
`Protocol`	łańcuch zawierający nazwę protokołu
`Host`	łańcuch zawierający nazwę serwera i nazwę domeny
`Hostname`	łańcuch zawierający pełną nazwę serwera łącznie z numerem portu
`Port`	łańcuch zawierający numer portu
`Pathname`	łańcuch zawierający adres ścieżki dostępu do pliku

Tabela 4.9. Metody obiektu **LOCATION**

Metoda	Opis
`assign()`	Wczytuje dokument o adresie wskazanym przez argument URL.
`reload()`	Wymusza ponowne wczytanie bieżącej strony.
`replace()`	Zastępuje bieżący dokument przez wczytany adres wskazany przez URL.

Obiekt DOCUMENT

Obiekt **DOCUMENT** zawiera metody i ma właściwości niezbędne do modyfikacji dokumentu HTML, który jest aktualnie wczytany przez przeglądarkę.

Tabela 4.10. Właściwości obiektu **DOCUMENT**

Właściwość	Opis
`Title`	łańcuch określający tytuł dokumentu
`Location`	łańcuch zawierający pełny adres URL bieżącego dokumentu
`lastModified`	data ostatniej modyfikacji dokumentu
`Referrer`	URL dokumentu, z którego został wywołany bieżący dokument
`bgColor`	kolor tła dokumentu
`fgColor`	kolor określający kolor tekstu w dokumencie
`linkColor`	kolor linku, który jeszcze nie został aktywowany
`vlinkColor`	kolor odwiedzonego linku
`alinkColor`	kolor aktywowanego linku
`forms[]`	tablica zawierająca wszystkie formularze w dokumencie
`links[]`	tablica zawierająca wszystkie linki znajdujące się w dokumencie

Tabela 4.11. Metody obiektu **DOCUMENT**

Metoda	Opis
`write()`	Wypisuje podany komunikat w bieżącym dokumencie.
`clear()`	Czyści zawartość bieżącego okna.
`close()`	Zamknięcie bieżącego okna.

Do odwołania się do elementu strony służą metody **`getElementById()`** oraz **`getElementByTagName()`**. Pierwszej z nich używamy, gdy chcemy odwołać się do elementu, który ma atrybut id, natomiast drugiej – gdy pobieramy kolekcję zawierającą elementy danego typu.

Obiekt STRING

Obiekt **`STRING`** służy do przechowywania łańcuchów znakowych. Ma on tylko jedną właściwość **`LENGTH`**, określającą liczbę znaków w łańcuchu. Obiekt ten zawiera natomiast dużą ilość metod.

Efekt działania skryptu jest następujący:

```
Dzisiaj jest: Sun Oct 01 2017 19:42:28 GMT+0200
Czas w GMT: Sun, 01 Oct 2017 17:42:28 GMT
Czas lokalny: 1.10.2017, 19:42:28
Miesiąc: 10
Dzień miesiąca: 1
```

Rys. 4.32. Efekt działania skryptu ilustrujący działanie metod obiektu DATE

Obiekt MATH

Obiekt **MATH** jest stosowany do wykonywania różnego rodzaju obliczeń matematycznych, udostępnia wiele stałych matematycznych oraz metod.

Tabela 4.14. Właściwości obiektu **MATH**

Właściwość	Opis
`LN10`	stała wartość; logarytm naturalny z 10
`LN2`	stała wartość; logarytm naturalny z 2
`PI`	stała wartość; liczba π
`SQRT1_2`	stała wartość; pierwiastek kwadratowy z ½
`SQRT2`	stała wartość; pierwiastek kwadratowy z 2

Tabela 4.15. Metody obiektu **MATH**

Metoda	Opis
`abs(x)`	wartość bezwzględna z liczby **x**
`ceil(x)`	najmniejsza liczba całkowita większa od **x** lub równa **x**
`cos(x)`	cosinus liczby **x**
`exp(x)`	stała Eulera podniesiona do potęgi **x**
`floor(x)`	największa liczba całkowita mniejsza od **x** lub równa **x**
`log(x)`	logarytm naturalny **x**
`max(x,y)`	wyznaczanie większej z liczb **x** i **y**
`min(x,y)`	wyznaczanie mniejszej z liczb **x** i **y**
`pow(x,y)`	obliczenie **x** do potęgi **y**
`round(x)`	zaokrąglenie liczby **x** do najbliższej całkowitej
`sin(x)`	sinus liczby **x**
`sqrt(x)`	pierwiastek kwadratowy z liczby **x**
`random(x)`	zwrócenie liczby pseudolosowej z przedziału [0, 1)

PRZYKŁAD 4.23.

```
<SCRIPT>
var a=prompt("Podaj pierwszą liczbę: ");
var b=prompt("Podaj drugą liczbę: ");
document.write("Liczba a wynosi: "+a);
document.write("<br>Liczba b wynosi: "+b);
document.write("<br>Pierwiastek kwadratowy z liczby "+a+" wynosi: "+Math.sqrt(a));
document.write("<br>"+a+" do potęgi "+b+" wynosi: "+Math.pow(a,b));
document.write("<br>Spośród liczb "+a+" oraz "+b+" większa jest liczba "+Math.max(a,b));
</SCRIPT>
```

Rys. 4.33. Działanie metod obiektu MATH

Na ekranie przeglądarki otrzymamy następujące informacje:

Liczba a wynosi: 36
Liczba b wynosi: 2
Pierwiastek kwadratowy z liczby 36 wynosi: 6
36 do potęgi 2 wynosi: 1296
Spośród liczb 36 oraz 2 większa jest liczba: 36

Rys. 4.34. Efekt działania skryptu ilustrujący działanie metod obiektu MATH

Obiekt ARRAY

Tablice służą do przechowywania wartości wielu zmiennych. Obiekt **ARRAY** służy do modyfikacji tablic zmiennych. Aby utworzyć nową tablicę, należy zastosować instrukcję:

```
var nazwa_tablicy=new Array();
```

lub

```
var nazwa_tablicy=[];
```

Jeżeli w nawiasach zostanie podana liczba 5, to zostanie utworzona tablica zawierająca pięć pustych elementów.

PRZYKŁAD 4.24.

```
var tab1=new Array(10);
```

Tablicę można również tworzyć, wstawiając do niej konkretne wartości:

```
var tab2=new Array('Asia', 'Basia', 'Kasia');
```

Aby uzyskać dostęp do elementów tablicy, należy podać numer indeksu danego elementu. Elementy tablicy są numerowane od zera.

Instrukcja `document.write(tab2[1]);` wyświetli imię `'Basia'`.

PRZYKŁAD 4.25.

```
<script type="text/javascript">
t=new Array(5);
for (i=0; i<5; i++)
{
 t[i]=Math.round(100*Math.random())
 document.write("element "+i+ " wynosi "+t[i]+"<br>");
}
</script>
```

Rys. 4.35. Deklaracja tablicy. Losowanie liczb do tablicy

W skrypcie została zadeklarowana pięcioelementowa tablica, dla której zostały wygenerowane liczby z zakresu od 0 do 100. W celu wypisania wygenerowanych liczb należy zastosować pętlę **FOR**.

PRZYKŁAD 4.26.

```
<script>
t=new Array(8);
s=0;
for (i=0; i<8; i++)
{
 t[i]=Number(prompt("Podaj liczbę: "));
 s=s+t[i];
 }
 document.write("Wpisałeś następujace wartości do tablicy:");
 document.write("<table ><tr>");

for (i=0; i<8; i++)
{
document.write("<td>"+t[i]+"</td>");
}
document.write("</tr></table>");
 document.write("Suma liczb wynosi: "+s);
</script>
```

Rys. 4.36. Zastosowanie pętli **FOR** do tablic jednowymiarowych

Użytkownik zostanie poproszony o wpisanie za pomocą pętli **FOR** wartości dla 8-elementowej tablicy, następnie wartości te zostaną wyświetlone na tablicy oraz pojawi się wynik sumy tych wartości.

Wpisałeś następujace wartości do tablicy:

1	4	8	-8	34	2	6	3

Suma liczb wynosi: 50

Rys. 4.37. Wynik działania skryptu ilustrującego korzystanie z tablic jednowymiarowych

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest konstruktor?
2. Czym jest metoda?
3. Jakie znasz obiekty wbudowane?

4.10 Zdarzenia w JavaScript

ZAGADNIENIA

- **Co to jest zdarzenie?**
- **Rodzaje dostępnych zdarzeń w JavaScript**
- **Przypisywanie funkcji do danego zdarzenia**

Na stronie internetowej mogą zachodzić różnego rodzaju zdarzenia, np. po kliknięciu na przycisk formularza zostanie uruchomiona funkcja wyświetlająca dane wprowadzone wcześniej do formularza.

Zdarzenia można podzielić na:

a) zdarzenia myszy,
b) zdarzenia klawiatury,
c) zdarzenia dokumentu,
d) zdarzenia formularza.

Tabela 4.16. Rodzaje zdarzeń

Nazwa zdarzenia	Rodzaj elementu	Opis działania
`OnAbort`	obraz	anulowanie pobrania grafiki
`OnBlur`	okno, formularz	usunięcie aktywności pola
`OnChange`	text, textarea, select	zmiana wartości
`OnClick`	dowolny znacznik	kliknięcie dowolnego znacznika
`OnDbClick`	dowolny znacznik	podwójne kliknięcie dowolnego znacznika
`onDragDrop`	okno	przeciągnięcie obiektu poza okno
`OnError`	obraz, okno	błąd podczas ładowania
`OnFocus`	okno, formularz	uaktywnienie elementu
`OnKeyDown`	dokument, obraz, link, textarea	wciśnięcie (przytrzymanie) klawisza na klawiaturze
`onKeyPress`	dokument, obraz, link, textarea	wciśnięcie (krótkotrwałe) klawisza na klawiaturze
`OnKeyUp`	dokument, obraz, link, textarea	zwolnienie klawisza
`OnLoad`	ciało dokumentu.	ładowanie strony
`onMouseDown`	dokument, przycisk, link	wciśnięcie (przytrzymanie) klawisza myszy
`onMouseOut`	wszystko	opuszczenie danego obszaru przez kursor myszy

Nazwa zdarzenia	Rodzaj elementu	Opis działania
`onMouseOver`	wszystko	najechanie kursorem myszy na dany obszar
`OnMouseUp`	wszystko	puszczenie klawisza myszy
`OnMove`	okno	poruszanie oknem
`OnReset`	formularz	wyczyszczenie formularza
`OnResize`	okno	zmiana rozmiaru okna
`OnSelect`	text, textarea	wybranie danego elementu
`OnSubmit`	formularz	zatwierdzenie formularza
`OnUnload`	ciało dokumentu	zamknięcie strony

PRZYKŁAD 4.27.

Rozważmy aplikację, w której zostanie wywołane zdarzenie **`onLoad`** w momencie uruchomienia strony. Zdarzenie to wywoła funkcję **`powitaj()`**, której argumentem jest imię podane poprzez okno dialogowe.

```
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Tworzenie skryptów w JS</TITLE>
<SCRIPT>
function podaj_imie()
{
var imie=prompt("Jak masz na imię? ");
if (imie==null || imie==" ")
{
document.write("Miło, że jesteś z nami");
}
else
{
document.write("Miło, że jesteś z nami "+imie);
}
return imie;
}
function powitaj(imie)
{
alert("Witaj na stronie "+imie);
}
</SCRIPT>
</HEAD>
<BODY onLoad="powitaj(imie)">
<script>
var imie=podaj_imie();
</script>
</BODY>
</HTML>
```

Rys. 4.38. Zastosowanie zdarzenia **`onLoad`**.

PRZYKŁAD 4.28.

Przykład ten obrazuje działanie zdarzenia **onClick**, które w tym przypadku wywołuje funkcję **koloruj()**. Zadaniem tej funkcji jest ustawienie koloru tekstu na niebieski, co zdarzy się po kliknięciu w tekst.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<META CHARSET="UTF-8">
<TITLE>Tworzenie skryptów w JS</TITLE>
<SCRIPT>
function koloruj()
{
document.fgColor = "blue";
}
</SCRIPT>
</HEAD>
<BODY>
<h1 onClick="koloruj()">PRZYKŁADOWY TEKST</h1>
</BODY>
</HTML>
```

Rys. 4.39. Zastosowanie zdarzenia **onClick**

PRZYKŁAD 4.29.

Rozważmy aplikację, w której po kliknięciu określonego obrazka zostanie wyświetlona nazwa zwierzątka i krótki wierszyk.
W sekcji głównej umieścimy tabelę zawierającą poszczególne ilustracje:

```
<div id="tabela" >
<table id="dane" >
<tr>
<td><img src="papuga.png" width="100px" height="100px" id="papuga" onclick="papuga()" > </td>
<td><img src="zyrafa.png" width="100px" height="100px" id="zyrafa" onclick="zyrafa()"></td>
<td><img src="krokodyl.png" width="100px" height="100px" id="krokodyl"onclick="krokodyl()"></td>
<td><img src="zebra.png" width="100px" height="100px" id="zebra" onclick="zebra()" ></td>
<td><img src="wilk.png" width="100px" height="100px" id="wilk" onclick="wilk()"></td>
</tr>
</table>
```

Rys. 4.40. Tabela z grafikami

W sekcji nagłówkowej umieszczamy skrypt, który zawiera funkcje uruchamiane dla poszczególnych obrazków.

```
<script>
function papuga(){
document.getElementById("wynik").innerHTML="PAPUGA"
document.getElementById("tresc").innerHTML="Papużko, papużko,<br> Powiedz mi coś na uszko.<br> Nic nie powiem, boś ty
plotkarz, <br>Powtórzysz każdemu, kogo spotkasz. ";
}
function zyrafa(){
document.getElementById("wynik").innerHTML="ŻYRAFA";
document.getElementById("tresc").innerHTML="Żyrafa tym głównie żyje,<br>że w górę wyciąga szyję. <br>A ja zazdroszczę
żyrafie, <br>ja nie potrafię.";
}
function krokodyl(){
document.getElementById("wynik").innerHTML="KROKODYL";
document.getElementById("tresc").innerHTML="Skąd ty jesteś, krokodylu?<br>Ja? Znad Nilu.<br>Wypuść mnie na kilka chwil,
<br>To zawiozę cię nad Nil.";
}
function zebra(){
document.getElementById("wynik").innerHTML="ZEBRA";
document.getElementById("tresc").innerHTML="Czy ta zebra jest PRAWDZIWA?<br>Czy to tak NAPRAWDĘ bywa?<br>Czy też malarz z
bożej łaski<br>Pomalował osła W PASKI?";
}
function wilk()
{
document.getElementById("wynik").innerHTML="WILK";
document.getElementById("tresc").innerHTML="Powiem ci w słowach kilku,<br>Co myślę o tym wilku:<br>Gdyby nie był na
obrazku,<br>Zaraz by cię zjadł, głuptasku. ";
}
</script>
```

Rys. 4.41. Definicje funkcji wywoływanych dla zdarzenia **onClick**

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wykorzystaj pokazane w rozdziale 4.10 fragmenty kodu i utwórz strukturę całej strony oraz arkusz CSS, aby osiągnąć efekt pokazany na rysunku 4.42. Kliknięcie w inny obrazek ma wywołać odpowiedni wierszyk.

Rys. 4.42. Efekt działania aplikacji ze zdarzeniami uruchamianymi po kliknięciu obrazka

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień grupy zdarzeń.
2. Wymień zdarzenia myszy.
3. Wymień zdarzenia klawiatury.

4.11 Obsługa formularzy

ZAGADNIENIA

- Zdarzenia obsługujące formularze
- Jak zabezpieczyć formularz?
- Stosowanie zdarzeń związanych z formularzem
- Sprawdzanie poprawności formularza

Formularze na stronach WWW są używane zazwyczaj do zebrania danych od osób odwiedzających stronę i wysłania ich do serwera WWW. Nic nie stoi jednak na przeszkodzie, aby z danych wpisanych do formularza korzystać także po stronie klienta. Język JavaScript pozwala na dostęp do formularzy znajdujących się na stronie WWW. Można m.in. odczytywać wprowadzane dane, zmieniać je lub wysłać formularz do serwera. Pomaga także w sprawdzaniu, czy wszystkie pola formularza zostały uzupełnione.

Do tworzenia formularzy w języku HTML służy znacznik <**FORM**>. Każdy element formularza jest traktowany jako osobny obiekt mający swój zestaw właściwości. Dodatkowo każdy formularz na stronie (ze swoimi elementami) jest osobną właściwością obiektu **document**, a odwołać się do niego można poprzez tablicę **forms** – należy podać odpowiedni numer porządkowy. Do pierwszego formularza zdefiniowanego na stronie można się odwołać poprzez wywołanie **document.forms[0]**, do drugiego – przez **document.forms[1]** itd. Jeżeli formularz ma nazwę, to odwołanie do niego wygląda następująco:

```
document.nazwa_formularza
```

Każdy formularz to osobny obiekt, reprezentowany przez obiekt **form** języka JavaScript. Do jego elementów można się odwołać poprzez tablicę **elements**:

```
document.forms[0].elements[0]    //pierwszy element pierwszego
                                 //formularza na stronie
```

lub

```
document.nazwa_formularza.nazwa_elementu.
```

Każdy element formularza ma również właściwości, do których można się odwołać przez podanie odpowiedniej nazwy właściwości:

```
document.forms[0]. elements[0].name
```

lub

```
document.forms[0]. elements[0]. value.
```

Odwołując się do elementu formularza wewnątrz jego definicji, możemy użyć konstrukcji **this.właściwość**.

PRZYKŁAD 4.30.

```
<SCRIPT>
function powitanie(form1)
{
    if (form1.imie.value=="")
    {
    document.write("Nie podałeś swojego imienia.");
    }
    else
    {
    document.write("Witaj "+form1.imie.value);
    }
}
</SCRIPT>
```

Rys. 4.43. Obsługa formularzy

W sekcji **BODY** dokumentu HTML został stworzony formularz o nazwie **form1** z jednym polem tekstowym i przyciskiem. Polu tekstowemu nadano nazwę **imie**. W skrypcie zdefiniowano funkcję **powitanie**, która zostanie uruchomiona po kliknięciu przycisku formularza (zostanie uruchomione zdarzenie **onClick**). Do funkcji **powitanie** przekazujemy **this.form**, czyli obiekt reprezentujący bieżący formularz o nazwie **form1**.

Funkcja sprawdza, czy w polu **imie** zostało wprowadzone imię. Jeśli użytkownik wpisał imię (lub dowolny cią znaków), zostanie wyświetlony komunikat powitalny, w przeciwnym wypadku przeglądarka powiadomi użytkownika, że nie zostało podane imię.

PRZYKŁAD 4.31.

```
<SCRIPT>
function pobierz()
{
    f=document.forms[0];
    napis=" \n";
    braki=false;
    if (f.imie.value=="")
    { napis+="imię\n";  braki=true; }

    if (f.nazw.value=="")
    { napis+="nazwisko\n"; braki=true;}

    if (f.data.value=="")
    { napis+="data urodzenia"; braki=true;}

    if (braki==false) alert("Wprowadziłeś wszystkie dane");
    else alert("Brakuje następujacych danych: \n"+napis);
}
</SCRIPT>
</HEAD>
<BODY>
<form method="post" name="form2">
<table>
<tr><td>Podaj swoje imię: </td><td> <input type="text" name="imie"></td></tr>
<tr><td>Podaj swoje nazwisko: </td><td> <input type="text" name="nazw"></td></tr>
<tr><td>Podaj rok urodzenia: </td><td> <input type="text" name="data" maxlength="4" size="4"></td></tr>
<tr><td colspan="2" align="right"><input type="button" name="p1" value="Kliknij" onClick="pobierz()"></td></tr>
</table>
</form>
</BODY>
```

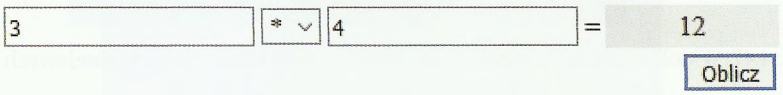
Rys. 4.44. Obsługa formularzy za pomocą tablicy **forms**

Funkcja **pobierz** sprawdza, czy do każdego z pól formularza zostały wprowadzone dane. Jeśli do formularza zostaną wprowadzone imię, nazwisko i data urodzenia, wyświetli się komunikat, że wszystkie dane są wprowadzone. Wystarczy, że choćby jedno pole zostawimy puste, a wyświetli się informacja, które pola nie zostały uzupełnione.

Obsługę formularza można także przeprowadzić, wykorzystując metodę **getElementById**.

PRZYKŁAD 4.32.

Rozważmy formularz będący kalkulatorem wykonującym cztery podstawowe działania arytmetyczne.



Rys. 4.45. Widok kalkulatora wykonującego cztery podstawowe działania arytmetyczne

Formularz zawiera dwa pola tekstowe, listę rozwijaną oraz przycisk, którego kliknięcie uruchomi funkcję **oblicz()**. Funkcja ta pobiera z formularza dwie liczby z pól tekstowych oraz znak działania arytmetycznego. Każdy z tych elementów ma swój unikatowy identyfikator **id**, wykorzystywany przez metodę **getElementById** do pobrania lub przypisywania wartości tego elementu.

```
<script>
function oblicz()
{
var x=Number(document.getElementById("liczba1").value);
var y=Number(document.getElementById("liczba2").value);
var z=document.getElementById("wyb").value;
if (z=="d1") document.getElementById("wynik").innerHTML=x+y;
if (z=="d2") document.getElementById("wynik").innerHTML=x-y;
if (z=="d3") document.getElementById("wynik").innerHTML=x*y;
if (z=="d4") document.getElementById("wynik").innerHTML=x/y;
}
</script>
</head>
<body>
<form method="post">
<table>
<tr><td><input type="text" id="liczba1"></td>
<td><select id="wyb" >
<option  value="d1">+</option>
<option  value="d2">-</option>
<option  value="d3">*</option>
<option  value="d4">/</option>
</select>
</td>
<td><input type="text" id="liczba2"></td>
<td>=</td><td id ="wynik"></td></tr>
<tr><td colspan="5"><input type="button" value="Oblicz" onClick="oblicz()"></td></tr>
</table>
</form>
```

Rys. 4.46. Obsługa formularzy za pomocą **getElementById**

PRZYKŁAD 4.30.

```
<SCRIPT>
function powitanie(form1)
{
    if (form1.imie.value=="")
    {
    document.write("Nie podałeś swojego imienia.");
    }
    else
    {
    document.write("Witaj "+form1.imie.value);
    }
}
</SCRIPT>
```

Rys. 4.43. Obsługa formularzy

W sekcji **BODY** dokumentu HTML został stworzony formularz o nazwie **form1** z jednym polem tekstowym i przyciskiem. Polu tekstowemu nadano nazwę **imie**. W skrypcie zdefiniowano funkcję **powitanie**, która zostanie uruchomiona po kliknięciu przycisku formularza (zostanie uruchomione zdarzenie **onClick**). Do funkcji **powitanie** przekazujemy **this.form**, czyli obiekt reprezentujący bieżący formularz o nazwie **form1**.

Funkcja sprawdza, czy w polu **imie** zostało wprowadzone imię. Jeśli użytkownik wpisał imię (lub dowolny cią znaków), zostanie wyświetlony komunikat powitalny, w przeciwnym wypadku przeglądarka powiadomi użytkownika, że nie zostało podane imię.

PRZYKŁAD 4.31.

```
<SCRIPT>
function pobierz()
{
    f=document.forms[0];
    napis=" \n";
    braki=false;
    if (f.imie.value=="")
    { napis+="imię\n";  braki=true; }

    if (f.nazw.value=="")
    { napis+="nazwisko\n"; braki=true;}

    if (f.data.value=="")
    { napis+="data urodzenia"; braki=true;}

    if (braki==false) alert("Wprowadziłeś wszystkie dane");
    else alert("Brakuje następujacych danych: \n"+napis);
}
</SCRIPT>
</HEAD>
<BODY>
<form method="post" name="form2">
<table>
<tr><td>Podaj swoje imię: </td><td> <input type="text" name="imie"></td></tr>
<tr><td>Podaj swoje nazwisko: </td><td> <input type="text" name="nazw"></td></tr>
<tr><td>Podaj rok urodzenia: </td><td> <input type="text" name="data" maxlength="4" size="4"></td></tr>
<tr><td colspan="2" align="right"><input type="button" name="p1" value="Kliknij" onClick="pobierz()"></td></tr>
</table>
</form>
</BODY>
```

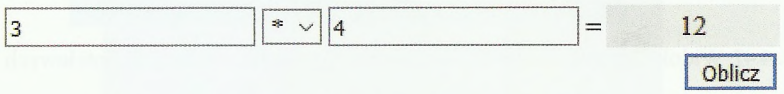
Rys. 4.44. Obsługa formularzy za pomocą tablicy **forms**

Funkcja **pobierz** sprawdza, czy do każdego z pól formularza zostały wprowadzone dane. Jeśli do formularza zostaną wprowadzone imię, nazwisko i data urodzenia, wyświetli się komunikat, że wszystkie dane są wprowadzone. Wystarczy, że choćby jedno pole zostawimy puste, a wyświetli się informacja, które pola nie zostały uzupełnione.

Obsługę formularza można także przeprowadzić, wykorzystując metodę **getElementById**.

PRZYKŁAD 4.32.

Rozważmy formularz będący kalkulatorem wykonującym cztery podstawowe działania arytmetyczne.



Rys. 4.45. Widok kalkulatora wykonującego cztery podstawowe działania arytmetyczne

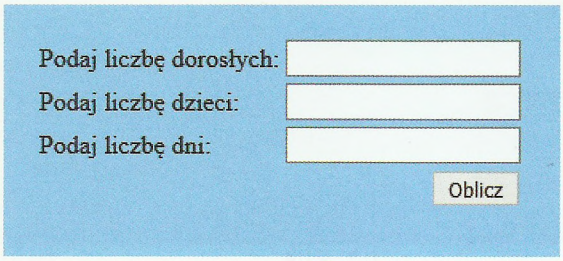
Formularz zawiera dwa pola tekstowe, listę rozwijaną oraz przycisk, którego kliknięcie uruchomi funkcję **oblicz()**. Funkcja ta pobiera z formularza dwie liczby z pól tekstowych oraz znak działania arytmetycznego. Każdy z tych elementów ma swój unikatowy identyfikator **id**, wykorzystywany przez metodę **getElementById** do pobrania lub przypisywania wartości tego elementu.

```
<script>
function oblicz()
{
var x=Number(document.getElementById("liczba1").value);
var y=Number(document.getElementById("liczba2").value);
var z=document.getElementById("wyb").value;
if (z=="d1") document.getElementById("wynik").innerHTML=x+y;
if (z=="d2") document.getElementById("wynik").innerHTML=x-y;
if (z=="d3") document.getElementById("wynik").innerHTML=x*y;
if (z=="d4") document.getElementById("wynik").innerHTML=x/y;
}
</script>
</head>
<body>
<form method="post">
<table>
<tr><td><input type="text" id="liczba1"></td>
<td><select id="wyb" >
<option  value="d1">+</option>
<option  value="d2">-</option>
<option  value="d3">*</option>
<option  value="d4">/</option>
</select>
</td>
<td><input type="text" id="liczba2"></td>
<td>=</td><td id ="wynik"></td></tr>
<tr><td colspan="5"><input type="button" value="Oblicz" onClick="oblicz()"></td></tr>
</table>
</form>
```

Rys. 4.46. Obsługa formularzy za pomocą **getElementById**

PRZYKŁAD 4.33.

Utwórzmy formularz do obsługi rezerwacji pokoju w hotelu. Formularz będzie umożliwiał wprowadzenie liczby osób dorosłych, dzieci oraz noclegów.



Rys. 4.47. Widok formularza umożliwiającego wprowadzanie danych hotelowych

Poniżej znajduje się kod tego formularza:

```
<div id="formularz">
<form method="post">
<table>
<tr><td>Podaj liczbę dorosłych:</td><td><input type="text" id="d1"></td></tr>
<tr><td>Podaj liczbę dzieci:</td><td><input type="text" id="d2"></td></tr>
<tr><td>Podaj liczbę dni:</td><td><input type="text" id="d3"></td></tr>
<tr><td colspan="2" align="right">
<input type="button" id="przycisk" value="Oblicz" onClick="przelicz()">
</td></tr>
</table></form></div>
<div id="wynik"></div>
```

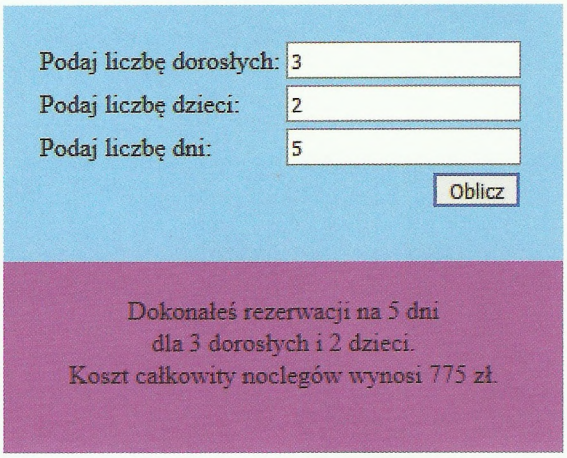
Rys. 4.48. Kod formularza umożliwiającego wprowadzanie danych hotelowych

Skrypt ma za zadanie obliczyć koszt pobytu każdej osoby dorosłej (35 zł za dobę) oraz koszt pobytu dziecka (25 zł za dobę). Skrypt sprawdza ponadto, czy użytkownik wprowadził wartość do formularza i czy jest ona liczbą.

```
<script>
function przelicz()
{
var dor=document.getElementById("d1").value;
var dz=document.getElementById("d2").value;
var dni=document.getElementById("d3").value;
if (!isNaN(dor) && !isNaN(dz) && !isNaN(dni)&& dor!="" && dz!="" && dni!="")
{
var koszt=(dor*35+dz*25)*dni;
document.getElementById("wynik").innerHTML="Dokonałeś rezerwacji na  "+dni+" dni <br> dla "+dor+" dorosłych i "+dz+ "
dzieci.<br>Koszt całkowity noclegów wynosi "+koszt+" zł.";
}}
</script>
```

Rys. 4.49. Funkcja obliczająca koszt rezerwacji hotelowej

Efektem aplikacji złożonej z powyższego formularza, skryptu oraz oczywiście sformatowanego według pewnego arkusza stylów jest następująca postać:



Rys. 4.50. Widok formularza z wprowadzonymi danymi dotyczącymi rezerwacji hotelowej

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień zdarzenia dotyczące obsługi formularza.
2. W jaki sposób sprawdzić, czy użytkownik wprowadził dane do formularza?

5. Grafika na stronach internetowych

- Obraz cyfrowy
- Edytory graficzne
- Zasady obróbki obrazów cyfrowych

5.1 Obraz cyfrowy

ZAGADNIENIA

- **Jak powstaje obraz cyfrowy?**
- **Podstawowe parametry opisujące obraz cyfrowy**
- **Grafika rastrowa**
- **Grafika wektorowa**
- **Rodzaje formatów plików grafiki cyfrowej**

Otaczającą nas rzeczywistość i ważne chwile z życia każdego z nas można utrwalić w postaci obrazów. Od dawna obrazy towarzyszą człowiekowi, są również świadectwem jego tryumfów i upadków. Do momentu pojawienia się systemów komputerowych dominował obraz malowany ręcznie lub drukowany, wywoływany z kliszy fotograficznej oraz zamieniony na sygnał telewizyjny, transmitowany na duże odległości.

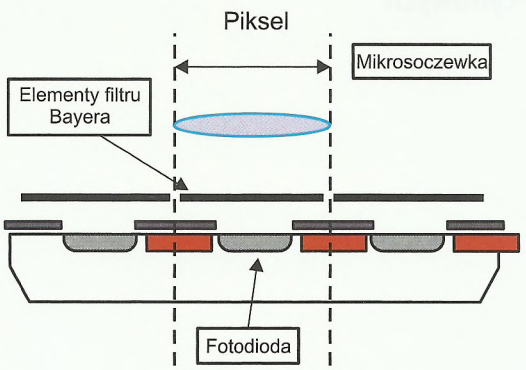
Obraz cyfrowy

Wraz z pojawieniem się technologii przechowywania różnych informacji w postaci cyfrowej istnieje możliwość zapisywania w taki sposób również obrazu.

Obraz cyfrowy (ang. *digital image*) to dwuwymiarowy obraz analogowy zapisany w postaci binarnej, możliwy do przechowywania w komórkach pamięci.

Obraz cyfrowy może powstać przez wykonanie zdjęcia aparatem z matrycą światłoczułą, zeskanowanie obrazu analogowego lub utworzenie mapy bitowej za pomocą aplikacji do edycji grafiki.

Pierwsze próby skanowania fotografii analogowej miały miejsce w roku 1957, jednak dopiero po wynalezieniu matrycy światłoczułej w roku 1969 nastąpił początek ery obrazu cyfrowego. Pierwszy egzemplarz matrycy **CCD** (ang. *Charge Coupled Device*) miał zaledwie osiem elementów światłoczułych umieszczonych w jednym rzędzie. Jeden element światłoczuły rejestruje, a następnie pozwala odczytać sygnał elektryczny dla jednego piksela.



Rys. 5.1. Przekrój matrycy CMOS

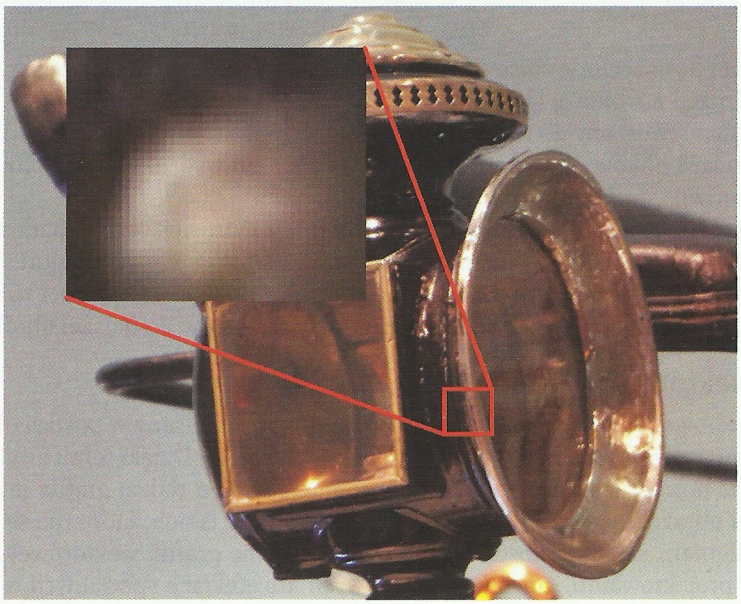
Piksel to najmniejszy element obrazu cyfrowego mający stałą barwę i określone położenie.

Dodatkowe zastosowanie filtru barwnego RGB daje możliwość rejestracji natężenia określonej szerokości spektrum światła w danym punkcie matrycy.

Grafika rastrowa

Matryca cyfrowa jest obecnie budowana z wielu milionów pikseli. Daje to możliwość zapisu i przetwarzania obrazów wysokiej jakości. Obraz zapisany za pomocą matrycy tworzy **bitmapę** (ang. *bitmap*), czyli tablicę o określonej liczbie pikseli opatrzonych atrybutami barwy. Obrazy cyfrowe zapisane w ten sposób nazywamy **grafiką rastrową** (ang. *raster image*).

Jakość grafiki rastrowej jest określana przez całkowitą liczbę pikseli (**rozdzielczość obrazu**) oraz ilość informacji o barwie zawartej w każdym pikselu. Im większa jest liczba pikseli w danym rozmiarze obrazu oraz liczba bitów opisujących barwę, tym lepsza jest jakość obrazu. Zbyt mała liczba pikseli powoduje powstanie efektu **pikselizacji**, czyli uwidocznienie się poszczególnych wiązek pikseli, prowadzące do deformacji obrazu.



Rys. 5.2. Efekt pikselizacji

Rozdzielczość jest podawana często jako liczba pikseli na jednostkę długości i opisana jednostką **ppi** (ang. *pixels per inch*). Dla przykładu, rozdzielczość wydrukowanego obrazu średniej jakości może wynosić 200 ppi. Oznacza to, że na wydruku o wielkości 1 na 1 cal znajduje się 40 000 pikseli.

Liczba bitów pamięci przeznaczonej do zapisania informacji o barwie jest nazywana głębią barwy, a jej jednostką jest **bpp** (ang. *bits per pixel*). Barwa opisana pojedynczym bitem tworzy obraz monochromatyczny, czyli jednokolorowy.

Współczesna grafika rastrowa stosuje głębie 24-, 32- oraz 48-bitową. Wierną reprezentację obrazu rzeczywistego daje już zastosowanie kodowania 24-bitowego. W modelu RGB przy zapisie 24-bitowym otrzymamy paletę 16,7 milionów kolorów. Liczbę tę otrzymamy, uzyskując 256 odcieni z każdej składowej modelu RGB ($2^8 \cdot 2^8 \cdot 2^8 = 16777216$).

Obrazy rastrowe możemy również tworzyć za pomocą aplikacji do edycji rastrowej. Grafika powstaje wówczas przez określenie wymiarów rysunku i nadanie określonych parametrów pikselom za pomocą narzędzi dostępnych w programie oraz zapisanie efektu pracy w pliku o formacie rastrowym.

Do najczęściej używanych edytorów grafiki rastrowej należą: Adobe Photoshop, GIMP czy MS Paint.

Podczas zapisywania obrazu cyfrowego, tworzonego zarówno przez kamerę czy aparat, jak i poprzez edycję w aplikacji, bardzo duże znaczenie ma wybór formatu pliku. Format może decydować o jakości, ilości pamięci potrzebnej do zapisu pliku lub o możliwości jego dalszej obróbki. W grafice rastrowej istotnym elementem operacji zapisu jest **kompresja obrazu** (ang. *data compression*). Kompresja pozwala zmniejszyć rozmiar pamięci zajmowanej przez plik. Jeżeli operacja kompresji pozwoli na powrót do oryginału, to nazywamy ją **kompresją bezstratną**. Jeżeli natomiast kompresja spowoduje utratę pewnej ilości danych, nazwiemy ją **kompresją stratną**.

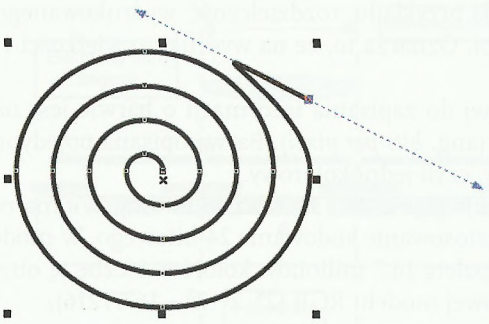
Do najpopularniejszych formatów plików grafiki rastrowej zaliczamy:

- **BMP** (domyślny format systemu Windows; charakteryzujący się dużym rozmiarem pliku);
- **GIF** (format powszechnie stosowany na stronach internetowych do zapisu prostej grafiki lub animacji z 8-bitową głębią koloru);
- **PNG** (format stworzony do zapisu grafiki na potrzeby stron internetowych);
- **TIFF** (standard plików graficznych przeznaczonych do wydruku lub publikacji o dużej głębi kolorów);
- **JPEG** (format powszechnie stosowany w fotografii cyfrowej, który charakteryzuje się dobrą jakością obrazu, niewielkimi rozmiarami pliku oraz stratną kompresją);
- **XCF** (wewnętrzny format popularnego programu GIMP bez możliwości kompresji);
- **RAW** (format zawierający wszystkie dane obrazu niezbędne do dalszej obróbki).

Grafika wektorowa

Grafika wektorowa (ang. *vector graphics*) to obraz cyfrowy zapisany w postaci obiektów lub figur geometrycznych opisanych wzorami matematycznymi. Dzięki temu zmiany rozmiaru obrazu nie powodują strat w jakości czy proporcjach. Ten rodzaj grafiki jest przydatny w tworzeniu obrazów z niewielką ilością szczegółów oraz takich, które nie są dokładnym odzwierciedleniem rzeczywistości. Główne zastosowanie grafiki wektorowej znajdziemy w reklamie, projektowaniu, grach komputerowych lub efektach specjalnych w filmach.

Obrazy wektorowe powstają w edytorach graficznych. Do konstrukcji figur geometrycznych służą **krzywe Béziera**, które są opisane punktami o współrzędnych wyrażonych funkcjami liczb rzeczywistych.



Rys. 5.3. Grafika wektorowa

Najczęściej używanymi edytorami grafiki wektorowej są: CorelDraw, Adobe Illustrator, Inkscape.

Do najpopularniejszych formatów plików grafiki wektorowej zaliczamy:

- **AI** (format programu Adobe Ilustrator);
- **SVG** (format do publikacji dwuwymiarowej grafiki wektorowej statycznej i animowanej na stronach internetowych);
- **SWF** (format wektorowej grafiki programu Adobe Flash);
- **CDR** (format macierzysty programu CorelDraw).

SPRAWDŹ SWOJĄ WIEDZĘ

1. W jaki sposób powstaje obraz cyfrowy?
2. Opisz matrycę CDD.
3. Czym charakteryzuje się grafika rastrowa?
4. Na czym polega różnica między kompresją bezstratną a kompresją stratną?
5. Wymień najpopularniejsze formaty plików grafiki rastrowej.
6. Na czym polega tworzenie grafiki wektorowej?

5.2 Edytory graficzne

ZAGADNIENIA

- **Zasady pracy w edytorze GIMP**
- **Zasady pracy w edytorze Inkscape**
- **Przykłady zastosowań praktycznych**

Do edycji grafiki cyfrowej służy wiele aplikacji o różnym poziomie możliwości tworzenia i przetwarzania obrazów. Wybraliśmy aplikacje, które oferują duże możliwości, a dostępne są w ramach licencji GPL (ang. *General Public License*), czyli nieodpłatnie, z otwartym źródłem.

GIMP

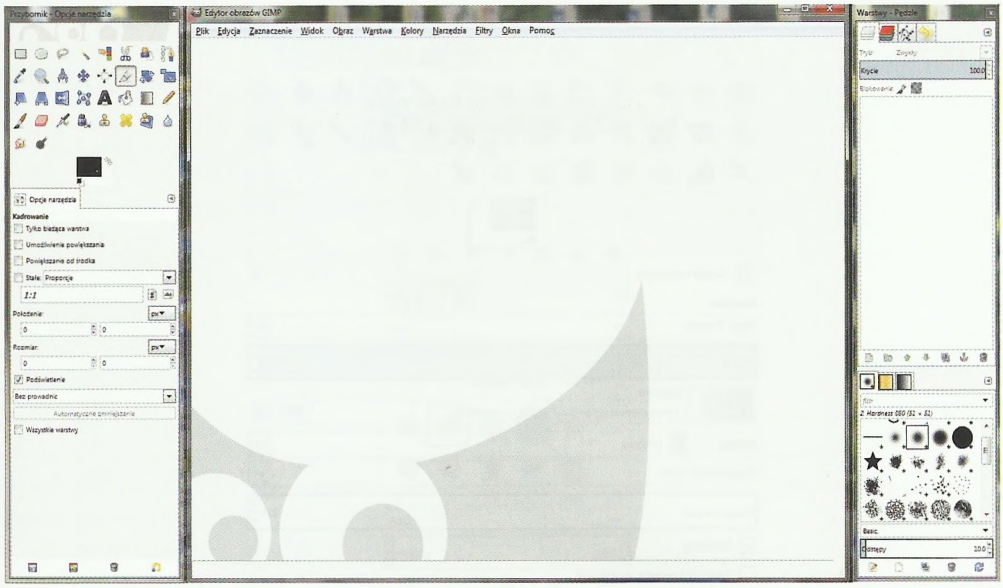
GIMP (ang. *GNU Image Manipulation Program*) – bezpłatny program do edycji grafiki rastrowej, który w ostatnich latach zdobył dużą popularność i stanowi poważną konkurencję dla Adobe Photoshop Elements – najtańszej wersji legendarnej aplikacji.

Technik informatyk podczas pracy nad witrynami i aplikacjami internetowymi edytuje prostą grafikę lub przetwarza już gotowe pliki (np. zdjęcia) – dopasowuje je do założeń projektowych lub technicznych. Na tym etapie nie powinny być realizowane zadania przeznaczone dla wykwalifikowanego grafika komputerowego. Dlatego taki edytor graficzny jak GIMP idealnie nadaje się do tego typu zadań. Bardziej zaawansowane projekty graficzne lub artystyczne wizualizacje powinny być realizowane przez profesjonalne agencje plastyczne.

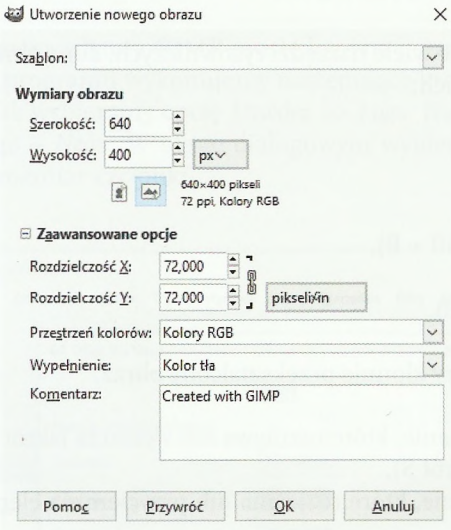
Program GIMP jest dostępny dla systemów: Windows, Linux oraz Mac OS. Pliki instalacyjne można pobrać bezpośrednio ze strony projektu: `https://www.gimp.org`.

Instalacja w systemach z rodziny Windows jest intuicyjna i nie powinna przysporzyć problemów. Po pierwszym uruchomieniu mamy do dyspozycji główne okno programu oraz okna przyborników.

Nowy obraz można utworzyć poprzez zakładkę *Plik*, umieszczoną w menu poleceń okna głównego. Podczas tworzenia nowego obrazu należy określić jego wymiary. Domyślnie ustawione są wymiary 640 px na 400 px z rozdzielczością 72 ppi i modelem barw RGB. Po wyborze opcji zaawansowanych można wybrać rodzaj wypełnienia, dodać komentarz lub zmienić domyślne ustawienia.

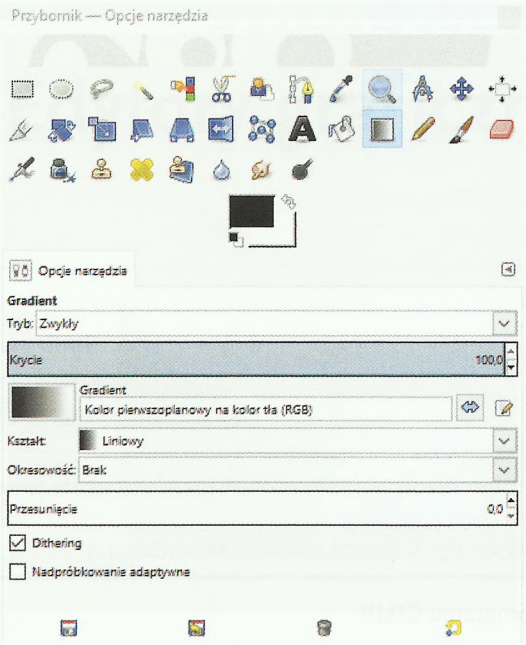


Rys. 5.4. Okno główne programu GIMP



Rys. 5.5. Okno tworzenia nowego obrazu w edytorze GIMP

Po utworzeniu nowego obrazu główne okno robocze zyskuje prowadnicę pionową oraz poziomą. W tym momencie jest gotowe do edycji. Edycję umożliwiają narzędzia umieszczone w przyborniku. Każde podwójne naciśnięcie lewego klawisza myszy na wybranym narzędziu w przyborniku powoduje otwarcie okna z opcjami narzędzia.



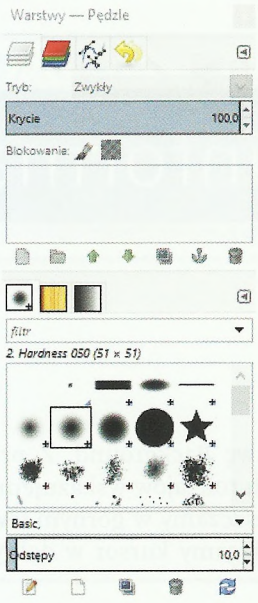
Rys. 5.6. Okno przybornika w edytorze GIMP

Program GIMP oferuje wiele narzędzi rysowniczych, domyślnie zgrupowanych w przyborniku. Zaliczamy do nich:

- ołówek (skrót N),
- pędzel (skrót P),
- aerograf (skrót A),
- stalówkę (skrót K),
- wypełnienie (skrót Shift + B),
- gradient (skrót L),
- gumkę (skrót Shift + E),
- klonowanie (skrót C),
- klon perspektywy, który klonuje przekształcony obraz,
- łatkę (skrót H),
- rozmywanie / wyostrzanie, które rozmywa lub wyostrza (skrót Shift + U),
- rozsmarowywanie (skrót S),
- rozjaśnianie / wypalenie, które rozjaśnia lub przyciemnia element obrazu (skrót Shift + D).

Obok przybornika znajduje się okno *Warstwy – Pędzle*. Znajdują się tutaj zakładki warstw, kanałów, ścieżek oraz historii działań. Warstwy to jedno z najważniejszych narzędzi w edytorach graficznych. Można z nich korzystać w pracy zarówno z obrazem rastrowym, jak i wektorowym.

Warstwa to niezależny poziom obrazu, który posiada unikatową nazwę i własności. Poszczególne warstwy leżą jedna na drugiej i tworzą stos. Wymiary każdej z warstw mogą być dowolne: warstwa może być większa lub mniejsza od głównego obrazu. Dzięki wykorzystaniu warstw można modyfikować poszczególne elementy stosu, nie naruszając całego obrazu.

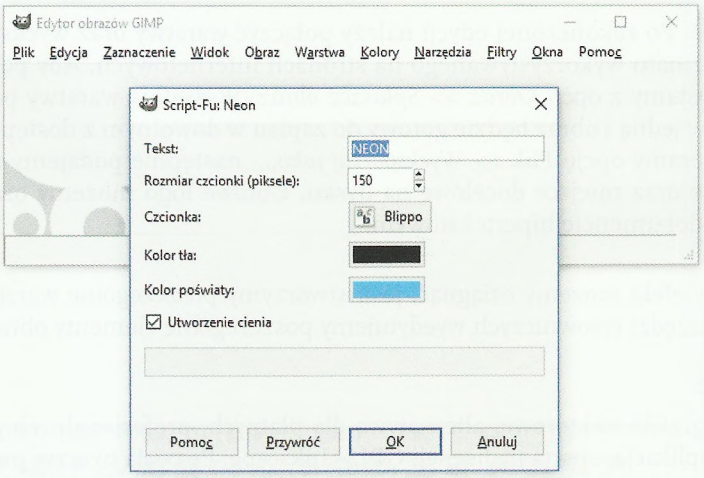


Rys. 5.7. Okno warstw i pędzli

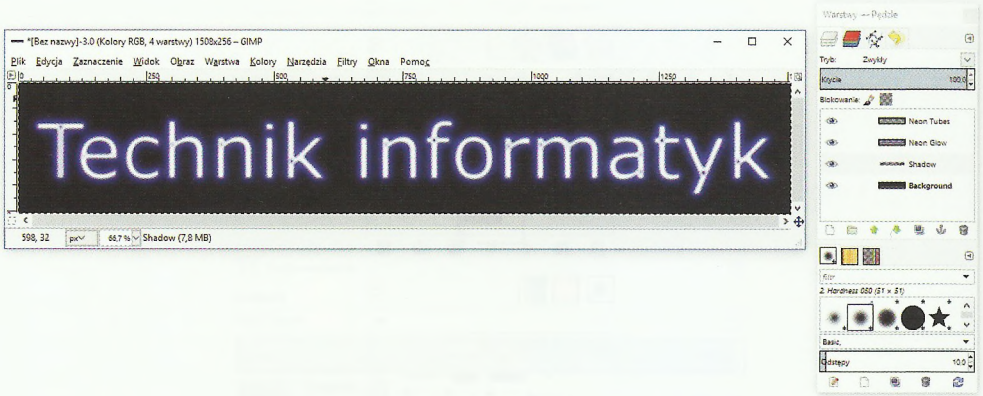
PRZYKŁAD 5.1.

Utworzymy za pomocą edytora GIMP proste logo na potrzeby strony internetowej. Po uruchomieniu programu wykonujemy następujące kroki:

Krok 1. Z menu *Plik* wybieramy opcję *Utwórz* >> *Logo*. Następnie z dostępnej listy wybieramy typ logo – *Neon*. W oknie dialogowym wybieramy kolor tła, kolor poświaty, tekst oraz rozmiar czcionki.



Krok 2. Po zatwierdzeniu zmian program wygeneruje nam gotowe logo. W oknie warstw widzimy warstwy, które posłużyły do utworzenia logo.



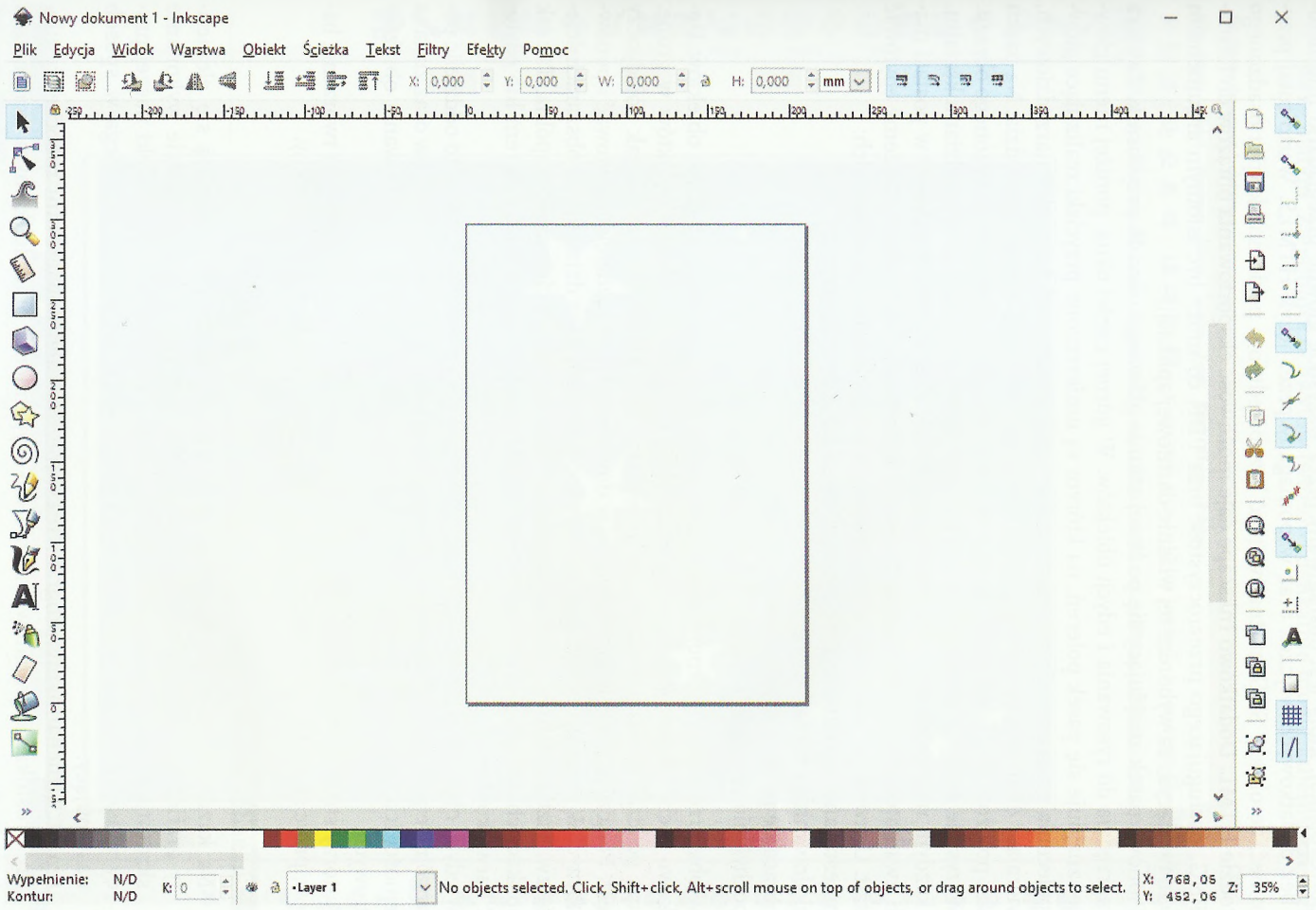
Krok 3. Zaznaczamy warstwę Background. Z przybornika wybieramy narzędzie *Gradient*. Z listy dostępnych rodzajów wskazujemy wzór *Skyline* oraz kształt liniowy. Kursor gradientu umieszczamy w górnym lewym rogu okna logo i naciskając lewy klawisz myszy, przeciągamy kursor w dolny prawy róg okna. Gotowy efekt pokazuje rysunek.

Krok 4. Po zakończonej edycji należy połączyć warstwy oraz wyeksportować obraz do formatu wykorzystywanego na stronach internetowych. Aby połączyć warstwy, korzystamy z opcji *Obraz* >> *Spłaszcz obraz*. Wszystkie warstwy połączą się wówczas w jedną i obraz będzie gotowy do zapisu w dowolnym z dostępnych formatów. Wybieramy opcję *Plik* >> *Wyeksportuj jako...*, następnie podajemy nazwę i rozszerzenie oraz miejsce docelowe na dysku. Gotowe logo możemy osadzać w dowolnym dokumencie hipertekstowym.

Podobny efekt możemy osiągnąć, jeśli stworzymy poszczególne warstwy ręcznie i za pomocą narzędzi rysowniczych wyedytujemy poszczególne elementy obrazu.

Inkscape

Do edycji grafiki wektorowej alternatywą dla płatnych, profesjonalnych programów jest darmowa aplikacja, oparta na licencji GPL – Inkscape. Pozwala tworzyć przede wszystkim symbole, znaki towarowe, przyciski, elementy tła czy ikony. Pliki instalacyjne programu można pobrać ze strony **`https://inkscape.org`**.



Rys. 5.8. Okno główne programu Inkscape

Obrazy utworzone w programie Inkscape są zapisywane w formacie SVG (ang. *Scalable Vector Graphics*), który jest obsługiwany przez wszystkie główne przeglądarki internetowe bez dodatkowych wtyczek oraz wspomagany przez język HTML5. Warto tutaj przypomnieć, że obrazy SVG można tworzyć bezpośrednio z poziomu skryptu osadzonego w kodzie HTML. Dodatkowo Inkscape daje możliwość eksportowania obrazów do formatu PNG, obsługującego przezroczystość oraz PDF, co może być istotnym argumentem przemawiającym za wyborem tej właśnie darmowej aplikacji.

Pionowy pasek znajdujący się po lewej stronie głównego okna to przybornik. Zawiera on narzędzia do rysowania i edycji obiektów. W górnej części okna, poniżej menu głównego, znajduje się pasek poleceń, na którym są umieszczone przyciski realizujące główne polecenia programu. Poniżej tego paska znajduje się pasek kontrolek narzędziowych, zawierający elementy kontrolne przypisane do aktualnie wybranego narzędzia. Na pasku stanu, położonym na dole okna, są wyświetlone informacje dotyczące tworzonego obrazu.

Wszystkie narzędzia w edytorze grafiki wektorowej nazywają się narzędziami kształtu. Na każdym utworzonym lub zaznaczonym kształcie są widoczne **uchwyty** w postaci małych kwadratów. Jeśli wybierzemy kursorem myszy dowolny uchwyt, możemy edytować kształt. Uchwyty pozwalają na wykonanie następujących operacji edytorskich:

- przesuwanie kształtu w dowolne miejsce na obszarze roboczym;
- skalowanie kształtu;
- obracanie płynnie zaznaczonego obiektu;
- pochylanie wybranej krawędzi lub całego kształtu.

W obszarze roboczym można zaznaczać jeden wybrany obiekt lub grupę obiektów. Dodatkowo cechy obrazu wektorowego pozwalają na łączenie różnych obiektów w grupę. Grupa podczas przemieszczania czy przekształcania zachowuje się tak jak pojedynczy obiekt. Połączone obiekty pozwalają się w dowolny sposób rozdzielać, edytować i ponownie łączyć. W programie Inkscape są dostępne zaawansowane działania na obszarach ścieżek (tzw. kształtowanie) – suma (spawanie), różnica (przycięcie), część wspólna. Tak jak w aplikacjach do edycji grafiki rastrowej, obiekty możemy wypełniać kolorem lub edytować barwę konturu.

W edycji obrazów wektorowych korzysta się również z warstw. Pozwala to oddzielić poszczególne etapy pracy. Przydatnymi funkcjami są także powielanie obiektów oraz zmiana kolejności ich położenia. Istotną cechą aplikacji jest możliwość importowania obrazów rastrowych.

Autorzy stron internetowych wykorzystują edytor grafiki wektorowej do tworzenia banerów, przycisków, ikon lub innych elementów graficznej konstrukcji strony.

PRZYKŁAD 5.2.

Na każdej stronie internetowej w okresie świątecznym pojawiają się elementy graficzne tworzące odpowiedni nastój. Naszym zadaniem będzie wykonanie skromnego banera świątecznego za pomocą edytora Inkscape. Efekt naszej pracy musimy zapisać w pliku swieta.png. Tworzenie obrazu przebiega w sposób następujący:

Krok 1. Uruchamiamy program Inkscape. Na całej powierzchni roboczej rysujemy prostokąt z granatowym kolorem wypełnienia oraz delikatnym gradientem liniowym.

Krok 2. W górnej części prostokąta rysujemy wiele gwiazd o różnej wielkości w kolorze białym. Za pomocą uchwytów (węzłów) delikatnie modyfikujemy ich kształt.

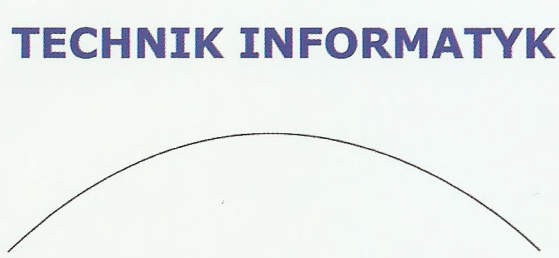
Krok 3. W dolnej części obrazu piórem rysujemy kontury pagórków, a w centralnej części choinkę. Za pomocą edycji węzłów kształtujemy kontury obiektów i wypełniamy innym odcieniem koloru niebieskiego. W górnej części rysujemy żółte koło i używając wypełnienia gradientem, kształtujemy półksiężyc.

Krok 4. Za pomocą narzędzia do wstawiania tekstu edytujemy tekst „Wesołych Świąt" – dobieramy artystyczną czcionkę, wielkość oraz odpowiedni kolor. Gotowy obraz zapisujemy w formacie PNG pod nazwą swieta.

Funkcją bardzo przydatną w tworzeniu obrazów wektorowych jest **opisywanie tekstu na krzywej**. Operacja ta pozwala na wykonanie ciekawych logotypów, ikon czy innych elementów strony.

PRZYKŁAD 5.3.

Aby kształtować tekst względem krzywej, należy wykonać następujące czynności:
Krok 1. Rysujemy linię, np. za pomocą pióra, i odpowiednio ją modelujemy, edytując węzły. Powyżej edytujemy tekst, jednocześnie nadając mu odpowiedni styl i kolor.



Krok 2. Kolejnym etapem pracy będzie zaznaczenie napisu oraz ścieżki i wybranie *Tekst* >> *Wstaw na ścieżkę*. Następnie możemy usunąć kolor konturu w oknie *Wypełnienie i kontur*. Efekt końcowy przedstawiono na rysunku.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Pobierz i zainstaluj program GIMP. Z pomocą nauczyciela i dokumentacji programu zapoznaj się z jego interfejsem i możliwościami. Skorzystaj z opcji *Plik* >> *Utwórz* i zaprojektuj przycisk menu, który będzie można wykorzystać do wysłania e-maila na dowolnej stronie internetowej. Najlepiej podczas eksportu wybierz format png z przezroczystym tłem. Efekt pracy zapisz pod nazwą przycisk.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co oznacza akronim GIMP?
2. Opisz działanie programu Inkscape.

5.3 Zasady obróbki obrazów cyfrowych

ZAGADNIENIA

- **Poprawianie jakości obrazów cyfrowych**
- **Zasady obróbki obrazów**
- **Wykonywanie zrzutów z ekranu**

Oprócz tworzenia obrazów cyfrowych w edytorach graficznych często wykonuje się obróbkę plików graficznych pochodzących ze skanera, kamery, aparatu cyfrowego lub zrzutu ekranowego. Obróbka tych materiałów sprowadza się do następujących operacji:

- poprawy parametrów obrazu, takich jak: naświetlenie, kolor, kontrast;
- kadrowania;
- skalowania;
- przekształcenia (obrót, odbicie);
- przekształcania geometrycznego.

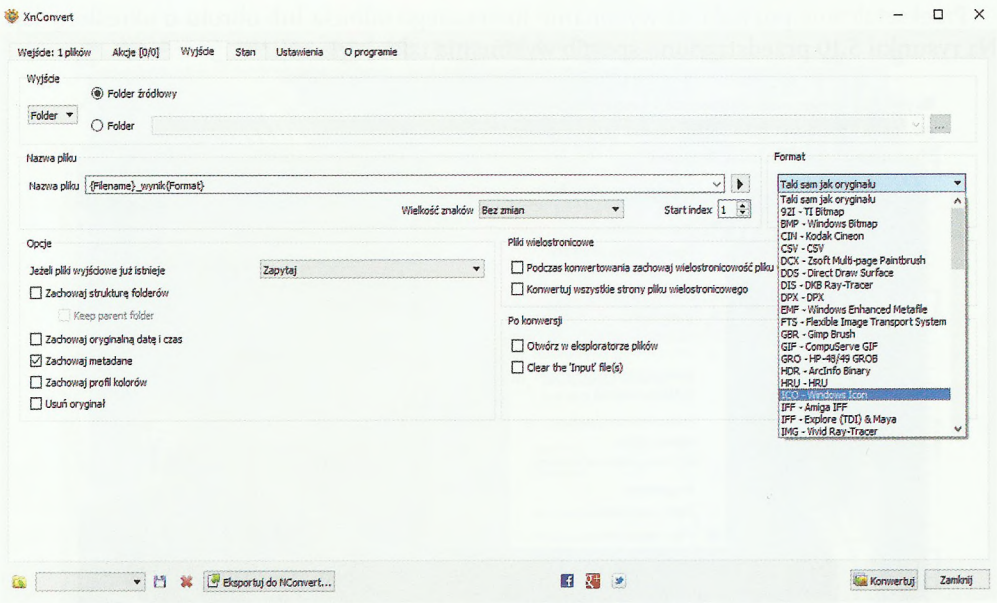
Dodatkowo wykonujemy konwersję między formatami zapisu pliku lub proste techniki poprawy zdjęć typu:

- redukcja czerwonych oczu,
- retusz,
- fotomontaż.

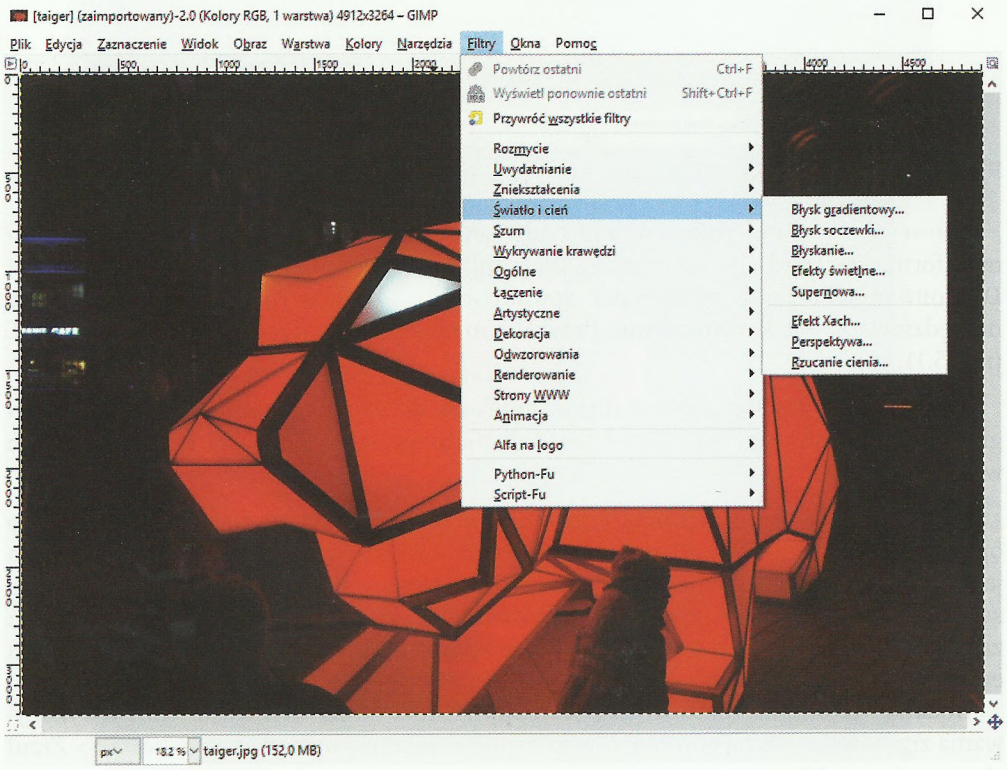
Często wykonywaną czynnością jest również tworzenie i obróbka zrzutów z ekranu.

Skalowanie to proporcjonalna zmiana rozmiarów obrazu względem oryginału. Skalowanie w programie GIMP można wykonać za pomocą opcji *Obraz* >> *Skaluj obraz*. Funkcja ta pozwala na dopasowanie rozmiaru obrazu do parametrów strony internetowej oraz zmiany rozdzielczości (rys. 5.8).

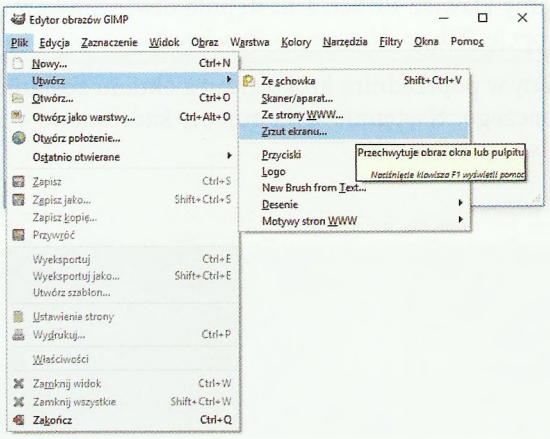
Kadrowanie to nic innego jak tylko przycięcie oryginalnego obrazu w celu uzyskania optymalnego kadru do wykorzystania jako oddzielny plik graficzny (rys. 5.9).



Rys. 5.11. Okno programu XnConvert



Rys. 5.12. Korzystanie z filtrów



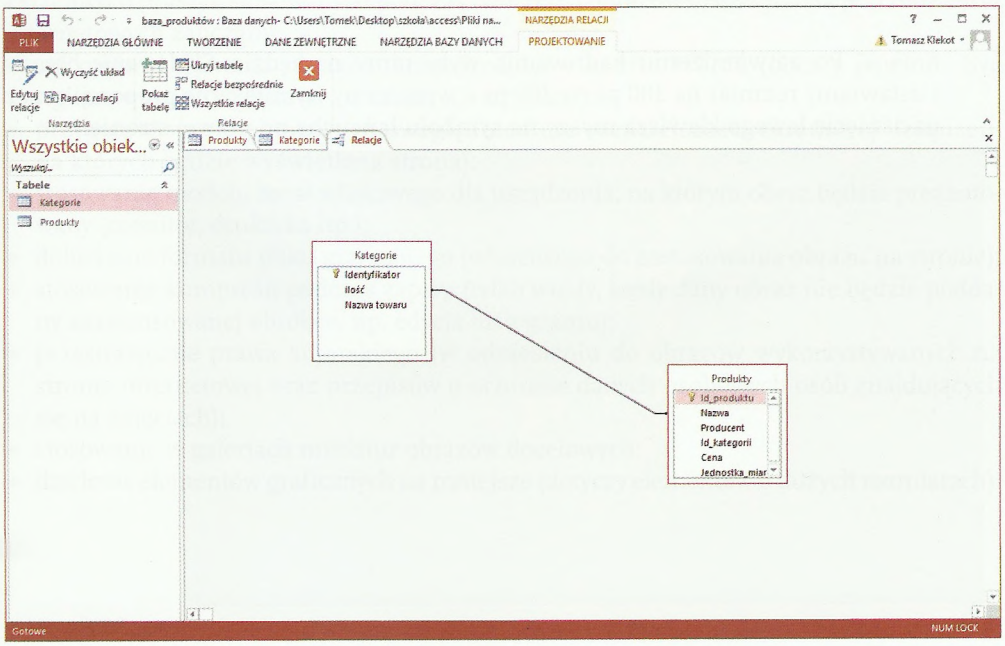
Rys. 5.13. Wykonywanie zrzutów ekranu w programie GIMP

Po utworzeniu zrzutu w programie GIMP lub wklejeniu zrzutu ze schowka systemowego możemy wprowadzić określone zmiany w rozdzielczości czy wielkości obrazu.

PRZYKŁAD 5.4.

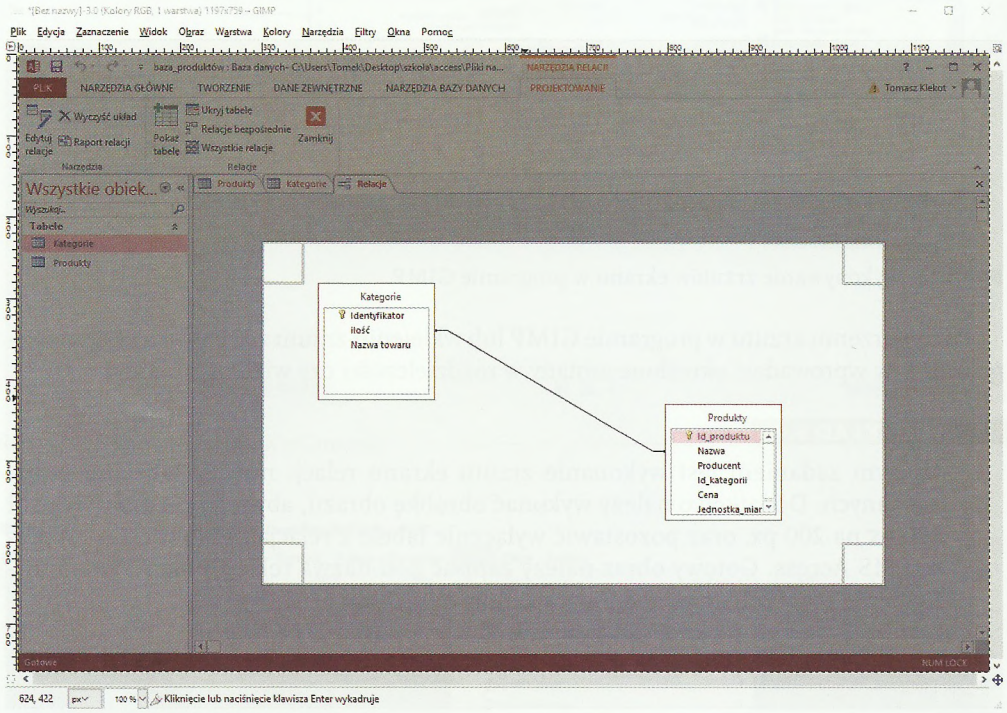
Naszym zadaniem jest wykonanie zrzutu ekranu relacji między tabelami w bazie danych. Dodatkowo należy wykonać obróbkę obrazu, aby jego rozmiar wynosił 300 px na 200 px, oraz pozostawić wyłącznie tabele z relacjami – bez ikon programu MS Access. Gotowy obraz należy zapisać pod nazwą relacje.png. Aby poprawnie wykonać to zadanie, kolejność czynności musi być następująca:

Krok 1. Otwieramy okno, które będzie źródłem zrzutu. W naszym przypadku będzie to okno relacji bazy danych w programie MS Access.

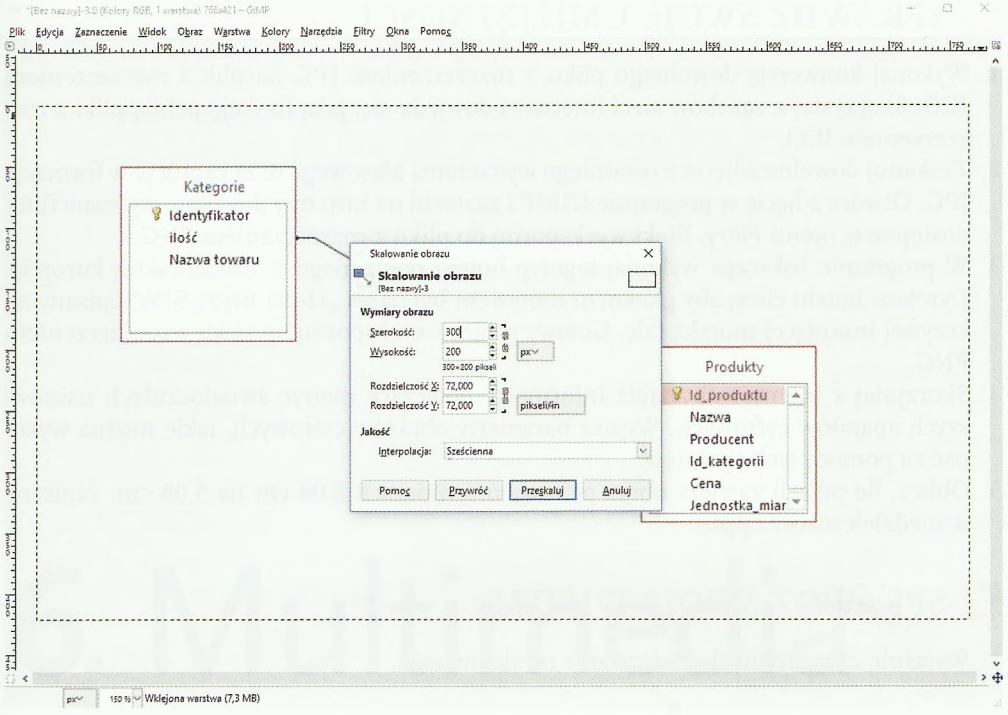


Krok 2. Naciskamy klawisz PrtScr, a następnie otwieramy program GIMP. W nim uruchamiamy opcję *Plik >> Utwórz >> Ze schowka*.

Krok 3. Wykonany w poprzednim kroku zrzut z ekranu zostanie wklejony do nowego obszaru roboczego. Następnie wykonujemy kadrowanie – korzystamy z narzędzia kadrowania w przyborniku.



Krok 4. Po zatwierdzeniu kadrowania wybieramy narzędzie *Skalowanie obrazu* i ustawiamy rozmiar na 300 px na 200 px – wyłączamy wymuszanie proporcji przez naciśnięcie lewego klawisza myszy na symbolu łańcucha po prawej stronie.



Krok 5. Po zatwierdzeniu skalowania gotowy obraz należy wyeksportować do pliku pod nazwą relacje.png.

Praktyczne wskazówki

Do dobrych praktyk stosowanych podczas tworzenia obrazów cyfrowych na potrzeby stron internetowych zaliczamy:

- unikanie skalowania obrazów bezpośrednio w kodzie HTML (obrazy powinny być wykonywane w rozmiarze, w jakim ich użyjemy);
- przygotowywanie obrazów (z uwzględnieniem rozdzielczości ekranowej urządzeń, na których będzie wyświetlana strona);
- stosowanie modelu barw właściwego dla urządzenia, na którym obraz będzie prezentowany (monitor, drukarka itp.);
- dobieranie formatu pliku graficznego (właściwego do zastosowania obrazu na stronie);
- stosowanie kompresji podczas zapisu (tylko wtedy, kiedy dany obraz nie będzie poddany zaawansowanej obróbce, np. edycja histogramu);
- przestrzeganie prawa autorskiego (w odniesieniu do obrazów wykorzystywanych na stronie internetowej oraz przepisów o ochronie danych osobowych osób znajdujących się na zdjęciach);
- stosowanie w galeriach miniatur obrazów docelowych;
- dzielenie elementów graficznych na mniejsze (dotyczy elementów o dużych rozmiarach).

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wykonaj konwersję dowolnego pliku z rozszerzeniem JPG na plik z rozszerzeniem ICO. Skorzystaj z zasobów sieci internet i dowiedz się, jaką funkcję pełnią pliki z rozszerzeniem ICO.
2. Zeskanuj dowolne zdjęcie z ostatniego wydarzenia klasowego oraz zapisz je w formacie JPG. Otwórz zdjęcie w programie GIMP i zastosuj na nim trzy dowolnie wybrane filtry dostępne w menu *Filtry*. Efekt wyeksportuj do pliku z rozszerzeniem PNG.
3. W programie Inkscape wykonaj logotyp hotelu położonego w nadmorskim kurorcie. Dyrektor hotelu chce, aby głównym motywem był napis „Hotel Bryza SPA" opisany na krzywej imitującej morską falę. Gotowy projekt wyeksportuj do pliku z rozszerzeniem PNG.
4. Skorzystaj z internetu i znajdź informacje dotyczące matryc światłoczułych najnowszych aparatów cyfrowych. Wypisz parametry obrazów cyfrowych, jakie można wykonać za pomocą tych aparatów.
5. Oblicz, ile pikseli zawiera obraz cyfrowy o wymiarach 5,08 cm na 5,08 cm, zapisany w rozdzielczości 72 ppi.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij, czym różni się kadrowanie od skalowania.

6. Multimedia na stronach internetowych

- Animacja na stronach internetowych
- Publikacja plików wideo na stronach internetowych
- Przygotowanie plików dźwiękowych

6.1 Animacja na stronach internetowych

ZAGADNIENIA
- **Animacja komputerowa – definicja**
- **Formaty plików z animacją**
- **Rodzaje animacji**
- **Przygotowanie animacji na potrzeby strony internetowej**

Animacja komputerowa

Animacja to tworzenie cyfrowych obrazów ruchomych z obrazów statycznych za pomocą zmiany położenia, parametrów wyświetlania, barwy, kształtu lub rozmiaru, odtwarzanych w określonym czasie.

Do podstawowych metod tworzenia animacji komputerowej zaliczamy:
- animację poklatkową, polegającą na tworzeniu poszczególnych ramek animacji po kolei, najczęściej w postaci mapy bitowej;
- metodę ramek kluczowych przez generowanie wybranych ramek (pozostałe są obliczane – interpolowane);
- skrypty, które opisują zmianę właściwości obiektów;
- animację interaktywną polegającą na tym, że obiekty są sterowane przez użytkownika.

Poszczególne obrazy w animacji zmieniają się z odpowiednią częstotliwością liczoną w **klatkach na sekundę** – **fps** (ang. *frames per second*). Animację uznajemy za płynną wtedy, gdy częstotliwość wynosi minimum 12 fps. Standardem powszechnie uznanym jest zakres wielkości od 24 do 35 fps.

Do podstawowych formatów plików, w których zapisujemy dynamiczną grafikę, zaliczamy:
- **GIF** (ang. *Graphics Interchange Format*) – popularny format internetowej grafiki statycznej i dynamicznej, obsługiwany przez większość przeglądarek internetowych;
- **SWF** – zamknięty format grafiki wektorowej przeznaczony do aplikacji Adobe Flash umożliwiający zapisywanie i odtwarzanie animacji oraz całych struktur stron internetowych.

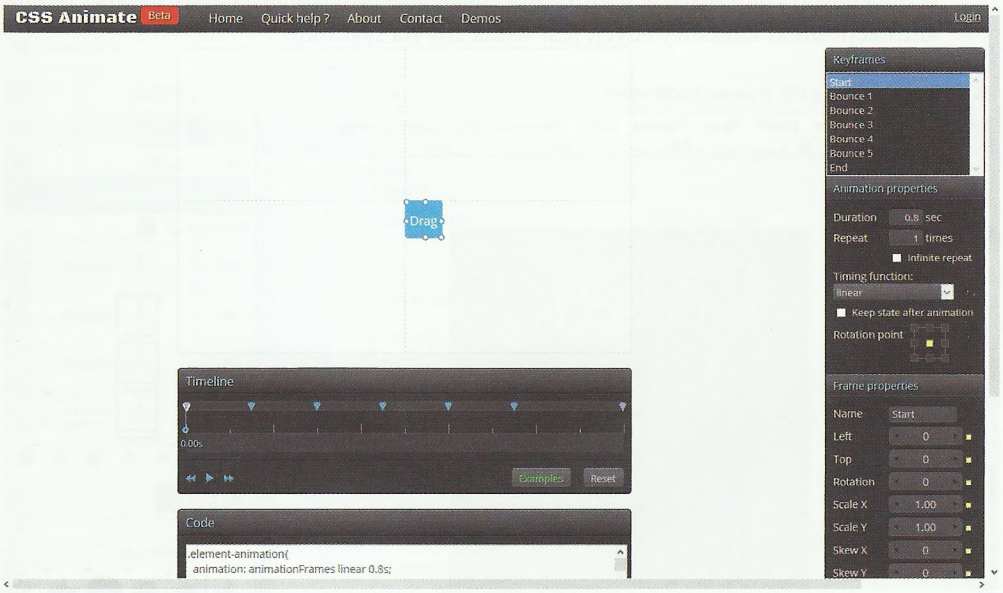
Animacja z poziomu kodu

Animację elementów dokumentów hipertekstowych umożliwiają skrypty CSS, wykorzystujące na przykład właściwość animation oraz reguły @keyframes. Na rysunku 6.1 pokazano przykład wykorzystania CSS do płynnej zmiany koloru tła w określonym czasie.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  animation: pulse 5s infinite;
}

@keyframes pulse {
  0% {
    background-color: #3322FF;
  }
  100% {
    background-color: #FF1245;
  }
}
</style>
</head>
<body>
<div>Przykład animacji poprzez zmianę koloru tła</div>
</body>
</html>
```

Rys. 6.1. Przykład kodu umożliwiającego płynną zmianę koloru tła pod tekstem

Dostępnych jest coraz więcej aplikacji internetowych służących do budowania animacji z wykorzystaniem arkuszy stylów CSS3. Przykładem może być strona **http://cssanimate.com/**, za pomocą której możemy utworzyć prostą animację na podstawie kilku udostępnionych przykładów oraz skopiować wygenerowany kod CSS.



Rys. 6.2. Okno cssanimate.com

Z powodzeniem można też tworzyć animację skryptową na poziomie skryptów JavaScript, ponieważ pozwalają one tworzyć wrażenie animacji poklatkowej.

Animowane obrazy GIF

Animowane GIF-y to najpopularniejszy i najprostszy sposób tworzenia animacji poklatkowej. Format ten może przechowywać wiele obrazków w jednym pliku i tworzyć z nich animację. Dodatkowo GIF obsługuje przeźroczystość monochromatyczną, co może mieć zastosowanie w graficznym projektowaniu stron internetowych.

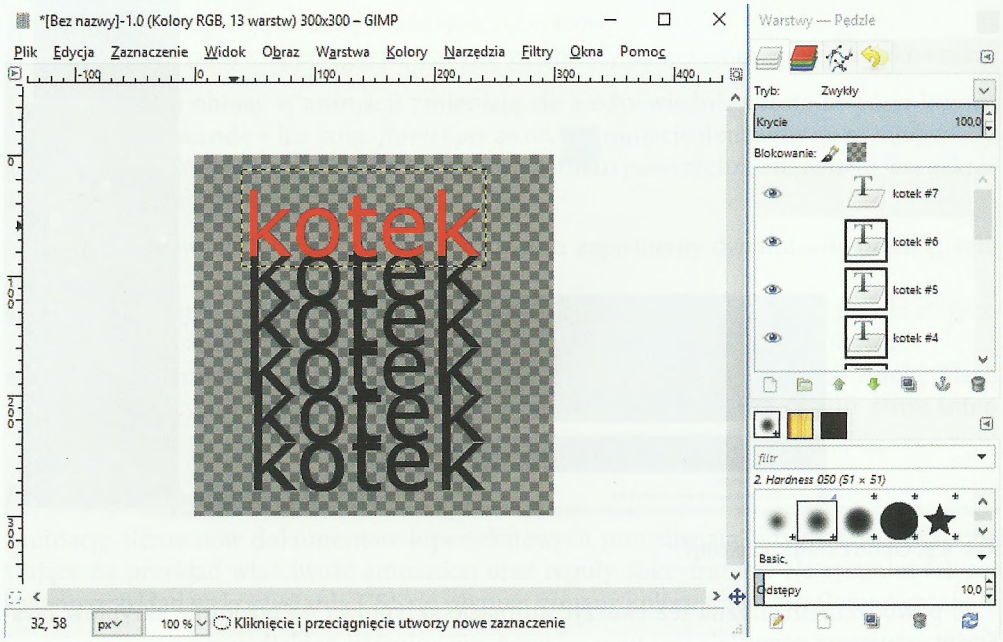
Do stworzenia animowanych GIF-ów możemy wykorzystać liczne oprogramowanie, np. GIF Animator, Free GIF Maker czy QGifer. Można również prostą animację poklatkową wykonać za pomocą popularnych edytorów graficznych, jak np. GIMP.

PRZYKŁAD 6.1.

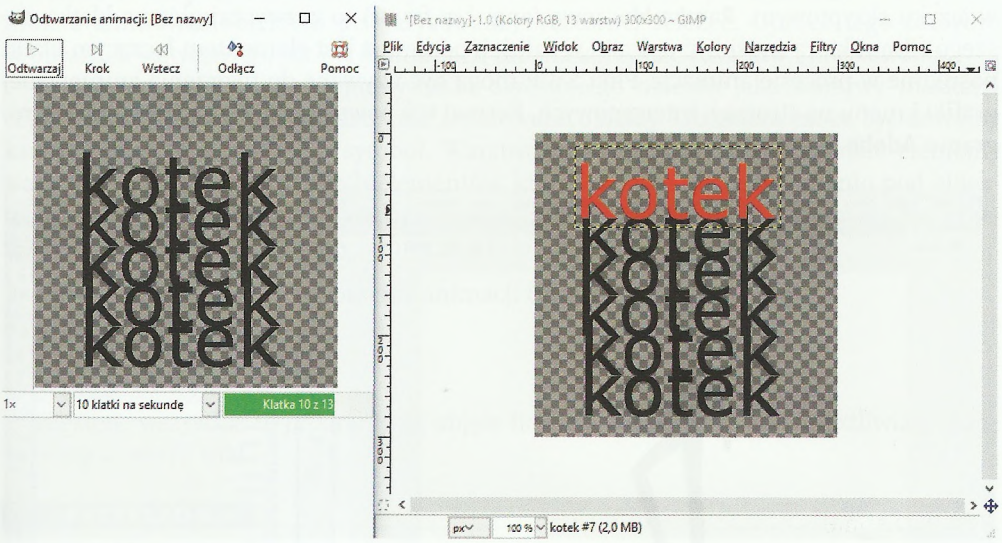
Animację poklatkową wykonujemy poprzez powielenie początkowego obrazu statycznego oraz wykonanie modyfikacji na zdublowanych warstwach. Przykładową animację w programie GIMP tworzymy w następujący sposób:

Krok 1. Tworzymy nowy dokument – ustawiamy założone rozmiary obrazu. Ponieważ tworzoną animację wykorzystamy jako logo na stronie mającej własne tło, wybieramy dla naszego obrazu przezroczystość.

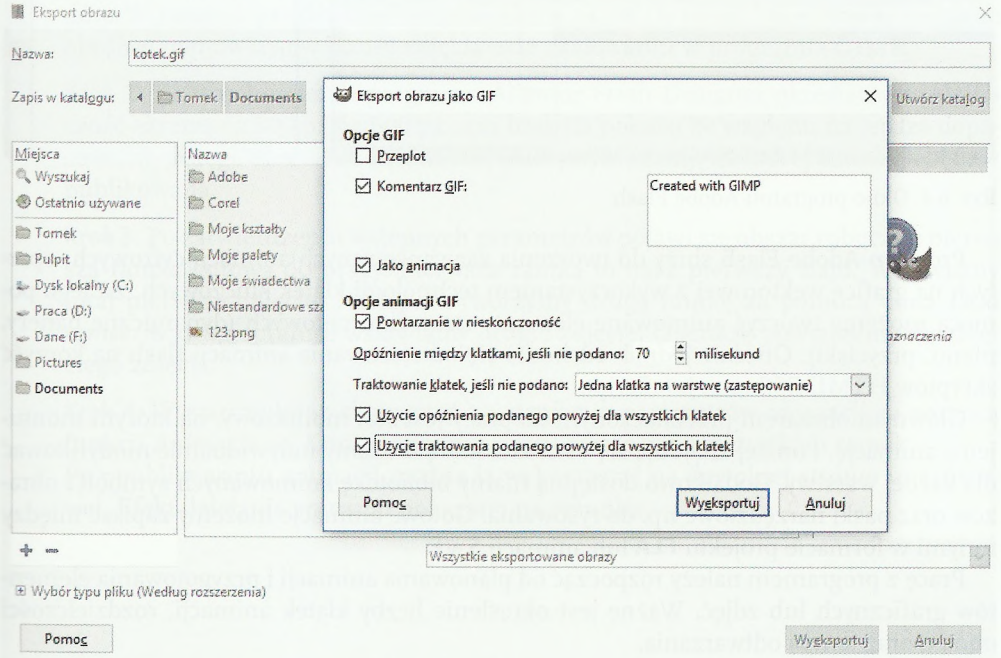
Krok 2. Po utworzeniu nowego obrazu tworzymy napis kotek. Piszemy literę k, następnie tworzymy duplikat warstwy za pomocą okna warstw. Powtarzamy tę czynność aż do momentu, kiedy powstanie cały napis. Tą samą metodą kopiujemy napis linia po linii do góry, a w ostatniej linii zmieniamy kolor napisu. Pamiętamy cały czas o wykonywaniu duplikatów warstw po każdej edycji napisu. Powstaje w ten sposób zbiór warstw, z których każda jest traktowana jako jedna klatka naszej animacji. Efekt końcowy edycji jest widoczny na rysunku.



Krok 3. Po zakończeniu edycji możemy uruchomić podgląd animacji za pomocą polecenia *Filtry >> Animacja >> Odtwarzanie.*



Krok 4. Następnie musimy wyeksportować obraz w formie animacji. W oknie wyeksportowania podajemy nazwę kotek.gif i ustawiamy opcje odtwarzania animacji. Przykład tej czynności pokazano na rysunku.

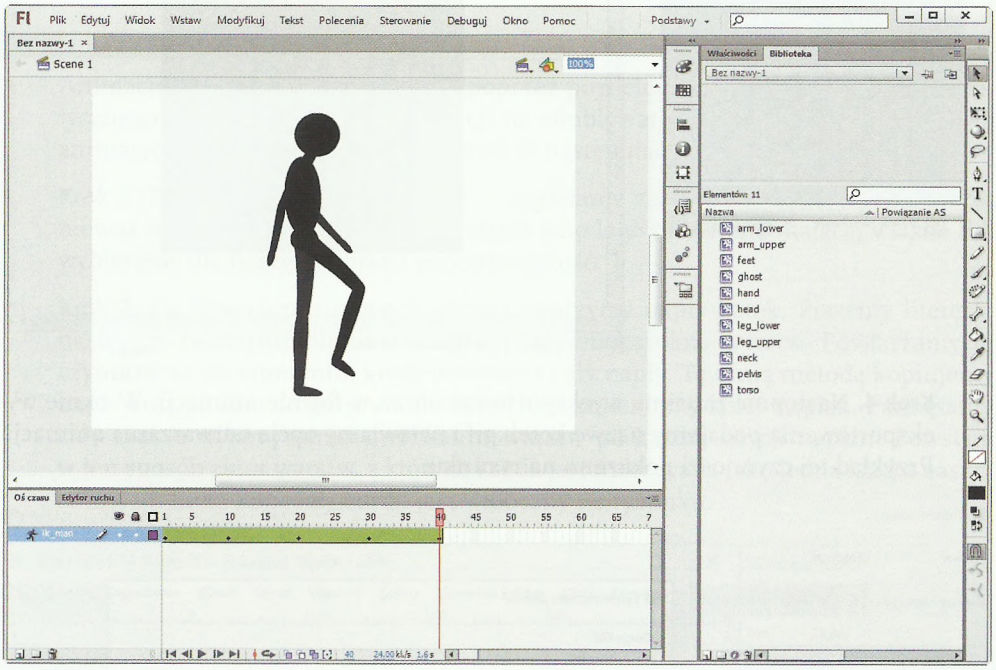


Rys. 6.3. Ustawienie odtwarzania animacji

Po zapisaniu obraz można odtworzyć w dowolnej przeglądarce.

Animacja SWF

Pliki SWF mogą zawierać animacje wykorzystujące metodę ramek kluczowych lub aplety w języku skryptowym. **Ramka kluczowa** (ang. *key frame*) to zazwyczaj główna klatka naszego filmu, która musi być opisana w całości, ponieważ jest elementem łączącym klatki pośrednie w procesie animacji. Pliki SWF mogą być używane do tworzenia animowanej grafiki i menu na stronach internetowych. Format ten powstał jako format zamknięty programu **Adobe Flash**.



Rys. 6.4. Okno programu Adobe Flash

Program Adobe Flash służy do tworzenia zaawansowanych animacji cyfrowych, opartych na grafice wektorowej z wykorzystaniem technologii klatek kluczowych. Za jego pomocą możemy tworzyć animowane elementy stron internetowych (dynamiczne banery, menu, przyciski). Obecnie jednak odchodzi się od stosowania animacji flash na korzyść skryptów HTML-CSS oraz JavaScript.

Głównym obszarem przeznaczonym do pracy jest stół montażowy, na którym montujemy animację. Poniżej znajduje się oś czasu, którą możemy indywidualnie modyfikować dla każdej warstwy. Dodatkowo dostępną mamy bibliotekę animowanych symboli i obrazów oraz paski narzędziowe np. do rysowania. Gotowe animacje możemy zapisać między innymi w formacie projektu **FLA** lub animacji **SWF**.

Pracę z programem należy rozpocząć od planowania animacji i przygotowania elementów graficznych lub zdjęć. Ważne jest określenie liczby klatek animacji, rozdzielczości obrazu oraz tempa odtwarzania.

Dzięki dostępności narzędzi do tworzenia grafiki wektorowej oraz edycji tekstu większość potrzebnych elementów animacji można wykonać w programie. Na koniec pracy można dodać dźwięk i opublikować gotową animację do formatu SWF, GIF lub samouruchamiających się plików EXE.

Adobe Flash dysponuje również obiektowym językiem skryptowym do sterowania animacjami **Action Script**. Skrypty można tworzyć w trakcie budowy animacji.

Adobe Flash pozwala na wykorzystanie w animacji **ruchu po ścieżce**. W ten sposób można animować obiektem po narysowanej wcześniej ścieżce. Oprócz tego zarówno w obrazach statycznych, jak i w animacjach można korzystać z warstw i **maskowania**.

Maskowanie polega na wykorzystaniu określonego obiektu jako warstwy maskującej, przez którą możemy oglądać inne animowane obiekty. Obiektem może być dowolny kształt wypełniony, tekst lub symbol. Warstwa maskująca zasłania wszystkie elementy warstw pod spodem oprócz tych elementów, które znajdują się bezpośrednio pod obiektem maskującym. Obiekt wyświetlany działający jako warstwa maskująca może być przemieszczany, animowany lub dynamicznie skalowany.

Innymi programami do tworzenia animacji flash są:
- Aligator Flash Designer,
- SwiSH Max,
- Sothink SWF Quicker.

Niestety, wszystkie te programy są objęte licencją komercyjną, ale umożliwiają pracę również w wersji trial.

PRZYKŁAD 6.2.

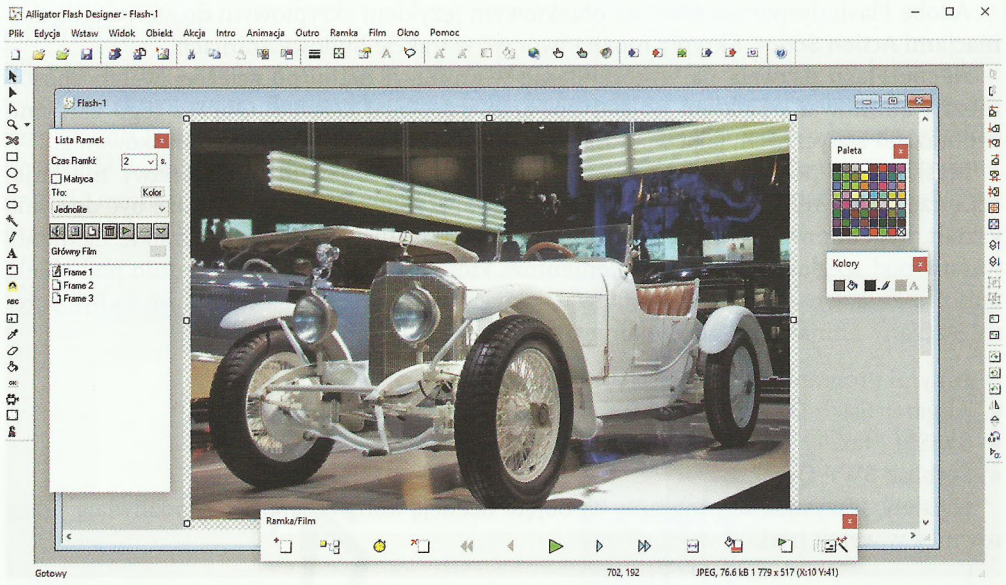
Przykład wykorzystania animacji flash pokażemy, wykorzystując program Aligator Flash Designer. Utworzymy prosty pokaz slajdów do wykorzystania na stronie internetowej, według następujących kroków:

Krok 1. Na wstępie musimy wyselekcjonować i przygotować zdjęcia do pokazu slajdów. W naszym przykładzie będą to pliki o nazwach 1.jpg, 2.jpg i 3.jpg, poddane obróbce poprawiającej jakość zdjęcia oraz skalowaniu w programie GIMP.

Krok 2. Po uruchomieniu programu Aligator Flash Designer określamy rozdzielczość ekranu na 800 px na 600 px oraz brak tła pokazu ze względu na lepsze dopasowanie takiego obrazu do konkretnej kolorystyki strony, na której animacja będzie publikowana.

Krok 3. Po zatwierdzeniu wstępnych parametrów pojawi się obszar roboczy z pierwszą ramką gotową do pracy. Pierwsza ramka to nasz pierwszy slajd. Wstawiamy tutaj pierwsze zdjęcie. Następnie tworzymy drugą ramkę za pomocą okna *Lista ramek*. W drugiej ramce wstawiamy drugie zdjęcie. Operację powtarzamy dla trzeciego zdjęcia.

Krok 4. Wystarczy teraz skorzystać z ustawienia parametrów animacji za pomocą funkcji *Animacja* >> *Kreator animacji* i zastosować je do wszystkich ramek.
Po opublikowaniu animacji można ją wykorzystać na dowolnej stronie internetowej. Efekt działania animacji pokazano na rysunku.



Rys. 6.5. Przygotowanie pokazu slajdów

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień metody tworzenia animacji.
2. Scharakteryzuj animację GIF.
3. Na czym polega różnica pomiędzy animacją GIF a animacją SWF?

6.2 Publikacja plików wideo na stronach internetowych

ZAGADNIENIA

- Metody odtwarzania filmów na stronach WWW
- Standardy obrazu
- Formaty plików wideo

Film na stronie internetowej to już nie ekstrawagancja, lecz konieczność i wymóg marketingu sieciowego.

Oprócz wstawiania plików multimedialnych z poziomu kodu HTML (znaczniki **<video>**, **<embed>**), odtwarzanych przez narzędzia przeglądarki, często wykorzystuje się również specjalnie instalowane **kodeki**. Plik wideo jest zazwyczaj skompresowanym formatem zawierającym wiele elementów niezbędnych do odtworzenia filmu. Kodeki pozwalają na sprawne przeprowadzenie procesu kodowania lub dekodowania pliku multimedialnego. Do popularnych kodeków zaliczamy: Quick Time Player, Adobe Flash Player czy Microsoft Silverlight.

Podczas ładowania pliku o określonym formacie lepiej jest wskazać kodek, w którym chcemy odtworzyć wideo. Można to zrobić, stosując atrybut **pluginspage = "adres URL kodeka"**. Wtedy umożliwia się pobranie i zainstalowanie określonego kodeka.

PRZYKŁAD 6.3.

```
<embed type="application/x-mplayer2" src="film.wmv"
„pluginspage="http://www.microsoft.com/Windows/MediaPlay-
er/">FILM</embed>
```

Można również za pomocą atrybutu codecs = „nazwa kodeka" wskazać wtyczkę, która posłuży jako odtwarzacz danego pliku.

PRZYKŁAD 6.4.

```
<source src="film.mp4" type='video/mp4; codecs="mp4v.20.8,
mp4a.40.2"'>
```

Oprócz osadzania plików wideo z zasobów własnego serwera można wyświetlić film opublikowany na serwerach publicznych. W ten sposób wstawiamy filmy, np. z serwisu YouTube, o czym pisaliśmy w rozdziale o języku HTML.

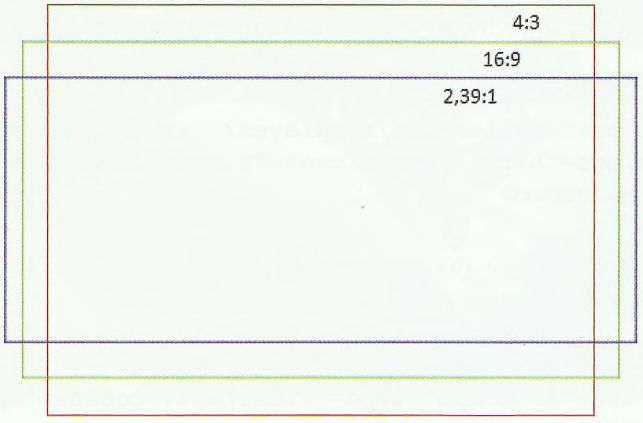
Żeby jednak odtworzyć film na stronie internetowej, należy wcześniej przygotować odpowiedni materiał wideo. Za przechowywanie materiałów wideo w postaci cyfrowej są odpowiedzialne odpowiednie formaty plików. Do popularnych formatów plików wideo zaliczamy:

- AVI (standardowy format Windows, występujący w postaci skompresowanej lub bez kompresji);
- MPEG (format stratnej kompresji obrazu wideo);
- WMV (format kompresji filmów opracowany przez firmę Microsoft);
- FLV (kontenerowy format wideo stworzony przez firmę Adobe; format ten wykorzystują takie serwisy, jak YouTube czy Google Video; zapewnia on pełną integrację z projektami Flash);
- Ogg (format kontenera strumieni danych wideo, stworzony, by wspierać inicjatywy związane z rozwojem wolnego oprogramowania do kodowania i dekodowania multimediów);
- MOV (format plików wideo firmy Apple).

Dodatkowym parametrem istotnym w obróbce filmów jest **standard jakości obrazu**. Najczęściej stosowanymi standardami jakości obrazu wideo są:

- VHS (ang. *Video Home System*) – opracowany w 1976 roku przez firmę JVC z rozdzielczością poziomą 240 linii;
- PAL (ang. *Phase Alternating Line*) – stosowany w Europie z rozdzielczością 704 x 576 px;
- NTSC (ang. *National Television System Committee*) – stosowany w Stanach Zjednoczonych, wyświetlający obraz z rozdzielczością 720 x 480 px;
- HDTV (ang. *High Definition TeleVision*) – z rozdzielczością 1280 x 720 px;
- HDV (ang. *High Definition Video*) – z rozdzielczością 1440 x 1080 px;
- HD (ang. *High Definition*) – z rozdzielczością 1920 x 1080 px;
- UHDTV (ang. *Ultra High Definition Video*) – z rozdzielczością 3840 x 2160 px.

Bezpośrednio związany ze standardem jakości obrazu i jego rozdzielczością jest **format obrazu**. Do najczęściej stosowanych obecnie formatów obrazów zaliczamy 4:3 oraz 16:9.



Rys. 6.6. Różnice między formatami obrazów

Do najczęściej wykonywanych operacji związanych z przygotowaniem plików wideo na strony internetowe zaliczamy:

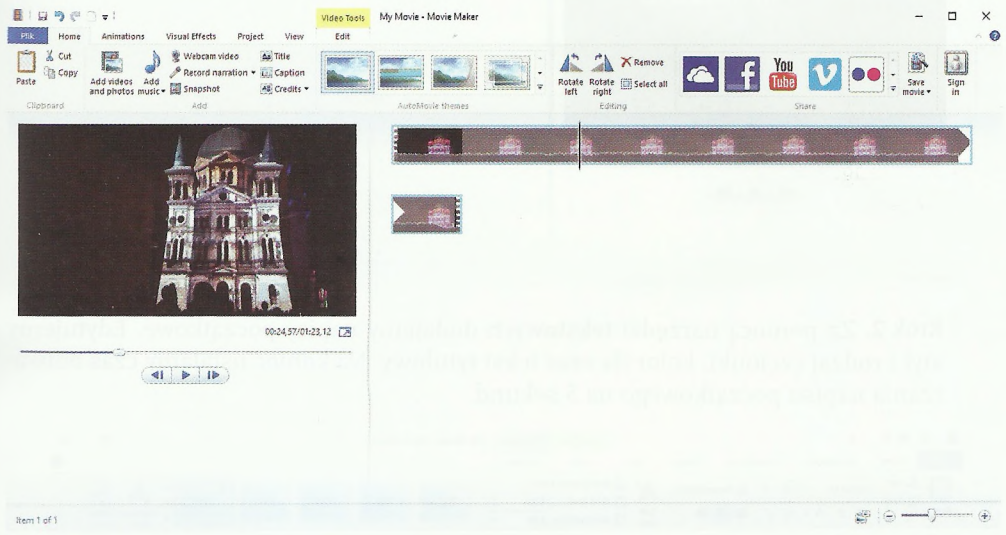
- przechwytywanie (ang. *capture*) obrazu wideo z pulpitu lub okna aplikacji;
- konwersję między formatami plików wideo;
- tworzenie krótkich filmów cyfrowych;
- montaż i edycję plików wideo;
- publikowanie plików wideo na stronach internetowych.

Wśród programów do edytowania plików wideo wyróżnić należy: Corel VideoStudio, Adobe Premiere, Movavi Video Editor lub Windows Movie Maker.

Opiszemy w tym rozdziale aplikację Windows Movie Maker, której plik instalacyjny można pobrać ze strony `http://www.windows-movie-maker.org`.

Program Windows Movie Maker jest dodatkowym składnikiem systemów MS Windows. Aplikacja oferuje wiele opcji edycyjnych, jak np. przyśpieszanie lub spowalnianie nagrań, dodawanie płynnych przejść między jednym a drugim fragmentem czy wstawianie napisów. Doskonale nadaje się do tworzenia prostych filmów, które mogą być publikowane na stronach internetowych.

Po instalacji programu i jego uruchomieniu otworzy się okno robocze z prostym i czytelnym interfejsem użytkownika. Dodanie filmu lub zdjęcia jest możliwe za pomocą opcji *Dodaj wideo i zdjęcia* (ang. *Add videos and photos*). Materiał źródłowy możemy przechwytywać również z kamery internetowej lub kamery telewizji analogowej.



Rys. 6.7. Okno programu Windows Movie Maker

Podczas importu plików źródłowych należy pamiętać, że aplikacja nie obsługuje takich formatów jak MP4 i MOV. Dlatego przed importem tego typu plików należy je przekonwertować na inny format.

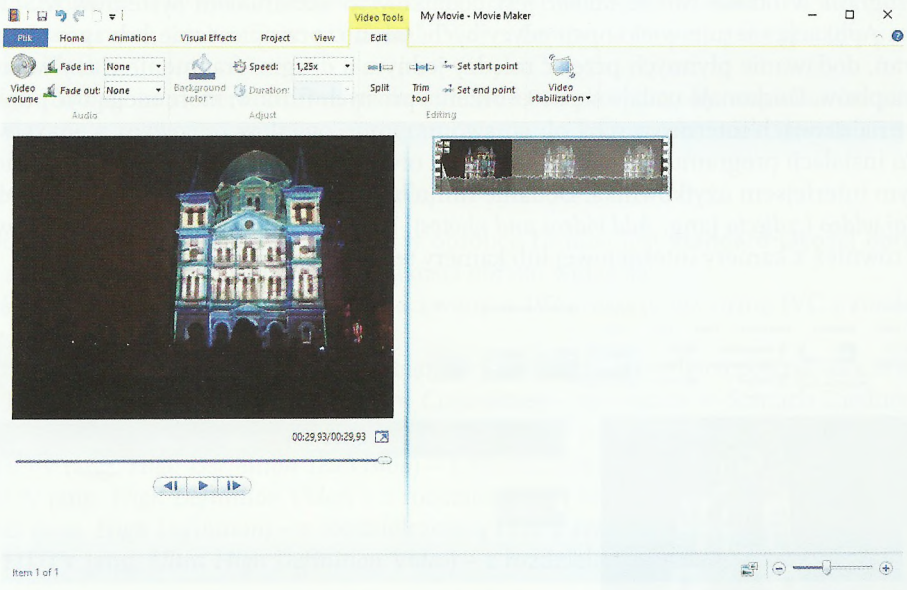
Zaimportowane obiekty można ułożyć w dowolnej kolejności. Za pomocą narzędzi wideo można dodać napisy początkowe i końcowe, zastosować efekty przejść albo obrót zaznaczonych fragmentów czy dostosować szybkość odtwarzania. Jeżeli film zawiera zbędne kadry, można przyciąć określony zakres klatek. Program umożliwia również dodanie podkładu muzycznego lub głosu lektora.

Na koniec utworzony film można opublikować bezpośrednio w serwisach społecznościowych lub zapisać w wybranym formacie.

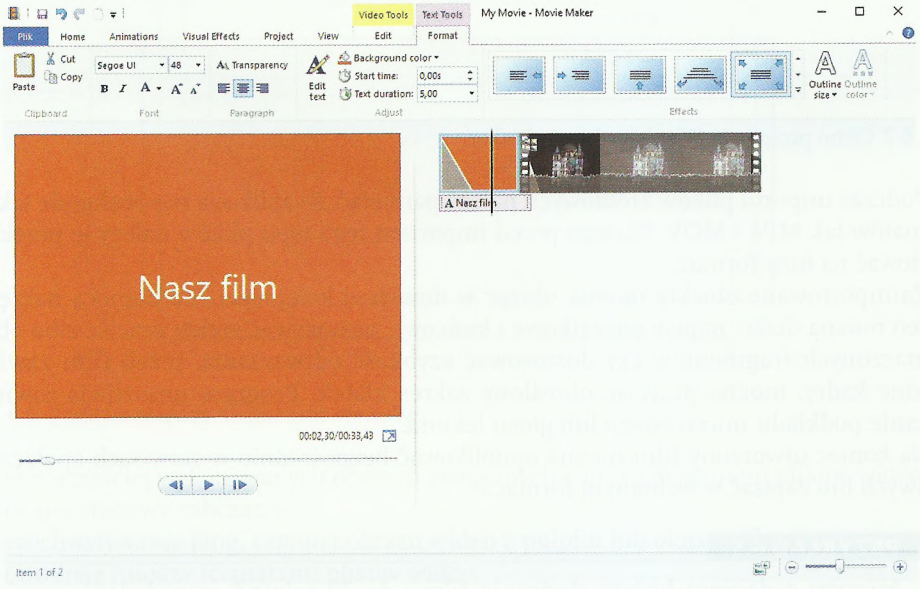
PRZYKŁAD 6.5.

Naszym zadaniem będzie skrócenie filmu do 30 sekund oraz dodanie napisów początkowych trwających 5 sekund. Gotowy plik należy zapisać w pliku film.wmv. Kolejność czynności jest następująca:

Krok 1. Otwieramy program i wczytujemy plik z materiałem źródłowym. Następnie za pomocą zakładki *Edycja* ustalamy punkt początkowy i końcowy filmu na poziomie 30 sekund odtwarzania. Po wskazaniu końcowego punktu film zostanie automatycznie docięty.



Krok 2. Za pomocą narzędzi tekstowych dodajemy napisy początkowe. Edytujemy styl i rodzaj czcionki, kolor tła oraz tekst tytułowy. Na koniec ustalamy czas odtwarzania napisu początkowego na 5 sekund.



Krok 3. Końcowy efekt zapisujemy pod nazwą film.wmv.

SPRAWDŹ SWOJĄ WIEDZĘ

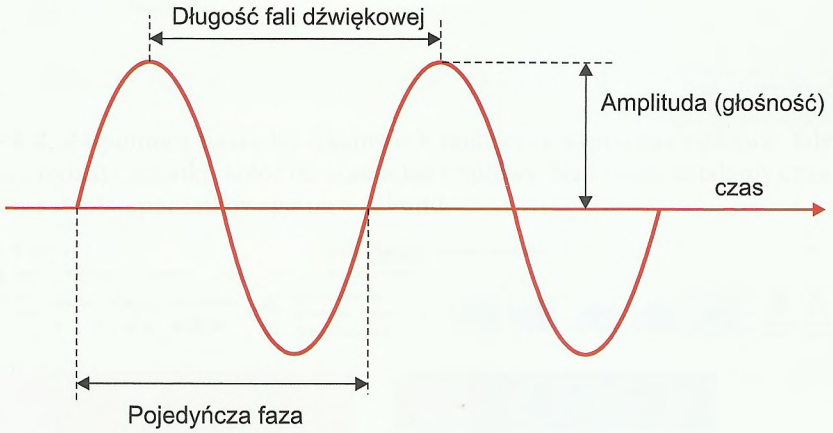
1. Do czego służą kodeki?
2. Wymień najpopularniejsze formaty plików wideo.
3. Opisz najczęściej stosowane standardy jakości obrazu wideo.
4. Jakie operacje dotyczące plików wideo są najczęściej wykonywane?

6.3 Przygotowanie plików dźwiękowych

ZAGADNIENIA

- **Definicja dźwięku**
- **Formaty plików dźwiękowych**
- **Edycja ścieżki dźwiękowej**

Dźwięk jest to wrażenie słuchowe wywołane przez falę akustyczną rozchodzącą się w wybranym środowisku (np. w powietrzu). Dźwięk ma swoją długość fali, częstotliwość i intensywność. Ludzki narząd słuchu wyłapuje dźwięki o częstotliwości fal z przedziału od 16 Hz do 20 kHz.



Rys. 6.8. Wykres fali dźwiękowej

Dźwięk można zapisać w formie cyfrowej. Odwzorowanie fali następuje poprzez wykonywanie pomiarów jej natężenia wiele razy w ciągu sekundy. Takie pomiary nazywają się próbkowaniem (ang. *sampling*).

Do podstawowych parametrów dźwięku cyfrowego zaliczamy:

- częstotliwość próbkowania sygnału dźwiękowego – ma ona bezpośredni związek z maksymalną częstotliwością dźwięku, jaką można zarejestrować i odtworzyć;
- rozdzielczość bitową dźwięku;
- przepływność danych (*Bitrate*), czyli ilość informacji (mierzoną w kilobitach) przesyłanych w jednostce czasu.

W procesie przetwarzania dźwięku analogowego na cyfrowy możemy wyróżnić następujące etapy:

- próbkowanie – polegające na prowadzeniu z określoną częstością pomiarów amplitudy sygnału dźwiękowego;
- kwantyzację – przyporządkowanie każdej wartości analogowej z próbkowania sygnału wartości dyskretnych z określonego zbioru;
- kodowanie – zapisanie otrzymanych kodów w formacie cyfrowym, zrozumiałym dla komputera;
- kompresję – zmiana zapisu danych dotyczących sygnału dźwiękowego prowadzącą do zmniejszenia rozmiaru pliku (kompresja stratna – usuwająca mniej istotne informacje, oraz kompresja bezstratna);
- konwersję – zmianę formatu zapisu danych sygnału dźwiękowego.

Jeżeli chcemy wykorzystać dźwięki na stronie internetowej, to musimy przygotować określone pliki dźwiękowe. W zależności od zastosowanej metody osadzenia dźwięku oraz rodzaju przeglądarki internetowej wybieramy odpowiedni format pliku. Do najpopularniejszych formatów plików dźwiękowych zaliczamy:

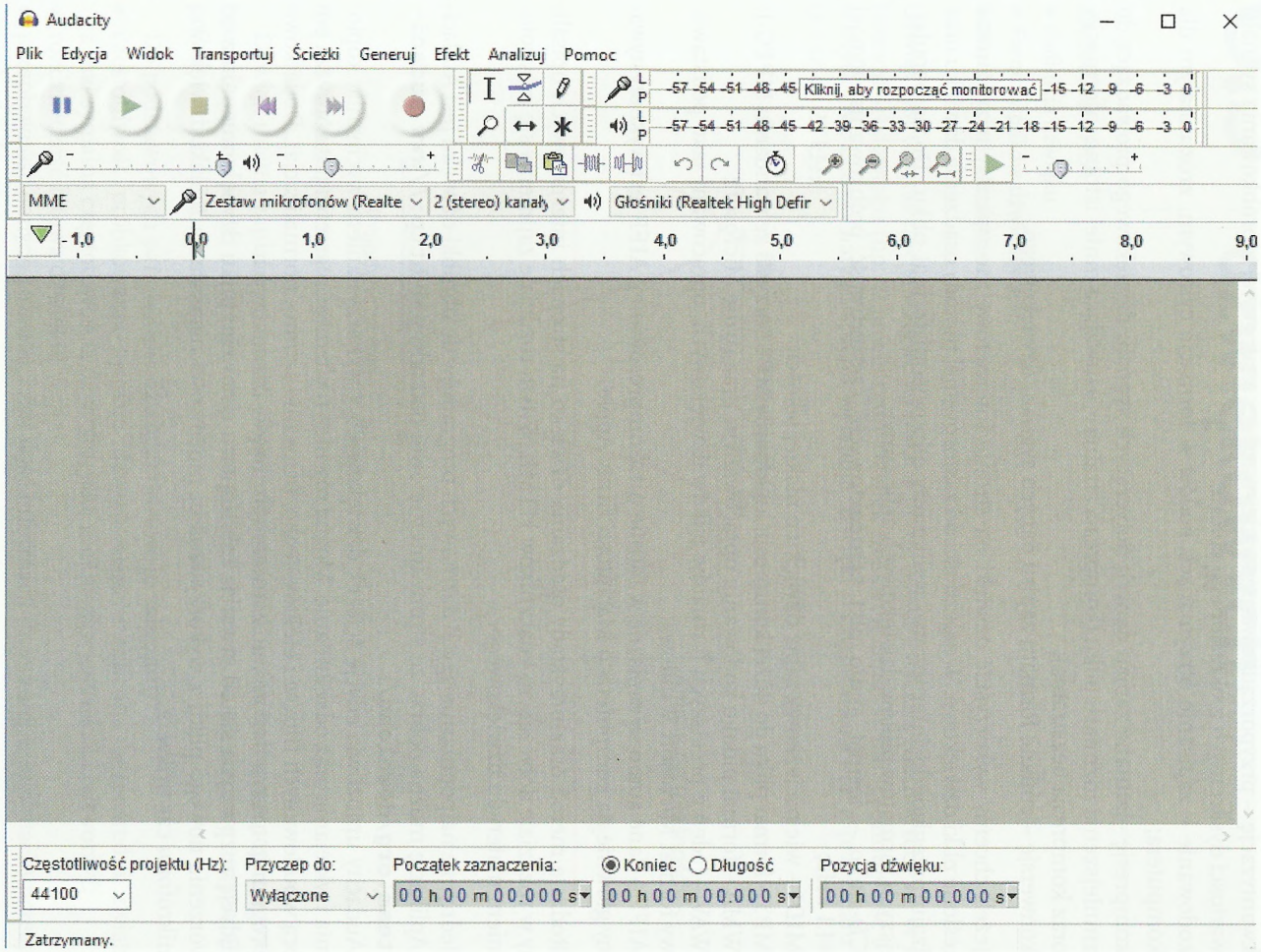
- CD-audio – format audio płyt kompaktowych; częstotliwość próbkowania wynosi 44.1 kHz;
- MIDI – wielościeżkowy zapis dźwięku o niskiej jakości;
- MP3 – format pliku do dużej kompresji; kodowanie MP3 korzysta ze standardu MPEG-I;
- WMA – format pliku o kodowaniu opracowanym praz firmę Microsoft;
- WAV – jeden z prostszych formatów plików dźwiękowych, opracowany jako podstawowy format Microsoft Windows;
- AIFF – dane audio są zapisane w formie nieskompresowanej; AIFF jest wykorzystywany głównie na komputerach Macintosh firmy Apple.

Rodzaje metod oraz przykłady osadzania dźwięku na stronach internetowych opisaliśmy w rozdziale dotyczącym znaczników HTML. W tym rozdziale zajmiemy się przygotowaniem materiału dźwiękowego.

Jednym z najpopularniejszych darmowych programów do obróbki dźwięku jest Audacity. Aplikację można pobrać ze strony internetowej pod adresem **`http://www.audacityteam.org/download/`**.

Audacity umożliwia edycję ścieżek dźwiękowych z wykorzystaniem wycinania, kopiowania i wklejania oraz odszumianie. Można również realizować mastering utworów za pomocą wbudowanych filtrów i efektów (echo, usuwanie szumu, normalizacja). Dodatkowo w programie można dokonywać konwersji formatów dźwiękowych.

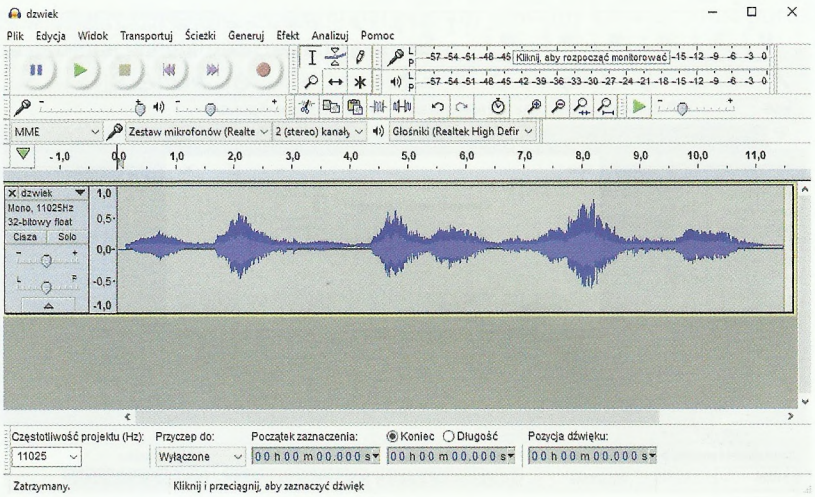
Edycję rozpoczyna się od otwarcia lub importu gotowego pliku dźwiękowego oraz od stworzenia nowego pliku z wykorzystaniem możliwości nagrywania ścieżki dźwiękowej z mikrofonu czy miksera.



Rys. 6.9. Okno programu Audacity

PRZYKŁAD 6.6.

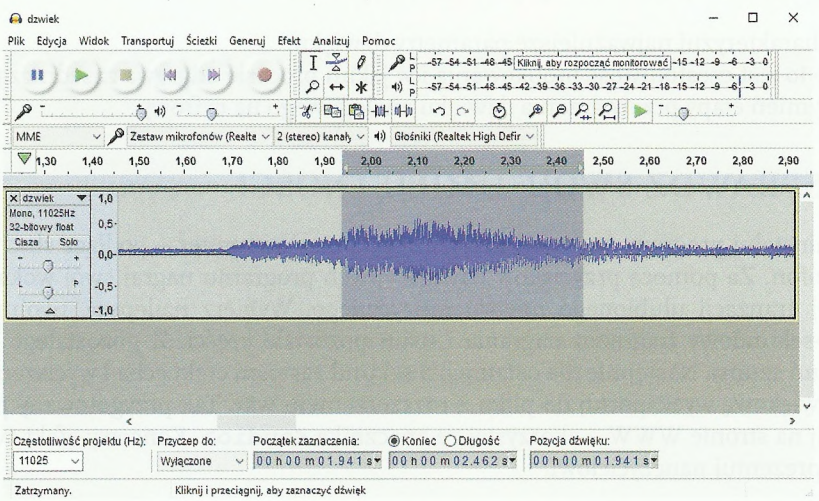
Aby wykonać obróbkę dźwięku przeznaczonego do wykorzystania na stronie internetowej, na początek wczytujemy określony plik źródłowy za pomocą polecenia *Otwórz* z menu *Plik*. Po wczytaniu pliku w oknie programu pojawi się przebieg czasowy dźwięku, który będzie poddany obróbce.



Rys. 6.10. Wczytany plik dźwiękowy

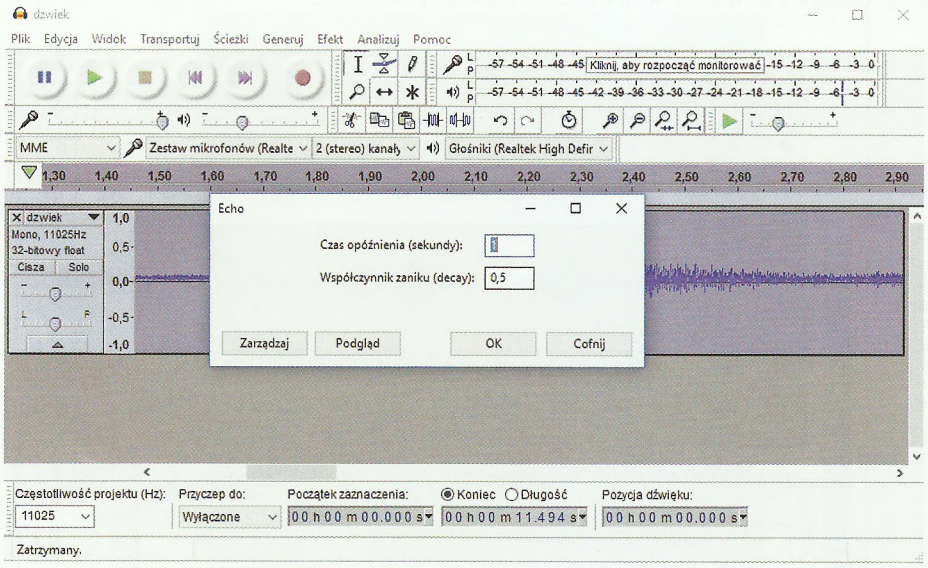
W górnym lewym rogu okna programu znajdują się przyciski sterujące odtwarzaniem dźwięku. Nad nimi jest widoczne menu główne programu. Osią czasu można sterować, dzięki czemu łatwiej będzie pracować nad poszczególnymi elementami utworu.

Do edycji fragmentu utworu są używane narzędzia zaznaczania znajdujące się po lewej stronie przycisku nagrywania. Zaznaczony fragment można wyciąć, powielić lub zastosować na nim dowolny efekt.



Rys. 6.11. Zaznaczenie fragmentu utworu

Przykładem stosowania efektu może być dodanie echa do zaznaczonego fragmentu utworu. W tym celu należy wybrać z menu *Efekt* opcję *Echo...* oraz ustawić parametry dotyczące jego działania. Po zatwierdzeniu zmian program przetworzy ścieżkę dźwiękową i doda pogłos do zaznaczonego efektu.



Rys. 6.12. Efekt echo

Po wykonaniu wszystkich zaplanowanych czynności zapisujemy efekt pracy – eksportujemy go do wybranego formatu dźwiękowego za pomocą menu *Plik*.
Tak zapisany plik dźwiękowy jest gotowy do publikacji na dowolnej stronie internetowej.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj najważniejsze parametry dźwięku.
2. Co to jest częstotliwość próbkowania i w jakim procesie odgrywa zasadniczą rolę?
3. Wymień etapy przetwarzania dźwięku analogowego na cyfrowy.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zainstaluj program Audacity. Przygotuj do pracy komputer – podłącz słuchawki i mikrofon. Za pomocą przycisków nawigacyjnych programu nagraj swój wokal w dowolnej aranżacji ulubionego utworu muzycznego. Wybierz najlepszy, twoim zdaniem, 10-sekundowy fragment nagrania i usuń pozostałe części. Z pozostałego fragmentu usuń szumy. Następnie dla ostatnich 5 sekund zastosuj efekt echa i wyciszenie. Ścieżkę dźwiękową wyeksportuj do pliku z rozszerzeniem wav. Tak przygotowany plik opublikuj na stronie WWW – skorzystaj ze znacznika `<audio>`. Końcowy efekt swojej pracy zaprezentuj nauczycielowi.

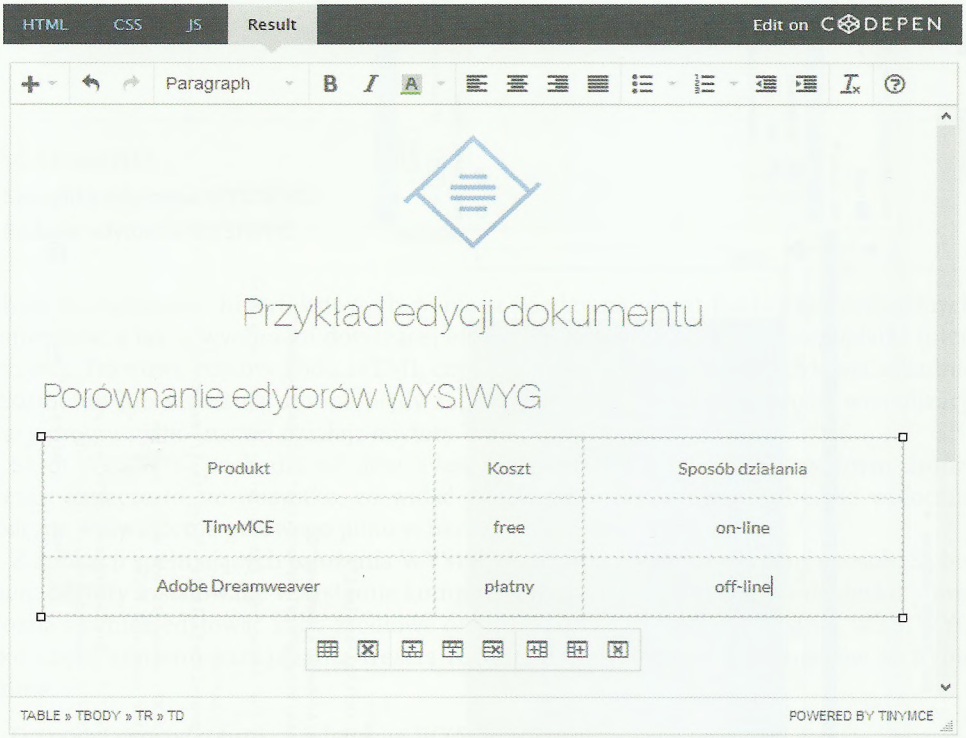
2. Za pomocą telefonu komórkowego nagraj krótki film o swojej szkole. Wykonaj montaż przygotowanego materiału – skorzystaj z dowolnego programu do edycji plików wideo. Dodaj do filmu napisy tytułowe, wprowadź swoje imię i nazwisko jako dane autora filmu. Zastosuj dowolny efekt w trakcie filmu. Wyeksportuj 30-sekundowy, gotowy materiał do formatu mov. Utwórz stronę internetową odtwarzającą film za pomocą znacznika `<video>`. Efekt pracy zaprezentuj nauczycielowi.
3. Wykonaj animację typu flash, w której narysowany dowolny kwiatek musi poruszać się po spirali. Animacja powinna trwać 10 sekund.

7. Edytory spełniające funkcje WYSIWYG

7. Edytory spełniające funkcje WYSIWYG

- Edytory WYSIWYG
- Tworzenie dokumentów w edytorach WYSIWYG

Spośród aplikacji online wybraliśmy do zaprezentowania edytor **TinyMCE**, który jest klasycznym przykładem aplikacji skryptowej do wykorzystania na stronach internetowych lub w systemach CMS. Strona domowa tego projektu znajduje się pod adresem https://www.tinymce.com.



Rys. 7.2. Okno aplikacji TinyMCE

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj działanie edytorów WYSIWYG.
2. Wymień funkcje edytorów WYSIWYG.
3. Jakie znasz edytory WYSIWYG?

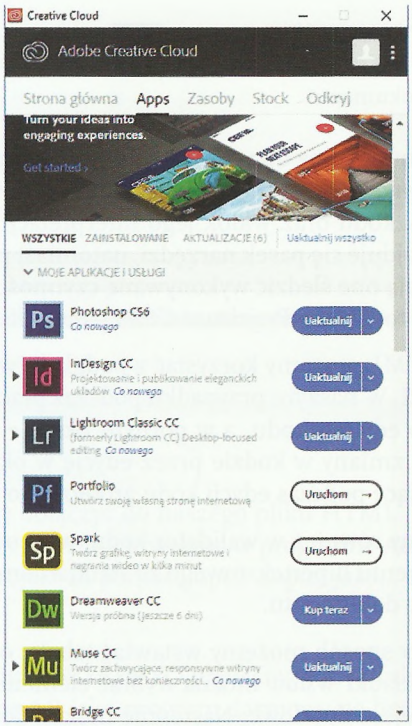
7.2 Tworzenie dokumentów w edytorach WYSIWYG

ZAGADNIENIA

- **Zasady edycji dokumentów hipertekstowych w edytorach WYSIWYG**
- **Przykłady wykorzystania edytorów WYSIWYG**

Adobe Dreamweaver CC

Pobranie wersji testowej aplikacji Adobe Dreamweaver CC jest możliwe po uprzedniej rejestracji na stronie www.adobe.com. Ciekawym rozwiązaniem jest scentralizowanie przez firmę Adobe wszystkich aplikacji pobieranych przez zarejestrowanego użytkownika w Adobe Creative Cloud. Webmasterowi usługa Creative Cloud zapewnia dostęp do wszystkich aplikacji firmy Adobe na komputery i urządzenia przenośne. Oprócz dostępu do programów i ich aktualizacji są tam wbudowane szablony ułatwiające rozpoczynanie projektowania stron internetowych, a także samouczki pozwalające krok po kroku doskonalić umiejętności w tym zakresie.



Rys. 7.3. Okno Adobe Creative Cloud

rozwiązanie jest szczególnie przydatne, gdy po publikacji witryny internetowej końcowy użytkownik projektu chce samodzielnie tworzyć i formatować wpisy na stronie. W normalnym trybie byłaby konieczna ingerencja webmastera, jednak wykorzystanie edytora w kodzie rozwiązuje ten problem.

Istnieją dwa sposoby korzystania z edytora. Pierwszy polega na pobraniu skompresowanych plików programu ze strony producenta, wypakowaniu archiwum w wybranej lokalizacji oraz podlinkowaniu w kodzie naszej strony. Drugi jest jeszcze prostszy i polega na podlinkowaniu programu bezpośrednio ze strony producenta.

PRZYKŁAD 7.1.

Skrypt pozwalający na podłączenie edytora TinyMCE do kodu naszej strony internetowej bezpośrednio z zasobów twórców programu wygląda następująco:

```
<!DOCTYPE html>
<html>
<head>
 <script
src="https://cloud.tinymce.com/stable/tinymce.min.js">
</script>
 <script>tinymce.init({ selector:'textarea' });</script>
</head>
<body>
 <textarea>Osadzanie edytora w kodzie strony</textarea>
</body>
</html>
```

Poza możliwością wykorzystania gotowego edytora na naszej stronie dostajemy narzędzia do zaawansowanej konfiguracji, która ma dopasować aplikację do naszych wymagań i potrzeb. Pełna lista możliwości konfiguracji jest dostępna na stronie **https://www.tinymce.com/docs/configure**.

Sam edytor pozwala nam na edycję kodu HTML, JS oraz CSS. Stwarza możliwość podglądu rezultatu naszej pracy w czasie rzeczywistym.

Przykład edycji dokumentu hipertekstowego w edytorze TinyMCE pokazano na rysunku 7.2.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co oznacza skrót WYSIWYG?
2. Wymień różnice między aplikacją Adobe Dreamwear a aplikacją TinyMCE.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Utwórz stronę internetową z osadzonym edytorem TinyMCE.
2. Zainstaluj wybrany przez siebie edytor WYSIWYG i utwórz za jego pomocą stronę internetową o dowolnym polskim bohaterze narodowym. Strona musi zawierać tekst, obraz i tabelę.
3. Wypisz w tabeli narysowanej w zeszycie wady i zalety edytorów WYSIWYG.

8. Projektowanie stron internetowych

- Zasady projektowania stron internetowych
- Brief projektu strony internetowej
- Szablon strony internetowej
- Dobór palety barw na stronę internetową
- Projekt graficzny strony internetowej
- Tworzenie stron zgodnie z projektem

8.1 Zasady projektowania stron internetowych

ZAGADNIENIA

- **Zasady projektowania stron internetowych**
- **Etapy projektowania stron internetowych**
- **Rodzaje struktur stron internetowych**

Projektowanie stron internetowych jest procesem złożonym i wymaga przeanalizowania wielu aspektów dotyczących wizualizacji treści. Efekt w postaci witryny musi być optymalnym kompromisem między życzeniami klientów, możliwościami technologicznymi i kosztami projektu. Nie ma jednej metody na stworzenie idealnej strony internetowej, ponieważ jest to dziedzina informatyki należąca do najbardziej rozwijających się technologicznie. Jednak na podstawie analizy technologii webowych i funkcjonujących rozwiązań webmasterskich można określić zbiór zasad stosowanych obecnie w projektowaniu i tworzeniu witryn internetowych. Poniżej wymieniono najważniejsze z nich.

1. **Temat przewodni strony**. Należy cały czas pamiętać, jaki jest cel tworzenia strony i jaką informację trzeba przekazać odwiedzającemu.
2. **Przejrzystość i minimalizm**. Użytkownik musi łatwo i szybko odszukać interesujące go treści. Odchodzi się od nadmiaru treści i elementów na stronie.
3. **Hierarchiczność struktury**. W strukturze strony poszczególne bloki muszą mieć określoną ważność w jej hierarchii. Na przykład stopka i informacje w niej zawarte nie mogą być bardziej wyeksponowane niż blok z informacją główną.
4. **Responsywność**. Jest to cecha umożliwiająca dostosowanie się strony do ekranu, na którym jest wyświetlana. Obecnie polega to na tworzeniu płynnych układów struktury strony. Oznacza to, że inna struktura może być wyświetlana na laptopie, a inna – na smartfonie.
5. **Czytelność**. Informacje na stronie muszą być czytelne. Nie można stosować zbyt małych czcionek, fontów szeryfowych czy zbyt małego kontrastu między tłem i kolorem czcionki.
6. **Szybkość ładowania**. Nikt nie lubi czekać na załadowanie się strony do okna przeglądarki. Dlatego należy ograniczać ilość elementów i wagę poszczególnych plików.
7. **Właściwy dobór palety barw**. Paleta barw to jeden z najważniejszych detali struktury graficznej strony. Najlepiej dobierać barwy z wykorzystaniem profesjonalnych aplikacji. Należy też pamiętać, że kolor może mieć określone znaczenie – w zależności od kontekstu strony, treści czy kultury regionu, w którym strona będzie publikowana. Na przykład byłoby nietaktem zastosowanie odcieni koloru czarnego dla strony sklepu sprzedającego suknie ślubne.
8. **Dopasowanie do wielu przeglądarek**. Po zaprojektowaniu i stworzeniu strony internetowej należy pamiętać o przetestowaniu jej na jak największej ilości przeglądarek, tak aby zoptymalizować wizualizację treści bez względu na oprogramowanie użytkownika.

9. **Przygotowanie do pracy z wyszukiwarkami – pozycjonowanie**. Strona internetowa musi być przygotowana do współpracy z robotami największych wyszukiwarek.
10. **Dostępność strony**. Jest to zbiór cech pozwalających na zapewnienie dostępu do zasobów internetowych możliwie jak największej liczbie użytkowników, bez względu na ich ewentualną niepełnosprawność, wiek, status majątkowy, posiadany przez nich sprzęt czy używane oprogramowanie. Na dostępność składają się: percepcja, funkcjonalność, zrozumiałość i rzetelność. Warunki spełnienia tych cech opisano w standardzie **WCAG 2.0** (ang. *Web Content Accessibility Guidelines*).
11. **Zgodność ze standardami**. Kod strony powinien być zgodny ze standardami W3C. Walidacja i testowanie muszą być elementami każdego etapu tworzenia strony.
12. **Dobrej jakości grafika i animacja**. Trudno sobie wyobrazić dobrą stronę internetową bez grafiki, animacji czy filmów. Należy jednak zwrócić uwagę na odpowiedni dobór grafiki, maksymalne zmniejszenie rozmiarów plików graficznych, wysoką jakość obrazu czy animację w formacie obsługiwanym przez większość przeglądarek. Zaleca się obecnie odejście od animacji flash na rzecz skryptów HTML, CSS i JS. Filmy najlepiej umieszczać na stronie w postaci odsyłacza do wydajnego serwera, np. YouTube. Należy pamiętać, żeby multimedia nie przesłaniały treści zawartych na stronie.
13. **Poprawność językowa**. Niedopuszczalne jest zamieszczanie informacji z błędami ortograficznymi, gramatycznymi lub stylistycznymi. Tekst powinien być umieszczony na stronie dopiero po korekcie – zarówno merytorycznej, jak i językowej. Jeżeli strona jest przeznaczona także dla obcokrajowców, należy zaplanować tłumaczenie treści i umieszczenie na stronie opcji zmiany języka.
14. **Interakcja z użytkownikiem**. Wszystkie elementy aktywne, takie jak odnośniki, banery czy pola formularza powinny mieć wyraźny wizualny fokus (zwykle w postaci ramki widocznej w trakcie nawigacji po stronie klawiszem *TAB*).
15. **Logiczne menu**. Nawigacja powinna być spójna, logiczna i niezmienna w obrębie strony. Nawigacja w obrębie całego serwisu musi być dostępna z poziomu klawiatury.

Projektowanie witryny internetowej to proces obejmujący następujące etapy:

- analiza wymagań klienta, przeznaczenia witryny oraz możliwości technicznych i finansowych;
- wstępna analiza biznesowa (kosztorys, SWOT itp.);
- dobór hostingu (serwery, domeny i subdomeny) oraz technologii i narzędzi do wykonania poszczególnych elementów witryny;
- określenie elementów polityki pozycjonowania i bezpieczeństwa dotyczących projektowanej witryny internetowej;
- stworzenie struktury logicznej i graficznej witryny i poszczególnych podstron; ewentualne projektowanie bazy danych dla witryny;
- dobór palety kolorystycznej dla witryny;
- opracowanie layoutu witryny internetowej (czcionki, rozdzielczości, tła, nawigacji, menu itp.);
- zaprojektowanie poszczególnych elementów graficznych;
- stworzenie struktury fizycznej witryny – drzewo URL plików.

W ramach struktury strony internetowej możemy wyróżnić:

- strukturę logiczną określającą topologię poszczególnych podstron, zwaną również mapą strony;
- strukturę graficzną, tworzącą szczegółowy layout poszczególnych podstron z projektami elementów graficznych i paletą barw;

- strukturę fizyczną, wskazującą drzewo katalogowe z poszczególnymi plikami składającymi się na zasoby strony internetowej.

Liczba czynności składających się na tworzenie projektu strony czy aplikacji internetowej jest tak duża, że najczęściej pracuje nad nimi określony zespół fachowców. Pozwala to na szybkie i profesjonale osiągnięcie celu, jakim jest stworzenie strony internetowej spełniającej wymagania zarówno zleceniodawcy, jak i wszystkich standardów webowych.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakich zasad należy przestrzegać, projektując stronę internetową?
2. Wymień etapy projektowania stron internetowych.
3. Opisz poszczególne struktury strony internetowej.

8.2 Brief projektu strony internetowej

ZAGADNIENIA
- **Definicja briefu**
- **Funkcje i budowa briefu**
- **Przykład briefu strony internetowej**

Analiza danych dotyczących projektowania struktury strony internetowej powinna zakończyć się stworzeniem briefu. **Brief** jest dokumentem zbierającym analizy, założenia techniczne, wymagania użytkowników oraz wyliczenia finansowe dotyczące projektowanej witryny. Z reguły jest tworzony w postaci tabeli z opisami najważniejszych – z punktu widzenia webmastera – czynników. Brief jest podstawą do dalszego projektowania elementów witryny internetowej.

PRZYKŁAD 8.1.

Poniżej przedstawiono przykład briefu wstępnego, zawierającego elementy niezbędne do planowania prac nad stroną internetową. W prawej kolumnie znajdują się przykładowe odpowiedzi klienta.

Tabela 8.1. Przykład briefu strony internetowej

Podstawowe informacje o kliencie	
Nazwa firmy	Sp. z o.o. MIŚ
Branża firmy	Handel zabawkami dla dzieci
Oferowane produkty i usługi	Zabawki pluszowe, klocki, książki, lalki itp.
Elementy identyfikacji firmy *(używane kolory, specyficzne elementy, logo)*	Miś, kolory pomarańczowy i zielony  – nasze logo
Osoba do kontaktu	Andrzej Abacki – prezes, tel. 603 000 234 prezes@gmail.com
Informacje o stronie www	
Cel tworzenia strony WWW	Reklama, zapoznanie klienta z ofertą handlową firmy
Charakter witryny *(serwis informacyjny, produktowy, korporacyjny, portal branżowy, tematyczny)*	Serwis informacyjny

Informacje o stronie www	
Adres obecnej strony WWW (jeśli istnieje)	Firma nie ma obecnie strony internetowej
Struktura logiczna strony WWW	1. Strona główna 2. Aktualności 3. O firmie • Struktura organizacyjna • Działalność 4. Oferta • Grupy produktów a) Zabawki b) Rowery c) Gry d) Książki 5. Zamówienia 6. Współpraca 7. Promocje 8. Kontakt
Wersje językowe. Jakie?	Tylko język polski
Czy firma dysponuje materiałami niezbędnymi do wykonania struktury graficznej strony WWW (grafiki, fotografie)?	Nie, tylko logo
Przewidywany budżet (opcjonalnie)	5000 zł
Oczekiwany termin wykonania projektu	3–4 tygodnie
Na jakich urządzeniach ma być wyświetlana strona?	Komputery stacjonarne i urządzenia mobilne
Preferowane przeglądarki	Firefox i Chrome
Elementy multimedialne (animacje, filmy, dźwięki, muzyka)	Animacja i podkład muzyczny
Informacje o domenie i hostingu	
Czy firma posiada domenę internetową, która będzie adresem strony WWW? *(Jeżeli nie, proszę podać preferowaną nazwę domeny do sprawdzenia jej dostępności.)*	Nie mamy www.mis.pl www.mis.lodz.pl
Czy firma posiada usługę hostingu?	Nie mamy
Dodatkowe informacje	
Czy będzie potrzebna usługa opieki i obsługi witryny WWW?	Nie, chcemy samodzielnie edytować informacje na stronie
Uwagi	

SPRAWDŹ SWOJĄ WIEDZĘ

1. Wymień podstawowe elementy briefu strony internetowej.

8.3 Szablon strony internetowej

ZAGADNIENIA
- **Rola i budowa szablonu strony internetowej**
- **Rodzaje szablonów stron WWW**
- **Narzędzia do tworzenia szablonów**

Następnym etapem jest stworzenie szablonu, nazywanego też makietą strony internetowej. Makieta jest w pewnym sensie prototypem strony lub aplikacji internetowej, ponieważ na jej podstawie powstaje finalny produkt.

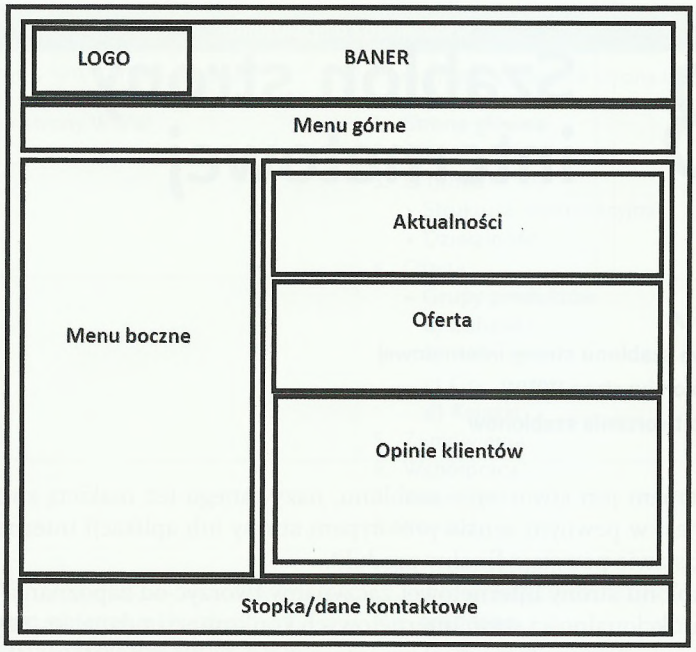
Projekt szablonu strony internetowej zaczynamy tworzyć od zapoznania się z briefem oraz analizy funkcjonalności stron internetowych konkurencji z danej branży. Każda branża może mieć specyficzne elementy stron internetowych, które są istotne dla ich klientów, i warto zobaczyć, jak wygląda to na działających już witrynach. Zawsze dobrze jest sporządzić kilka szkiców propozycji szablonu strony, który będzie określał położenie, wielkość i kształt takich elementów, jak:
- logotypy i grafiki obowiązkowe;
- banery;
- menu nawigacyjne, dodatkowe;
- bloki informacyjne;
- odsyłacze;
- inne elementy (np. stopka, formularze, wyszukiwarki, dodatkowe treści itp.).

Po konsultacji z klientem można wybrać właściwą strukturę strony – tę, która będzie bazą do tworzenia projektu graficznego.

Ważnym elementem na tym etapie prac jest ustalenie rozdzielczości, w jakiej ma pracować dany szablon. Pozwoli to na określenie proporcji poszczególnych elementów strony. Jeżeli witryna ma pracować na różnych urządzeniach, należy sporządzić makiety stron na każdy typ urządzenia.

Makietę strony można wykonać w jednej z dwóch wersji – jako:
1. makietę uproszczoną o małym stopniu szczegółowości (ang. *low-fidelity*), która zawiera podstawowe elementy interakcji oraz pozwala szybko stworzyć kilka alternatywnych opcji;
2. makietę o dużym stopniu szczegółowości (ang. *high-fidelity*), która jest bardzo zbliżona do wersji końcowej strony lub aplikacji internetowej.

Makiety uproszczone można wykonać odręcznie na kartce papieru lub skorzystać z dowolnego edytora graficznego (np. MS Paint). W takim szablonie musimy tylko nanieść kontenery o określonej wielkości na obszar strony internetowej i przypisać im określone funkcje.

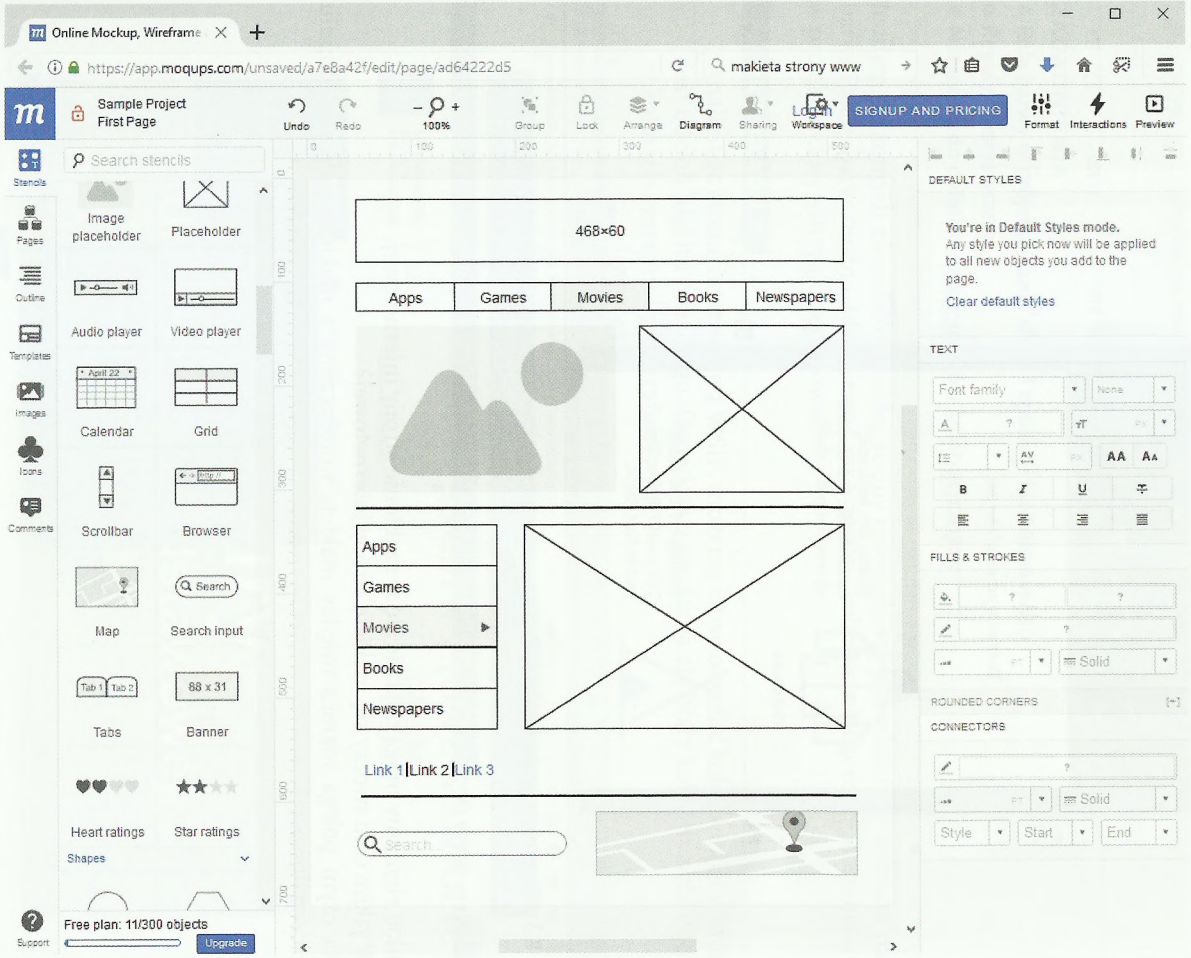


Rys. 8.1. Przykład prostej makiety strony

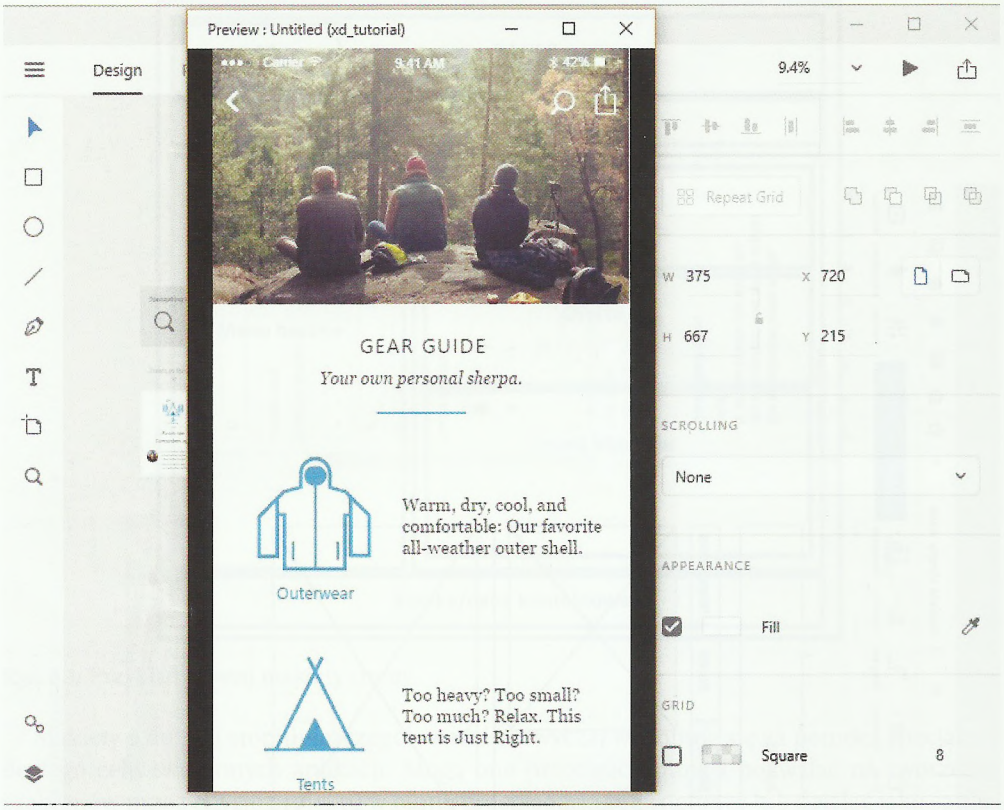
Makiety o dużym stopniu szczegółowości zazwyczaj wykonuje się za pomocą specjalnie do tego celu tworzonych aplikacji. Mogą one pracować online i pozwalać na tworzenie szablonów przez edycję projektu w oknie przeglądarki internetowej lub działać jako samodzielne aplikacje narzędziowe.

Przykładem może być aplikacja internetowa **Moqups.com**. Jest to w pełni funkcjonalna aplikacja do tworzenia makiet online. Ma intuicyjny interfejs użytkownika i bogatą bibliotekę elementów stosowanych do budowy dowolnego szablonu strony internetowej. Do korzystania z niej jest wymagana rejestracja.

Przykładem aplikacji desktopowej jest **Adobe XD** (ang. *Experience Design*). Aplikację tę można pobrać poprzez opisywaną wcześniej Adobe Creative Cloud. Jest to prosta aplikacja do tworzenia szkiców aplikacji, makiet witryn internetowych i udostępniania ich innym członkom zespołu pracującego nad projektem. Ma czytelny interfejs i duży wybór szablonów roboczych dla urządzeń mobilnych.



Rys. 8.2. Makieta o dużym stopniu szczegółowości



Rys. 8.3. Okno programu Adobe XD

SPRAWDŹ SWOJĄ WIEDZĘ

1. Jakie elementy mogą pojawić się w szablonie strony internetowej.
2. W jakim celu tworzy się makietę strony internetowej?
3. Na czym polega różnica między makietą uproszczoną a makietą o dużym stopniu szczegółowości?

8.4 Dobór palety barw na stronę internetową

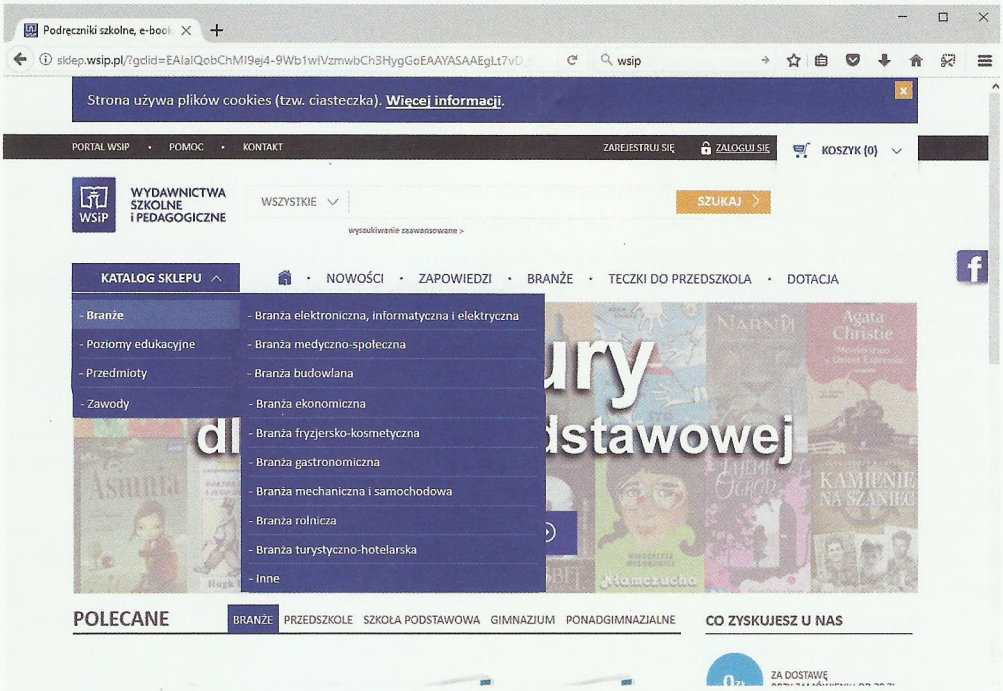
ZAGADNIENIA

- **Rola barwy w projektowaniu stron i aplikacji internetowych**
- **Reguły kolorystyczne**
- **Narzędzia do definiowania motywów kolorystycznych**

Jednym z istotnych elementów każdej strony internetowej jest właściwie dobrana paleta barw. Paletę barw można zdefiniować jako zestaw kolorów wiodących – dominujących w projekcie całej witryny internetowej.

PRZYKŁAD 8.2.

Analizując stronę internetową WSiP, możemy łatwo określić kolor wiodący, którym jest odcień błękitu pruskiego. Dodatkowo wyróżnia się biały kolor tła oraz dodatki w kolorach czarnym i pomarańczowym.



Rys. 8.4. Przykład palety barw strony internetowej WSiP

Kolory zawsze oddziałują na samopoczucie człowieka, a nawet mogą zaważyć na jego opinii dotyczącej danego przedmiotu czy usługi. Symbolika barw to utrwalony w kulturze ich związek z emocjami i wartościami istotnymi dla określonych grup społecznych lub całych narodów.

Poniżej opiszemy symbolikę niektórych kolorów, ponieważ ich wybór może zaważyć na odbiorze społecznym całego projektu.

Kolor czarny to symbol elegancji, luksusu oraz szykowności. Jest wykorzystywany tam, gdzie trzeba podkreślić powagę, tajemnicę, kompetencję lub autorytet. Kojarzony również z żałobą i śmiercią.

Kolor biały to uniwersalny kolor symbolizujący pokój i czystość. Jest kojarzony z prawdomównością, prostotą, dobrem, jasnością lub przestrzenią.

Kolor brązowy jest symbolem spokoju, powagi, ciepła, neutralności i niezawodności.

Kolor czerwony symbolizuje przygodę, energię, niebezpieczeństwo, miłość, pasję, wigor i siłę. To także symbol życia i sprzecznych, skrajnych emocji, piękna i witalności.

Kolor pomarańczowy to barwa młodości, kreatywności, lekkości i aktywności. Symbolizuje zabawę i entuzjazm.

Kolor żółty symbolizuje radość, ciepło, słońce i dobrą pogodę.

Kolor zielony to barwa natury, kojarzy się z życiem i harmonią. Jest też symbolem uzdrowienia, nadziei i wolności.

Kolor niebieski przywodzi na myśl spokój, stabilność, sukces, pewność, lojalność i bezpieczeństwo.

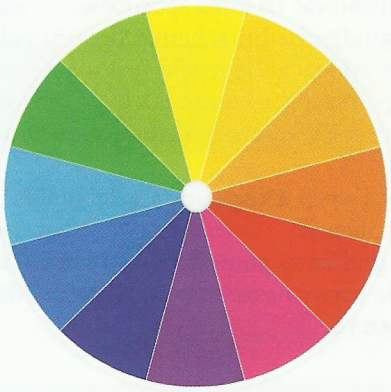
Opisano sześć klasycznych reguł kolorystycznych opartych na tak zwanym **kole kolorów** (ang. *color wheel*) przedstawionym na rysunku 8.5. Dzięki nim można uzyskać harmonijne kompozycje kolorystyczne również na stronach i aplikacjach internetowych. Składają się na nie niżej opisane reguły.

1. Dopełniające (komplementarne) – są to kolory kontrastujące ze sobą. Znajdują się po przeciwnych stronach koła kolorów.
2. Analogowe – są to kolory sąsiadujące ze sobą. Barwy analogowe zwykle dobrze mieszają się ze sobą oraz są harmonijne.
3. Monochromatyczne – reprezentujące odcienie jednego koloru. Monochromatyczne kolory dobrze do siebie pasują i dają uspokajający efekt.
4. Dopełniające rozbite – są to trzy kolory kontrastujące ze sobą.
5. Tworzące trójkąt (triada) – są to trzy kolory znajdujące się w równej odległości (co 120°) od siebie na kole kolorów.
6. Tworzące kwadrat (złożone) – są to barwy znajdujące się na czterech wierzchołkach kwadratu wpisanego w koło kolorów.

Pierwsze koło kolorów powstało w XVII wieku i do dzisiaj jest podstawowym wzorcem w malarstwie, reklamie czy projektowaniu wnętrz. Cechą charakterystyczną koła kolorów jest to, że po jednej stronie są skumulowane kolory ciepłe, a po przeciwnej – chłodne.

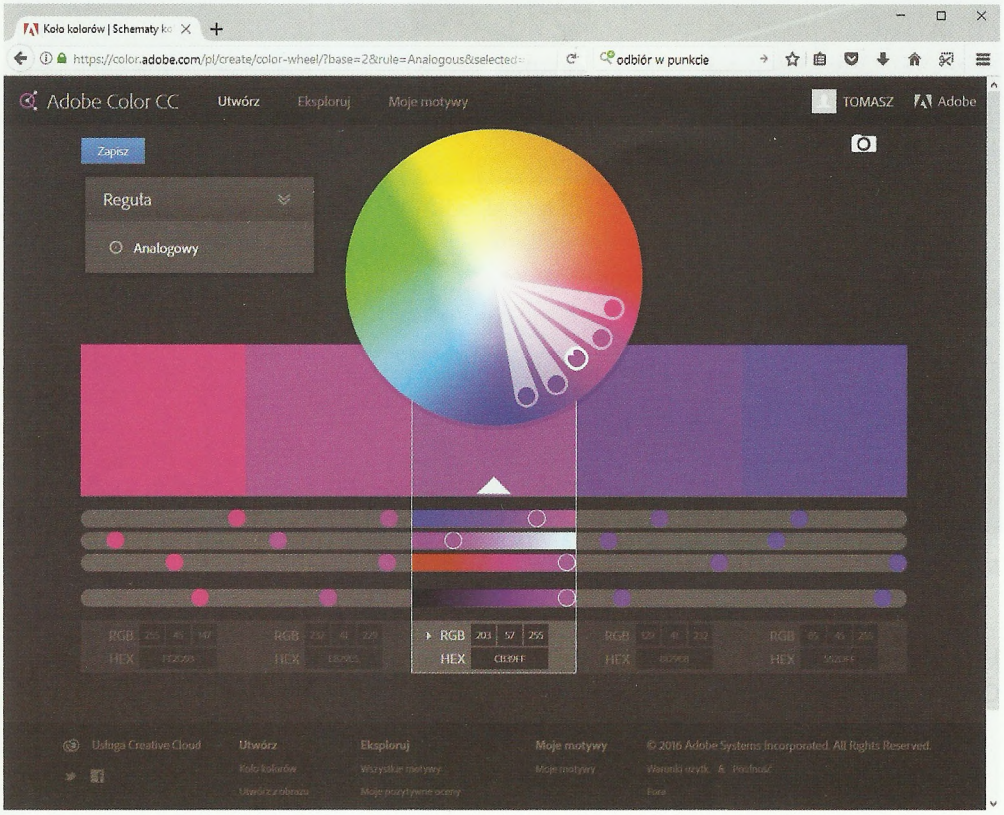
Samodzielny dobór palety barw wymaga jednak pewnego doświadczenia i zmysłu artystycznego. Współczesny webmaster może skorzystać z pomocy profesjonalnych aplikacji, które pomogą ustalić najlepszy zbiór barw do określonego zastosowania.

W tego typu aplikacjach motyw kolorystyki strony można utworzyć samodzielnie według określonych reguł lub na podstawie związanych ze stroną obrazów wczytywanych online. Przykładem takiej aplikacji internetowej, uruchamianej w przeglądarce internetowej, jest Adobe Color CC dostępny na stronie `https://color.adobe.com`.



Rys. 8.5. Koło kolorów

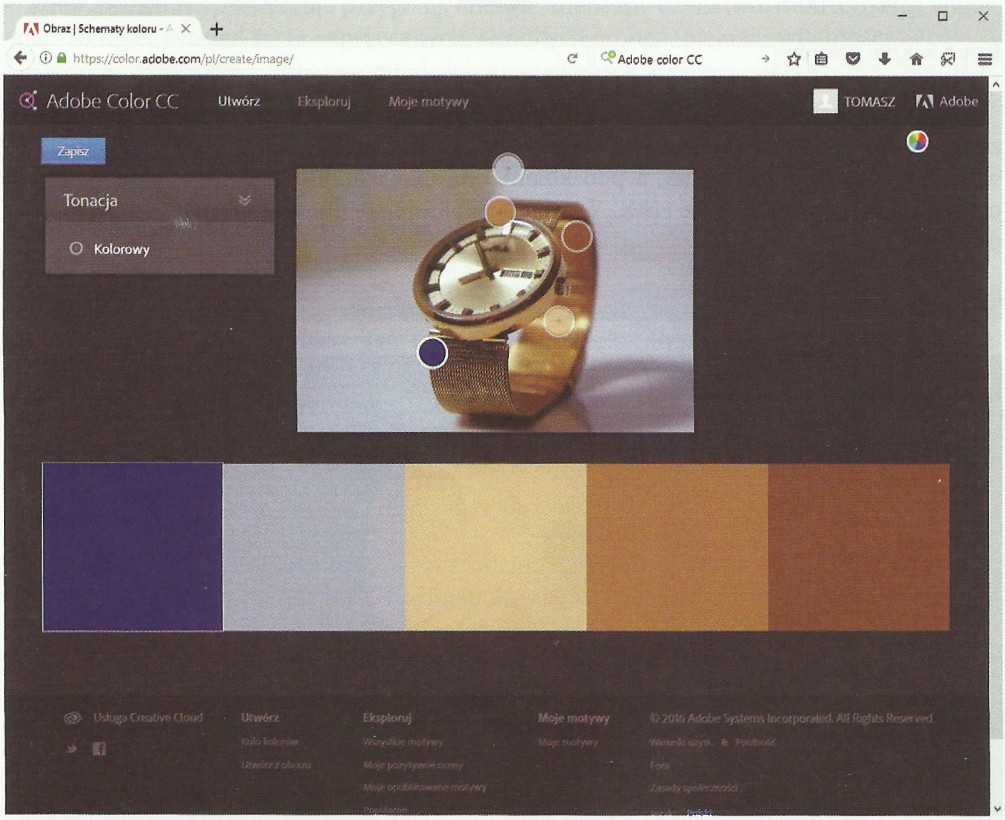
Okno aplikacji jest czytelne, a interfejs obsługi bardzo intuicyjny. Na górnym pasku znajduje się proste menu nawigacyjne, w centralnym oknie – koło kolorów z punktami wskazującymi wybrane barwy, a pod kołem – próbki palety pięciu barw z możliwością odczytania kodów RGB oraz HEX. W stopce znajduje się dodatkowe menu nawigacyjne użytkownika.



Rys. 8.6. Okno aplikacji Adobe Color CC

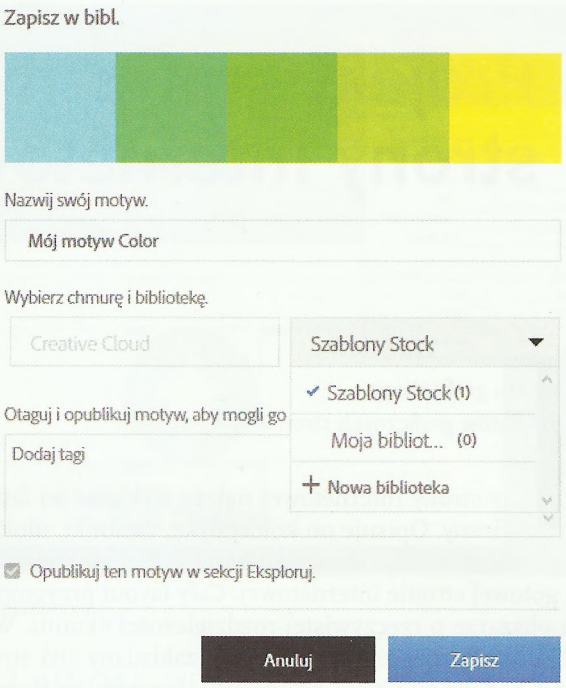
Paletę barw możemy zdefiniować na kilka sposobów.

1. Wybrać z menu **Reguła** odpowiedni schemat doboru palety oraz określić punktem wskaźnika pierwszy kolor wiodący. Do dyspozycji mamy schematy: analogowy, monochromatyczny, triada, dopełniający, złożony, odcienie oraz własny.
2. Wprowadzić kod RGB lub HEX koloru podstawowego (wiodącego) i dobrać pozostałe według określonej reguły.
3. Wybrać motyw za pomocą opcji **Eksploruj** z biblioteki gotowych palet zapisanych przez użytkowników tej aplikacji.
4. Wczytać główne logo lub grafikę za pomocą opcji **Utwórz z obrazu** i na bazie ich kolorystyki dobrać pozostałe barwy naszej palety. Wówczas dobieramy parametry tonacji palety barw z menu **Tonacja**.



Rys. 8.7. Dobór palety kolorów przez obraz

Po zakończeniu procesu definiowania palety barw możemy zapisać kody wybranych kolorów w notacji RGB lub HEX, a także zapisać wypracowany motyw w bibliotece Adobe Color CC.



Rys. 8.8. Okno zapisywania motywu w bibliotece

Jak można zauważyć, dobór barw ma często związek z wyborem innych materiałów graficznych wykorzystywanych do projektowania strony. Dlatego też do tego etapu należy podejść w momencie, gdy mamy już przygotowane obrazy, zdjęcia i inne materiały multimedialne. Pozwoli to na ujednolicenie standardu kolorów dla wszystkich elementów graficznych strony. Po wyborze palety kolorów może się okazać, że niektóre zdjęcia lub grafiki należy poddać dodatkowej obróbce, aby dopasować je do przyjętego zestawu barw.

Po ustaleniu kolorystyki elementów strony internetowej oraz przygotowaniu zdjęć obrazów i multimediów można przystąpić do następnego etapu prac projektowych – graficznego layoutu strony.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym charakteryzuje się monochromatyczny standard kolorystyczny oparty na kole kolorów?
2. Jaką symbolikę mają kolory czerwony i brązowy?
3. Do jakich celów stosujemy koło kolorów?
4. Jaką rolę na stronie odgrywa kolor wiodący?

8.5 Projekt graficzny strony internetowej

ZAGADNIENIA

- **Layout strony WWW**
- **Zasady tworzenia projektu graficznego**
- **Programy do edycji projektów graficznych stron WWW**

Po zatwierdzeniu makiety strony internetowej należy wykonać jej layout, czyli stworzyć układ kompozycyjno-graficzny. Opisuje on kolorystykę, czcionki, ulokowanie elementów i grafik, wielkość nagłówków, ustala również hierarchię elementów w taki sposób, jak będzie to widoczne na gotowej stronie internetowej. Cały layout przygotowuje się już w skali 1 : 1 na przyjętym obszarze o rzeczywistej rozdzielczości ekranu. W fazie projektowej layout ma postać pliku graficznego prezentującego zakładany styl strony. W następnym etapie prac layout jest kodowany i cięty na gotowe elementy graficzne wykorzystywane później na gotowej stronie czy aplikacji.

Podczas tworzenia projektu graficznego witryny należy konsekwentnie trzymać się zdefiniowanej palety barw oraz szablonu strony. Wykorzystujemy również przygotowane wcześniej elementy, takie jak:

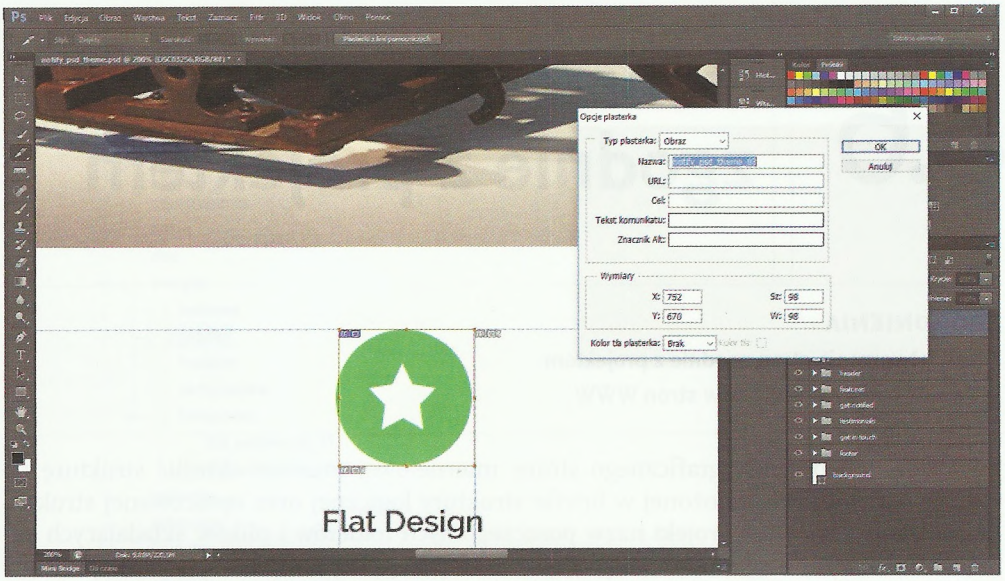
- grafiki statyczne,
- zdjęcia,
- animacje,
- filmy.

Ponieważ layout strony jest plikiem graficznym, można go wykonać w edytorach grafiki rastrowej. Najczęściej wybierany przez webmasterów jest program **Adobe Photoshop**, ale można też wykorzystać znany już **GIMP**.

Program Adobe Photoshop oferuje wiele funkcji przydatnych dla webmasterów. Ułatwia opracowywanie zawartości dla różnych wielkości ekranów i urządzeń oraz grafiki dzięki dostępowi do nieograniczonego obszaru kompozycji. **Obszary kompozycji** usprawniają proces projektowania przez udostępnienie nieskończonego obszaru roboczego, na którym można edytować jednocześnie wersje przeznaczone dla różnych urządzeń i ekranów o różnych rozdzielczościach. Podczas tworzenia obszaru kompozycji można wybrać jedno z wielu ustawień predefiniowanych lub określić własny rozmiar. Dodatkowo obszary kompozycji są zoptymalizowane do pracy w trybie kolorów RGB.

Obszary kompozycji można eksportować jako obrazy JPEG, GIF, PNG i SVG oraz jako dokumenty PDF.

Dodatkowym narzędziem w projektowaniu layoutu strony WWW jest plasterek. **Plasterki** dzielą obraz na mniejsze obrazy, aby potem, za pomocą języka HTML lub CSS, można było je łączyć w stronę internetową. Plasterki są tworzone na podstawie warstw albo za pomocą narzędzia *Plasterki*. Cięcie projektu graficznego na plasterki jest szybkie i pozwala zapisać wszystkie za jednym razem na koniec pracy z projektem.



Rys. 8.10. Tworzenie plasterka

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest plasterek i do czego służy?
2. Opisz programy graficzne wykorzystywane podczas opracowywania projektu graficznego strony internetowej.

8.6 Tworzenie stron zgodnie z projektem

ZAGADNIENIA

- **Etapy tworzenia stron zgodnie z projektem**
- **Zasady realizacji projektów stron WWW**

Po wykonaniu projektu graficznego strony internetowej musimy określić strukturę fizyczną. Na podstawie założonej w briefie struktury logicznej oraz opracowanej struktury graficznej tworzymy projekt nazw poszczególnych folderów i plików składających się na zasoby witryny internetowej. Uporządkuje to pracę podczas przekształcania projektu w dokumenty hipertekstowe, arkusze stylów, skrypty czy graficzne elementy strony.

Struktura logiczna treści to schemat, który pomaga nam ułożyć informacje tak, by były one czytelne dla potencjalnego użytkownika witryny. Schemat ten może mieć postać listy w dokumencie tekstowym lub drzewa katalogowego. Klient, dla którego jest tworzona strona internetowa, powinien brać czynny udział w ustaleniu priorytetów dla struktury fizycznej.

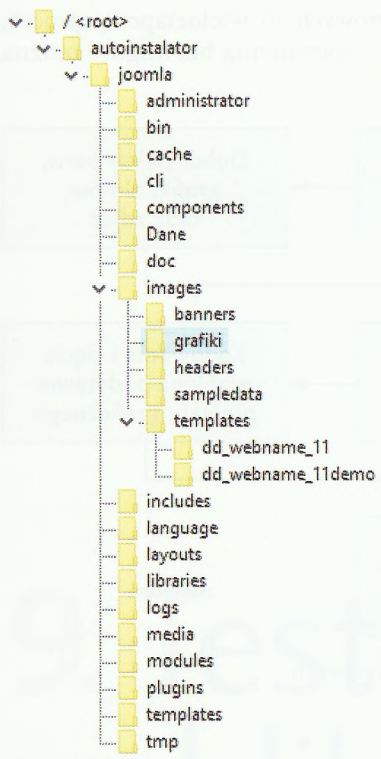
Po zatwierdzeniu struktury fizycznej można przystąpić do cięcia i eksportowania elementów graficznych z projektu graficznego oraz kodowania poszczególnych podstron i arkuszy stylów.

Przyjęta makieta strony pozwoli na zdefiniowanie w języku HTML odpowiednich kontenerów oraz nadanie im określonych wymiarów i wyglądu w arkuszach CSS.

Plasterki pozwolą na przygotowanie grafik przycisków, logo, symboli, banerów, ikon oraz elementów graficznych makiety strony. Kolorystyka, położenie oraz wymiary elementów zostaną odpowiednio zakodowane w języku HTML i CSS. Zaprojektowane treści można skopiować do dokumentów hipertekstowych. W określonych w projekcie miejscach kodujemy również osadzenie przygotowanych obiektów graficznych i multimediów.

Pocięty i zakodowany projekt zapisujemy w plikach oraz katalogach według przyjętej struktury fizycznej strony. W ten sposób zasoby strony są gotowe do publikacji na serwerze WWW.

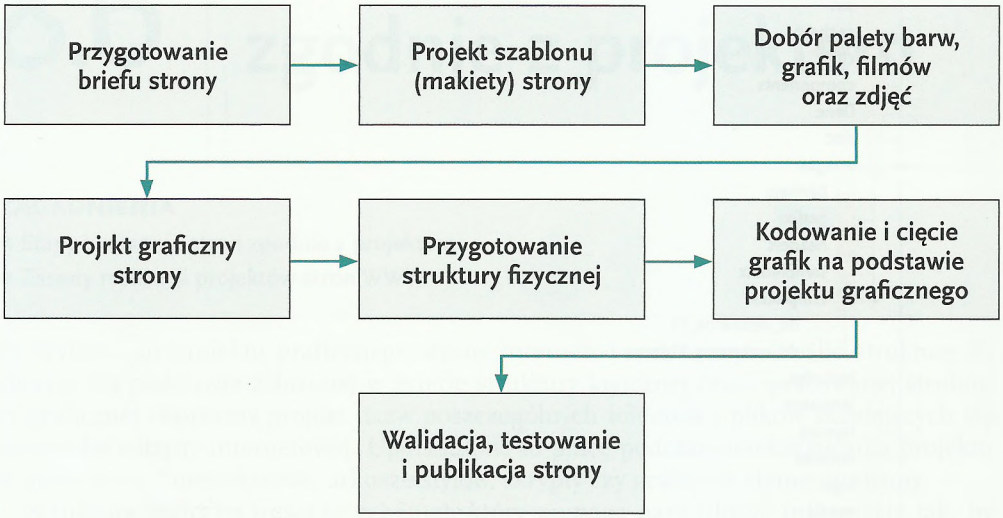
Po skopiowaniu gotowej struktury fizycznej na serwer WWW można rozpocząć proces testowania i optymalizacji witryny internetowej.



Nazwa	Rozmiar	Zmodyfikowany	Prawa	Właścic...
..				
str.png	3 KB	28.07.2016 14:33:50	rw-r--r--	
pismo.jpg	241 KB	27.07.2016 11:07:22	rw-r--r--	
logoFull.png	7 KB	27.07.2016 15:22:23	rw-r--r--	
logobip.png	4 KB	27.07.2016 15:46:10	rw-r--r--	
librus.png	3 KB	28.07.2016 14:29:15	rw-r--r--	
kontakt1.png	3 KB	28.07.2016 14:31:38	rw-r--r--	
kontakt.png	3 KB	28.07.2016 14:30:05	rw-r--r--	
h.jpg	70 KB	27.07.2016 11:07:35	rw-r--r--	
galerie.png	3 KB	28.07.2016 14:32:16	rw-r--r--	
fb.png	3 KB	28.07.2016 14:42:51	rw-r--r--	
curie.png.bmp	44 KB	27.07.2016 15:20:21	rw-r--r--	
bbip1.png	2 KB	12.01.2017 08:52:46	rw-r--r--	
bbip.png	3 KB	12.01.2017 08:49:52	rw-r--r--	

Rys. 8.11. Przykład struktury fizycznej CMS Joomla

Jak widać, projektowanie stron i aplikacji internetowych to wieloetapowy proces, który – jeśli pominiemy aspekty techniczne i kwestię zapewnienia hostingu – można podzielić na następujące etapy:



Rys. 8.12. Etapy procesu projektowania stron i aplikacji internetowych

SPRAWDŹ SWOJĄ WIEDZĘ

1. Na czym polega projektowanie struktury fizycznej witryny internetowej?
2. Omów etapy tworzenia stron zgodnie z projektem.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Wykonaj projekt makiety strony o rozdzielczości 800 px na 600 px, która ma się składać z następujących elementów:
   - nagłówka zawierającego logotyp po lewej stronie;
   - poziomego menu z pięcioma przyciskami, umieszczonego pod nagłówkiem;
   - dwóch sekcji z treścią informacyjną pod menu po stronach lewej i prawej;
   - stopki na dole strony.
2. Otwórz w przeglądarce stronę twojej szkoły. Określ paletę kolorów wykorzystaną do jej wykonania. Wyznacz z opisanej palety kolor wiodący.

9. Testowanie, publikacja oraz optymalizacja stron internetowych

- Weryfikacja oraz walidacja stron internetowych
- Publikacja gotowych stron WWW
- Pozycjonowanie i optymalizacja

9.1 Weryfikacja oraz walidacja stron internetowych

ZAGADNIENIA

- Weryfikacja a walidacja
- Testowanie stron WWW
- Analiza wyników testów

Weryfikacja procesu tworzenia stron i aplikacji internetowych, służy sprawdzeniu zgodności z założeniami. Weryfikacja powinna występować na każdym etapie projektu. Walidacja natomiast pomaga sprawdzić zgodność kodu ze standardami W3C oraz możliwościami przeglądarek internetowych lub urządzeń, na których produkt będzie użytkowany. Te dwie czynności pozwalają na uzyskanie pewności, że program, aplikacja internetowa czy strona WWW działa poprawnie oraz spełnia wymagania użytkownika.

Najlepszą metodą weryfikacji jest testowanie. Testowanie powinno się odbywać na podstawie utworzonego wcześniej harmonogramu. Dokument harmonogramu testów musi zawierać opis procesu testowania, czyli tego, w jaki sposób test będzie przeprowadzany oraz kto będzie za niego odpowiedzialny. Plan testów oprogramowania czy stron internetowych powinien być tak skonstruowany, by umożliwić śledzenie wymagań użytkownika. Plan testów musi również zawierać zakres kryteriów, według których będzie oceniana poprawność produktu.

W przypadku stron WWW stosuje się często wielokryterialną metodę oceny serwisów internetowych WAES (ang. *Website Attribute Evaluation System*). Polega ona na binarnej weryfikacji kilkunastu kryteriów podzielonych na dwa obszary: przejrzystość oraz interaktywność/dostępność.

Jeżeli chodzi o sposób oraz liczbę zaangażowanych w testowanie osób, można wyróżnić testowania:

- jednostkowe, gdy osoba testująca jest twórcą kodu;
- integracyjne i systemowe, gdy jest testowana współpraca poszczególnych elementów całego projektu;
- wdrożeniowe, w których uczestniczą końcowi użytkownicy.

Walidacja, czyli sprawdzanie poprawności technologicznej stron WWW, polega na ocenie następujących parametrów:

- czasu ładowania strony;
- weryfikacji działania odnośników wewnętrznych i zewnętrznych;
- weryfikacji działania formularzy;
- zgodności z wyszukiwarkami;
- zgodności z przeglądarkami;
- użycia elementów zdeprecjonowanych;

- poprawności językowej;
- błędów składni kodu.

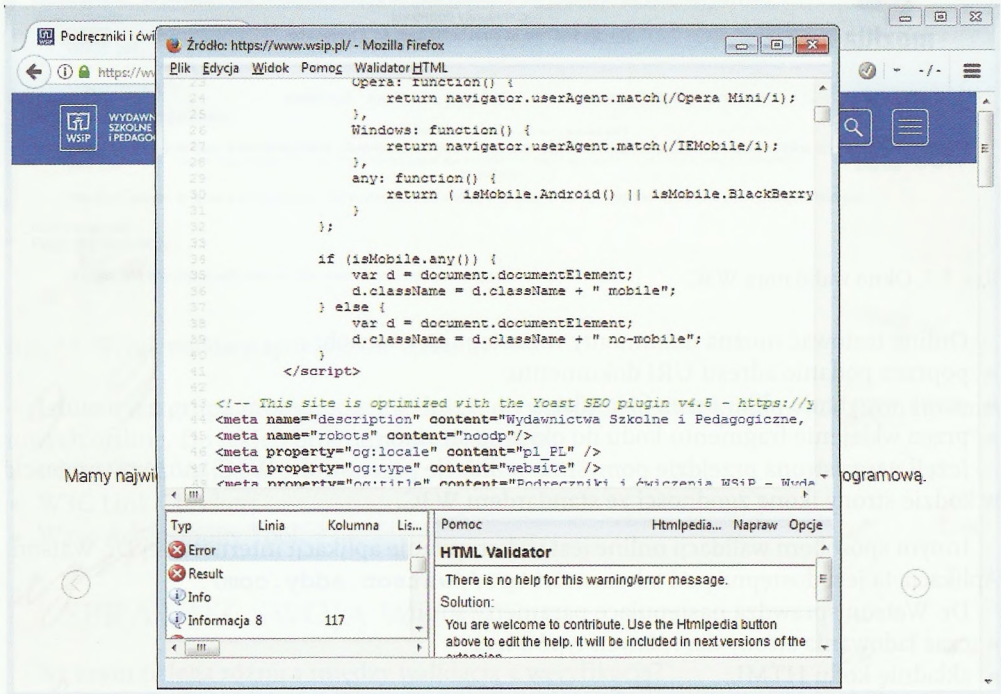
Walidację możemy przeprowadzać za pomocą skryptów na stronie (np. sprawdzanie poprawności danych wprowadzanych przez użytkownika) lub specjalnie do tego celu przygotowanych aplikacji lokalnych oraz online.

Pierwszy etap walidacji powinien następować w trakcie pisania kodu aplikacji. Dotyczy to przede wszystkim poprawności użycia danych źródłowych w skryptach lub sprawdzania właściwego wypełnienia pól dowolnego formularza. Efektem właściwej walidacji powinna być informacja zwrotna wyświetlana na ekranie, informująca użytkownika o błędnym wprowadzeniu danych lub ich braku.

Natomiast testowanie błędów składni kodu można wykonać za pomocą parserów. **Parser** to program przeprowadzający analizę składniową danych wejściowych w celu określenia ich struktury gramatycznej w odniesieniu do określonej gramatyki wzorcowej. W walidacji stron internetowych możemy korzystać z aplikacji lokalnych, na serwerach zdalnych (W3C) oraz z odpowiednich wtyczek instalowanych w przeglądarkach internetowych.

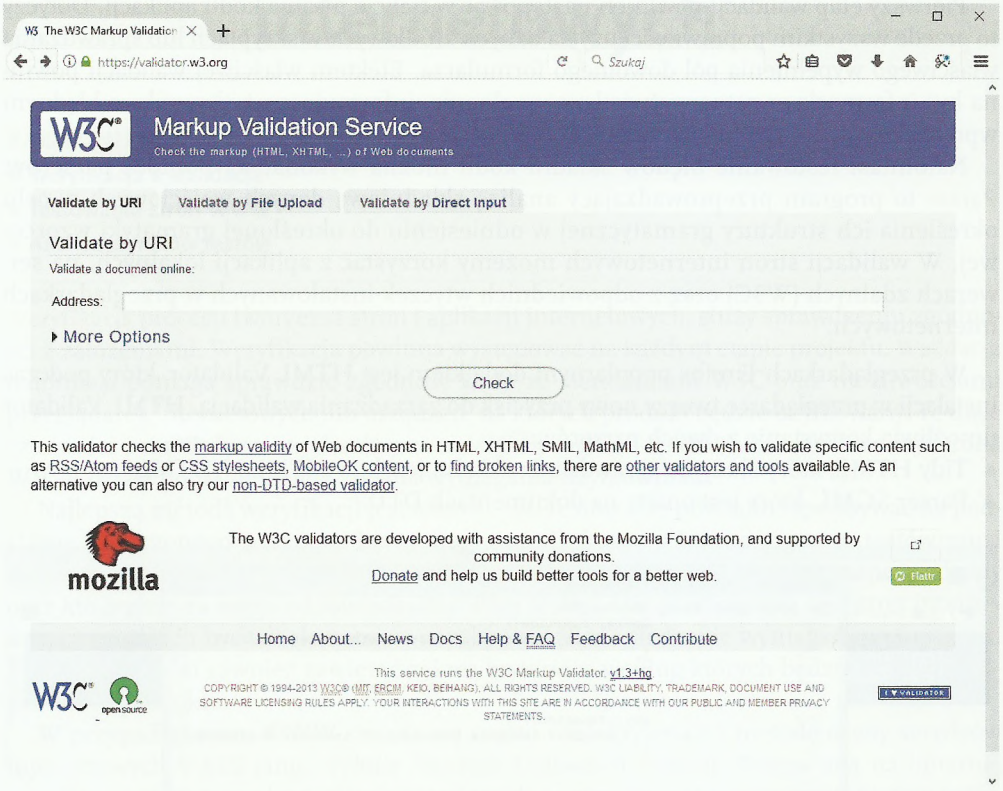
W przeglądarkach Firefox popularnym dodatkiem jest HTML Validator, który podczas instalacji w przeglądarce tworzy nowy przycisk do zarządzania walidacją. HTML Validator umożliwia korzystanie z dwóch parserów:

- Tidy HTML, który służy do sprawdzania składni języków HTML i XHTML;
- Parser SGML, który jest oparty na dokumentach DTD.



Rys. 9.1. Okno walidatora przeglądarki

Walidację mogą przeprowadzić instalowane dodatki do przeglądarek, ale istnieje też jest możliwość testowania kodu plików składających się na stronę internetową za pomocą walidatorów online W3C. Możemy testować w ten sposób dokumenty HTML, XHTML, SMIL, MathML i inne. Dodatkowo do dyspozycji mamy linki do narzędzi walidacji RSS, arkuszy CSS czy odnośników.



Rys. 9.3. Okno walidatora W3C

Online testować można dokumenty w następujący sposób:

- poprzez podanie adresu URI dokumentu;
- przez wczytanie pliku przez formularz;
- przez wklejenie fragmentu kodu do okna na stronie.

Jeżeli nasza strona przejdzie pomyślnie walidację przez stronę W3C, możemy umieścić w kodzie strony ikonę zgodności ze standardem W3C.

Innym sposobem walidacji online jest wykorzystanie aplikacji internetowej Dr. Watson. Aplikacja ta jest dostępna pod adresem **`http://watson.addy.com/`**.

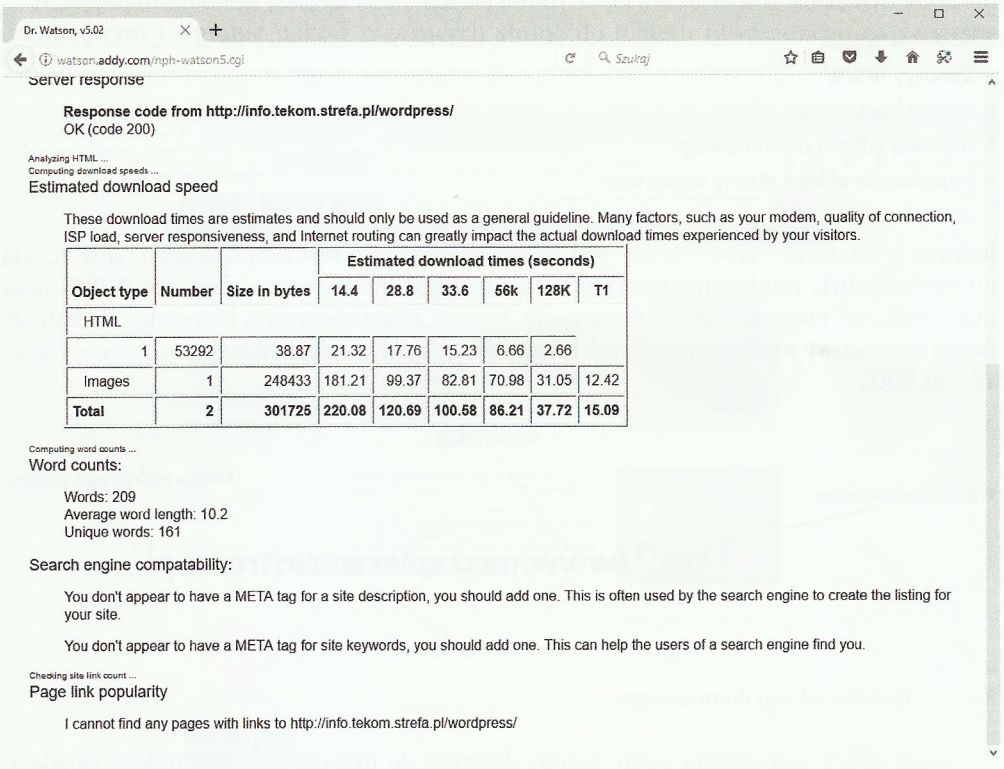
Dr. Watson sprawdza następujące parametry strony:

- czas ładowania strony dla różnych typów łącz;
- składnię kodu HTML;
- statystykę słów;
- liczbę słów, przeciętną długość słowa;
- pisownię;
- weryfikację odnośników oraz odnośników do plików graficznych;

- poprawność znaczników META, ułatwiającą właściwe indeksowanie przez przeglądarki;
- popularność badanej strony;
- liczbę linków do badanej strony.

Testowanie strony polega na podaniu adresu URL strony oraz zaznaczeniu opcji walidacji.

Dodatkowo zawsze dobre efekty przynosi testowanie gotowej strony internetowej w różnych przeglądarkach internetowych. Pozwala to na wyłapanie anomalii w ich wyświetlaniu i umożliwia likwidowanie błędów na bieżąco.



Rys. 9.3. Wynik walidacji aplikacją Dr. Watson

Istnieje też grupa programów lokalnych do walidacji kodów źródłowych stron internetowych offline. Do takich programów zaliczamy:

- CSE HTML Validator;
- W3C Link Checker;
- Wave Accessibility Tool.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Na czym polega różnica między walidacją a weryfikacją?
2. Omów metodę oceny serwisów internetowych WAES.
3. Co to jest parser?
4. W jaki sposób można testować dokumenty hipertekstowe za pomocą walidatora W3C?

9.2 Publikacja gotowych stron WWW

ZAGADNIENIA

- **Serwery WWW**
- **Usługi hostingowe**
- **Budowa adresu domenowego**
- **Kopiowanie plików strony na serwer**

Jednym z ostatnich etapów pracy nad stroną internetową jest jej publikacja. Aby strona internetowa była publicznie dostępna, musi być opublikowana na serwerze WWW oraz mieć unikalny publiczny adres domenowy, będący alfanumeryczną konwersją adresu IP. Adres domenowy wpisujemy w pasek adresu przeglądarki internetowej w końcowej części adresu URL.

Rys. 9.4. Budowa adresu domenowego

Serwer WWW udostępnia swoje zasoby dyskowe do przechowywania plików składających się na stronę internetową oraz obsługuje żądania protokołu komunikacyjnego HTTP, który pozwala na przesyłanie dokumentów hipertekstowych w sieci internet. Dodatkowo serwer WWW oferuje wiele usług pozwalających na stabilną i bezpieczną dostępność strony internetowej w sieci oraz oprogramowanie backendowe, ułatwiające zarządzanie witryną. Obecnie do najpopularniejszych serwerów zaliczamy darmowy **Apache** oraz **IIS** firmy Microsoft.

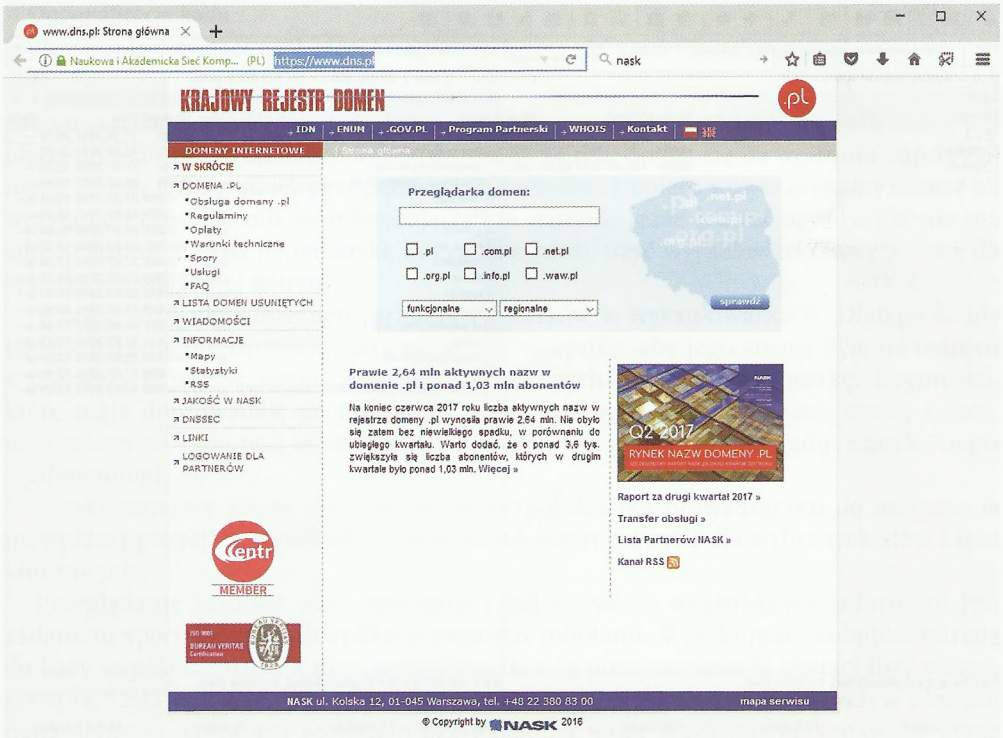
Stronę internetową możemy publikować na serwerze własnym lub skorzystać z usług jednego z obecnych na rynku dostawców usług hostingowych (ang. *provider*). Jeżeli decydujemy się na własny serwer, musimy pamiętać, że musimy wydzielić do tego celu osobny komputer pracujący przez 24 godziny na dobę oraz mieć zewnętrzny (publiczny) adres IP „widoczny" w sieci internetowej.

Zewnętrzny serwer WWW w ramach usługi hostingowej jest wygodniejszym i bezpieczniejszym rozwiązaniem, ale z reguły wiąże się z opłatami. Często jednak w jednym miejscu otrzymujemy zarówno gotowy do pracy serwer WWW, jak i adres domenowy widoczny w sieci internet.

Jeżeli jednak naszym celem jest posiadanie domeny pozwalającej na dobre pozycjonowanie, możemy samodzielnie ją zdefiniować i zarejestrować w Krajowym Rejestrze Domen prowadzonym przez NASK. Rejestr domen jest dostępny pod adresem **`https://www.dns.pl/`**.

Na stronie można sprawdzić dostępność domen poziomów najwyższego i drugiego oraz zarejestrować i opłacić wybraną domenę. Jeżeli zarejestrujemy domenę w NASK czy u innego operatora domen lokalnych, możemy ją delegować do naszego dostawcy usług hostingowych.

Następnym krokiem jest złączenie naszej strony w serwerze WWW z naszym adresem domenowym i wgranie plików tworzących stronę do folderu utworzonego na serwerze WWW.



Rys. 9.5. Strona Krajowego Rejestru Domen

Do kopiowania plików na serwer zdalny możemy wykorzystać panel administratora usługi hostingowej lub klienta usługi FTP. W przypadku serwera lokalnego wystarczy skopiowanie plików do odpowiedniego folderu. Jeśli chodzi o własny serwer Apache, to jako root możemy skopiować pliki strony do folderu /var/www/html. Jeżeli korzystamy z pakietu serwerowego XAMPP, pliki kopiujemy do folderu **`C:\xampp\htdocs\`**.

W przypadku serwerów zdalnych provider powinien poinformować, gdzie znajduje się folder do kopiowania plików strony, oraz podać parametry logowania do usługi FTP. Najczęściej w takiej sytuacji serwery WWW zawierają foldery przypisane do użytkowników:

`domains/nazwadomeny/public_html`

Do czynników wpływających na korzystne pozycjonowanie strony internetowej zaliczamy między innymi:

- zastosowanie dużej liczby dobrej jakości hiperłączy wewnętrznych i zewnętrznych;
- płynna komunikacja między poszczególnymi stronami danej witryny;
- trafnie dobrane słowa i frazy kluczowe oraz ich odpowiednie rozmieszczenie w treści poszczególnych stron;
- optymalizację kodu źródłowego poszczególnych stron;
- łączenie odsyłaczami naszej strony z innymi witrynami o innych adresach IP;
- zastosowanie złożonego tytułu strony internetowej, np.:

```
<title> Wydawnictwa Szkolne i Pedagogiczne WSiP - podręczniki,
ćwiczenia, książki, sklep internetowy</title>
```

- marketing strony prowadzony w mediach społecznościowych;
- stosowanie opisów hiperłączy.

Optymalizacja strony internetowej może również dotyczyć jej responsywności. **Responsywny design** (ang. *responsive web design*) to właściwość strony internetowej dostosowującej się automatycznie do wielkości ekranu, na którym jest wyświetlana.

Do zalet responsywnej strony internetowej należy zaliczyć:

- łatwość oglądania (ponieważ wielkość liter i obrazy dopasowują się do wyświetlacza);
- potencjalnie większą liczbę odwiedzin (każdy bowiem może taką stronę otworzyć w dowolnym miejscu i czasie, na dowolnym urządzeniu);
- oszczędności dla projektu strony (ponieważ nie trzeba inwestować dodatkowo w kosztowną wersję mobilną lub specjalne aplikacje).

Responsywność można wprowadzać już od etapu projektowania aż po etap tworzenia kodu strony w języku HTML i kaskadowych arkuszy stylów CSS.

W dokumentach hipertekstowych możemy użyć metaznaczników zapewniających prawidłowe wyświetlanie strony na urządzeniach mobilnych.

PRZYKŁAD 9.1

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
```

Dodatkowo należy w odpowiedni sposób sformatować poszczególne elementy strony za pomocą arkuszy CSS. Na przykład szerokość bloków powinno podawać się w procentach. Należy również korzystać z selektorów i właściwości pozwalających na automatyczną zmianę stylu w zależności od rozdzielczości ekranu.

PRZYKŁAD 9.2

Na rysunku przedstawiono przykład kodu strony internetowej, która wybiera styl w zależności od zmiany wielkości okna przeglądarki. W przypadku dużego okna jest wyświetlany obraz o większych rozmiarach, natomiast gdy okno przeglądarki jest mniejsze niż 400 px, zmienia się również rozmiar wyświetlanego obrazu.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
/* Dla rozdzielczości poniżej 400px: */
body {
    background-repeat: no-repeat;
    background-image: url('mały_kotek.jpg');
}

/* Dla rozdzielczości powyżej 400px: */
@media only screen and (min-width: 400px) {
    body {
        background-image: url('kotek.jpg');
    }
}
</style>
</head>
<body>
<p style="margin-top:300px;">Przykład responsywności grafiki</p>
</body>
</html>
```

Rys. 9.7. Przykład kodu, który dopasuje obraz do rozdzielczości ekranu przeglądarki

Optymalizacja strony to również dopasowanie jej treści do aktualnych wytycznych prawnych lub dostosowanie do obsługi przez osoby niepełnosprawne.

Aspekty prawne mogą dotyczyć elementów, logotypów lub odsyłaczy, które muszą się pojawić na stronach internetowych określonych jednostek organizacyjnych. Dla przykładu, na stronach internetowych instytucji podległych rządowi lub samorządom lokalnym muszą pojawić się takie elementy, jak godło państwowe i odsyłacz do Biuletynu Informacji Publicznej (BIP). Ich brak może spowodować konsekwencje prawno-finansowe dla właściciela strony.

Również ważnym aspektem interfejsu obsługi publicznych stron internetowych jest uwzględnienie elementów pomagających w korzystaniu z niej przez osoby niepełnosprawne. Często ułatwienia takie są nazywane **dostępnością** strony internetowej. Przykładem jest tutaj umieszczanie na stronie narzędzi do dopasowywania sposobu wyświetlania treści, dodanie wskaźnika fokusa lub odtwarzanie audio treści artykułów.

Rys. 9.8. Przykłady elementów przeznaczonych dla osób niepełnosprawnych oraz element odsyłający do informacji publicznej (BIP)

Wskaźnik fokusa jest to wizualne oznaczenie elementu interaktywnego (takiego jak odnośnik czy przycisk), który w danym momencie ma fokus. Na ogół wskaźnik fokusa ma postać obramowania otaczającego dany element. Jest on niezbędny dla osób korzystających wyłącznie z klawiatury.

Rys. 9.9. Przykład zastosowania wskaźnika fokusa

Niedostosowanie stron do wymagań osób słabowidzących może być spowodowane również przez niewystarczający kontrast między tłem a tekstem.

PRZYKŁAD 9.3.

Formatowanie elementu w celu otrzymania wskaźnika fokusa możemy przeprowadzić w kaskadowym arkuszu stylów CSS według wzoru:

```
button {
 color: #FFF100;
 background: #000000;
 border: 2px solid #000000;
}
button:focus {
 color: #000000;
 background: #FFF100;
 border-style: dashed;
}
```

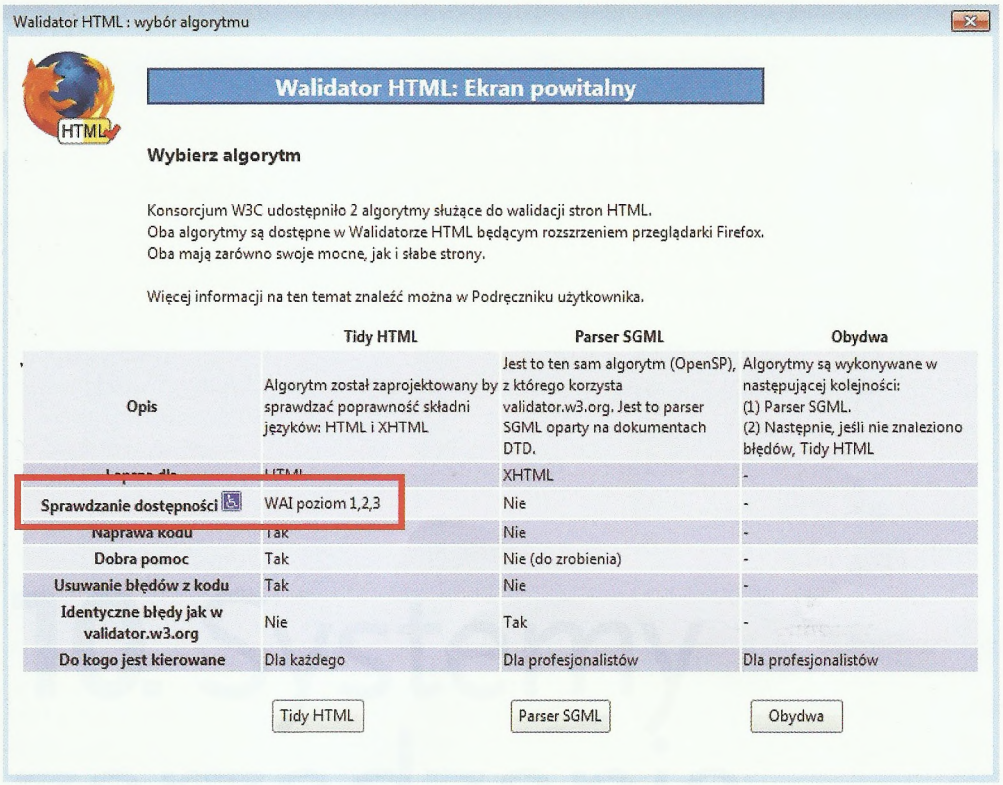
Tego typu elementy należy wziąć pod uwagę na etapie wstępnej analizy wytycznych do projektu strony internetowej.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Dlaczego stosowanie wskaźnika fokusa może zwiększyć dostępność strony internetowej?
2. Co to jest PageRank?
3. Wymień czynniki, które mogą wpłynąć na lepsze pozycjonowanie strony internetowej.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. W opcji walidacji dokumentu hipertekstowego za pomocą parsera przeglądarki internetowej mamy do wyboru kilka wariantów. Na rysunku zaznaczono element sprawdzany przez jeden z parserów. Skorzystaj z zasobów sieci internet i wyjaśnij, jakie obszary kodu będą przez niego sprawdzane na każdym z trzech poziomów.


Walidator HTML : wybór algorytmu

Walidator HTML: Ekran powitalny

Wybierz algorytm

Konsorcjum W3C udostępniło 2 algorytmy służące do walidacji stron HTML.
Oba algorytmy są dostępne w Walidatorze HTML będącym rozszrzeniem przeglądarki Firefox.
Oba mają zarówno swoje mocne, jak i słabe strony.

Więcej informacji na ten temat znaleźć można w Podręczniku użytkownika.

	Tidy HTML	Parser SGML	Obydwa
Opis	Algorytm został zaprojektowany by sprawdzać poprawność składni języków: HTML i XHTML	Jest to ten sam algorytm (OpenSP), z którego korzysta validator.w3.org. Jest to parser SGML oparty na dokumentach DTD.	Algorytmy są wykonywane w następującej kolejności: (1) Parser SGML. (2) Następnie, jeśli nie znaleziono błędów, Tidy HTML
Sprawdzanie dostępności	WAI poziom 1,2,3	Nie	-
Naprawa kodu	Tak	Nie	-
Dobra pomoc	Tak	Nie (do zrobienia)	-
Usuwanie błędów z kodu	Tak	Nie	-
Identyczne błędy jak w validator.w3.org	Nie	Tak	-
Do kogo jest kierowane	Dla każdego	Dla profesjonalistów	Dla profesjonalistów

Tidy HTML | Parser SGML | Obydwa

2. Zapisz trzy propozycje nazwy adresu domenowego drugiego poziomu dla strony internetowej twojej szkoły. Sprawdź na stronie internetowej KRD, czy są one dostępne.

10. Systemy zarządzania treścią CMS

- Rodzaje systemów zarządzania treścią
- Instalacja systemu CMS
- Personalizacja i zarządzanie systemem CMS

10.1 Rodzaje systemów zarządzania treścią

ZAGADNIENIA

- **Definicja CMS**
- **Rodzaje systemów zarządzania treścią**
- **Wybór CMS do określonych zadań**

Co to jest CMS?

System zarządzania treścią **CMS** (ang. *Content Management System*) jest obecnie najczęściej stosowanym rozwiązaniem do tworzenia mniej lub bardziej zaawansowanych witryn internetowych. Przeważnie są to gotowe rozwiązania stanowiące grupę aplikacji internetowych do tworzenia i personalizacji stron internetowych oraz zarządzania nimi. Podstawowym celem oprogramowania CMS jest oddzielenie zawartości informacyjnej strony internetowej od sposobu jej prezentacji. W praktyce wygląda to tak, że uprawniony użytkownik wprowadza nowe treści jako artykuł, który zostaje zapisany w bazie danych. Następnie system zarzadzania treścią generuje dynamicznie strony internetowe na podstawie treści pochodzącej z bazy danych oraz zadeklarowanych szablonów. Pozwala to na bardziej elastyczne, czytelne i wygodniejsze zarządzanie stroną internetową, niż ma to miejsce w przypadku zastosowania statycznych stron internetowych.

Systemy zarządzania treścią są budowane z wykorzystaniem baz danych, serwerów WWW i języków skryptowych wykonywanych po stronie serwera (ang. *server-side*), np. PHP. Do obsługi systemu, np. aktualizacji strony, CMS może wymagać użycia oprogramowania klienckiego, korzystającego z takich protokołów, jak FTP czy SFTP, natomiast do edycji artykułów wbudowanych – edytorów WYSIWYG.

Istotnym elementem budowy systemu CMS są **templatki** (ang. *template*), czyli gotowe szablony strony, które definiują prezentację treści. Wykorzystanie templatek sprawia, że zmiana koncepcji graficznej całego serwisu sprowadza się do przygotowania i zamiany szablonu. W skład systemu CMS wchodzą również gotowe, intuicyjne panele zarządzania stroną. Dzięki temu proces publikacji strony w internecie i jej obsługa stają się wygodniejsze i prostsze. Użytkownik nie musi ingerować bezpośrednio w kod strony, a tym samym nie wymaga się od niego znajomości wszystkich zastosowanych w CMS technologii. Dodatkowo treści na stronie można prezentować jednocześnie w różnych formatach, np. PDF.

Najczęściej systemy CMS są pisane w takich językach, jak: HTML, ASP, JSP, PHP czy Python.

Rodzaje systemów CMS

Obecnie systemy zarządzania treścią są często spotykanymi produktami w projektach internetowych, dlatego możemy spotkać wiele rodzajów tego oprogramowania. Ze względu na to, kto i w jakim celu tworzy CMS oraz jakie są warunki jego rozpowszechniania i licencjonowania, rozróżniamy:

- **systemy komercyjne**
  Płatne systemy i tworzone na zamówienie konkretnego klienta, które nazywane są często systemami o zamkniętym kodzie źródłowym. Charakteryzują się zwiększonym zakresem bezpieczeństwa, indywidualnym szablonem (templatką) oraz okrojoną liczbą dodatków (ang. *widgets*) czy wtyczek (ang. *plugins*).
- **open source**
  Bezpłatne systemy z otwartym kodem źródłowym, oferujące bardzo dużą liczbę dodatków, aplikacji czy wtyczek.
- **systemy własne**
  Autorskie CMS-y do prywatnego stosowania.

Ze względu na budowę systemu CMS rozróżniamy:

- Content Management Frameworks, czyli narzędzie do budowy CMS-ów; jest to zbiór aplikacji potrzebnych do projektowania i programowania systemu CMS;
- Page-Based Systems, czyli systemy o transparentnych konsolach; pozwalają na edycję w ciele strony, nie wymagają konsoli do zarządzania treścią;
- Module-Based Systems, czyli systemy CMS bazujące na modułach; jest to najbardziej popularny rodzaj CMS;
- Content Object Systems, czyli systemy koncentrujące się wokół tzw. obiektów – małych części informacji, które można reprodukować w wielu miejscach strony WWW.

Ze względu na zastosowanie systemów zarządzania treścią rozróżniamy:

- systemy do tworzenia systemów zarządzania treścią (np. Aplin);
- systemy do zarządzania przepływem dokumentów, wiedzy i zasobów;
- systemy do zarządzania treścią informacyjnej witryny internetowej lub blogu (np. Joomla! czy WordPress);
- systemy do prezentacji zdjęć i innych zasobów multimedialnych i zarządzania nimi (np. Gallery);
- systemy do tworzenia sklepów internetowych i zarządzania nimi (np. Magento);
- systemy do organizacji nauczania na odległość (np. Moodle);
- systemy do zarządzania projektami i ich realizacją (np. Redmine).

Wybór systemu CMS

Jeśli weźmiemy pod uwagę opis rodzajów systemów zarządzania treścią, a także ich funkcjonalność, zdamy sobie sprawę, jak ważną decyzją jest wybór właściwego systemu do konkretnego zastosowania.

Poza funkcjonalnością systemu ważne jest, aby sprawdzić wymagania danej wersji CMS dotyczące serwera WWW i obsługiwanych aplikacji (np. wersji serwera bazy danych MySQL czy języka skryptowego PHP).

W przypadku instalacji systemu CMS na serwerze zdalnym warto upewnić się, czy dysponujemy odpowiednią ilością miejsca na dysku i możliwością utworzenia nowej bazy danych.

Systemy zarządzania treścią najczęściej stosujemy do tworzenia i obsługi informacyjnych stron internetowych. Do najpopularniejszych zaliczamy CMS Joomla! i WordPress. Ich popularność wynika głównie ze stabilności pracy, bezpieczeństwa, bezpłatnego dostępu i ogromnej ilości dodatków i rozszerzeń, pozwalających na tworzenie różnorodnych witryn do użytku prywatnego oraz firmowego.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Czym różni się system CMS od statycznej strony internetowej?
2. Do czego służą templatki?
3. Opisz rodzaje systemów CMS.
4. Co decyduje o wyborze systemu CMS?

10.2 Instalacja systemu CMS

ZAGADNIENIA
- **Przygotowanie do instalacji**
- **Rodzaje i przebieg instalacji**

Po wyborze systemu CMS należy przygotować się do jego instalacji. Obecnie mamy różne możliwości publikacji własnej strony internetowej opartej na CMS i do każdej inaczej musimy się przygotować.

Publiczną stronę internetową możemy uruchomić na własnym serwerze WWW lub można wykupić usługę hostingową. W podręczniku opiszemy przykład instalacji systemu WordPress opartej na usłudze profesjonalnego hostingu, tak aby w pełni opisać metodę publikacji strony internetowej na serwerze zdalnym.

Krok 1. Wybór serwera WWW i domeny
Nasza strona musi istnieć w internecie. Dlatego powinniśmy mieć publiczny adres IP i domenę, pod którą będzie widoczna nasza witryna. Najprostszym rozwiązaniem będzie wykupienie usługi hostingowej. Obecnie na rynku jest wiele firm oferujących szeroki zakres usług. Różnią się one liczbą baz danych, pojemnością konta czy limitem transferu. Mogą również oferować w cenie usługi dostęp do domeny i oszczędzić nam pracy z wyborem, opłaceniem i transferem domeny z NASK, czyli instytutu krajowego rejestru nazw internetowych.

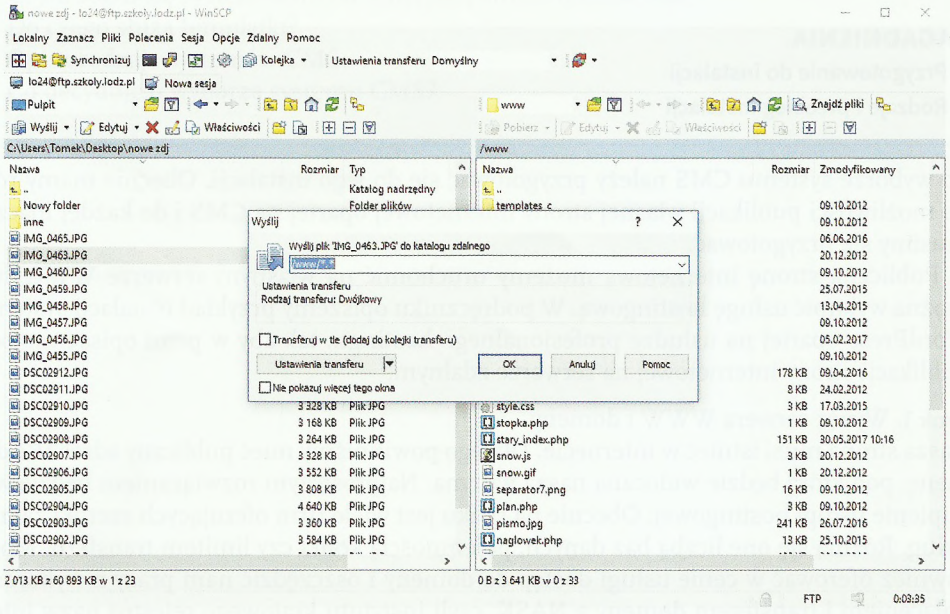
Krok 2. Przygotowanie bazy danych i wgranie plików instalacyjnych
Po zarejestrowaniu usługi i otrzymaniu własnej domeny przystępujemy do przygotowania instalacji na naszym serwerze witryny opartej na systemie CMS.

UWAGA

Jeżeli mamy domenę kupioną i zarejestrowaną w NASK, należy delegować ją na serwery DNS naszego obecnego usługodawcy. W tym celu należy zadeklarować w konfiguracji domeny adresy DNS operatora, do którego delegujemy domenę, a następnie przypisać domenę do wykupionej usługi. Zazwyczaj do jednej usługi hostingowej można przypisać wiele domen.

W panelu konfiguracyjnym naszej usługi hostingowej odnajdujemy opcję dodania nowej strony internetowej i tworzymy dla niej oddzielny katalog oraz przypisujemy domenę. Jeżeli na naszym serwerze WWW można utworzyć tylko jedną stronę internetową i jedną bazę danych, wówczas nie musimy tworzyć podkatalogu, a pliki instalacyjne skopiujemy do katalogu głównego usługi.

Jeżeli mamy katalog gotowy do wprowadzenia na stronę internetową, możemy przystąpić do kopiowania plików systemu CMS. Przygotowujemy plik instalacyjny WordPress, który można pobrać z polskiej strony projektu **http://www.wordpress-polska.pl/**. Pobrany skompresowany plik musimy rozpakować. Tak przygotowane pliki należy skopiować do przygotowanego folderu na zdalnym serwerze za pomocą klienta FTP. Można do tej czynności wykorzystać bezpłatne aplikacje, np. WinSCP. Wszystkie informacje dotyczące konfiguracji klienta FTP znajdziemy w panelu administratora na profilu naszej usługi hostingowej.



Rys. 10.1. Proces kopiowania plików na zdalny serwer za pomocą WinSCP

Innym możliwym sposobem jest skorzystanie z powszechnie spotykanej usługi o nazwie **Autoinstalator**. Usługa ta polega na wykorzystaniu gotowych do instalacji systemów CMS, przygotowanych przez naszego usługodawcę hostingu. Opiszemy to w następnym kroku.

Krok 3. Instalacja systemu CMS

Mamy możliwość zastosowania trzech metod instalacji systemu WordPress:

- instalacja z edycją pliku wp-config.php;
- instalacja bez edytowania żadnych plików – wszelkie dane do połączenia z bazą podaje użytkownik podczas instalacji;
- instalacja za pomocą autoinstalatora, bez jakiejkolwiek ingerencji użytkownika.

W przypadku zastosowania pierwszej metody wszystkie parametry instalacji (nazwa bazy danych, adres serwera SQL, nazwa użytkownika bazy i hasło dostępu do bazy) muszą zostać wpisane do pliku wp-config.php. Edycję pliku możemy wykonać, korzystając z dowolnego edytora plików tekstowych.

Jeśli wybierzemy drugą metodę, należy uruchomić instalację poprzez wpisanie adresu strony w przeglądarce. W zależności od tego, czy pliki WordPressa zostały skopiowane do głównego katalogu domeny, czy też do dedykowanego podkatalogu, adres powinien wyglądać następująco:

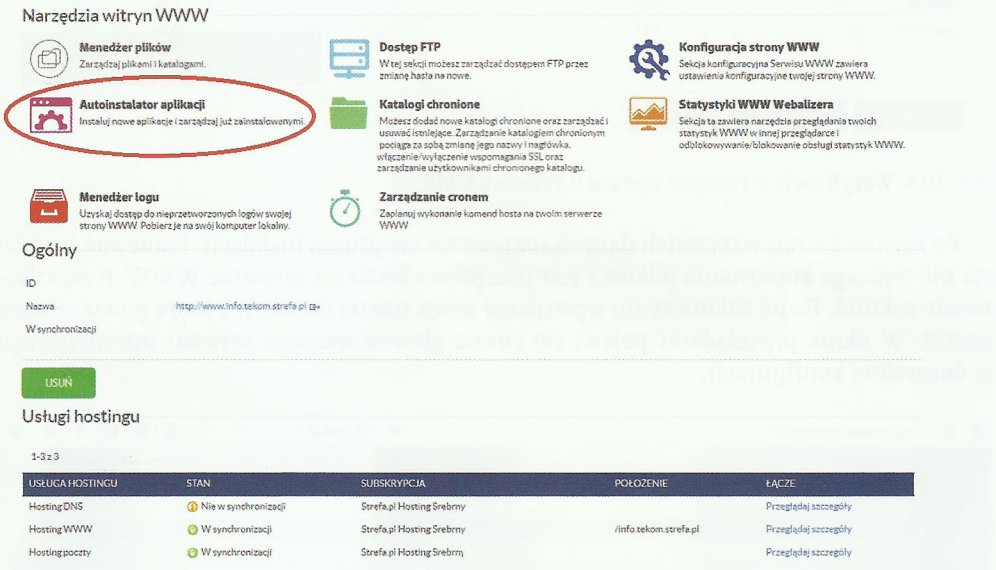
`http://www.nasza-domena.pl/wp-admin/install.php`

lub

`http://www.nasza-domena.pl/nazwa-katalogu/wp-admin/install.php`

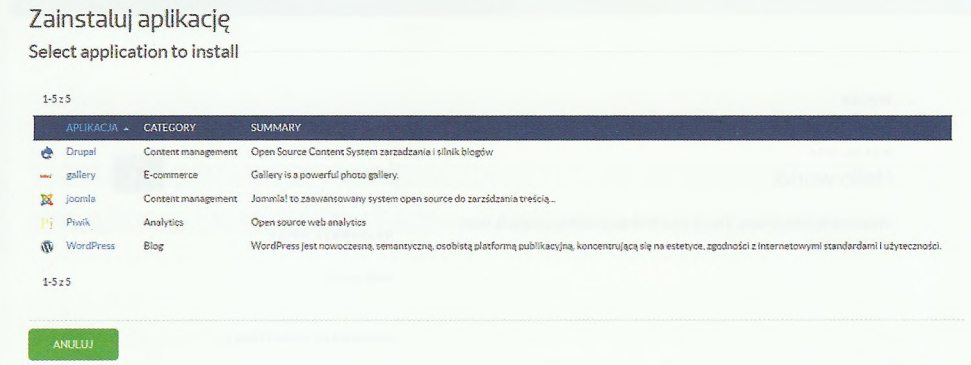
Po uruchomieniu instalacji postępujemy zgodnie z kolejnymi instrukcjami, czyli podajemy takie informacje, jak tytuł strony, login i hasło do konta superadministratora.

Trzecią metodą jest wykorzystanie autoinstalacji, dostępnej w wielu kontach hostingowych. Autoinstalator znajduje się najczęściej w karcie narzędzi witryn WWW. Na rysunku 10.2 pokazano przykład okna narzędzi witryn WWW w hostingu strefa.pl.



Rys. 10.2. Wybór autoinstalatora systemów CMS

Po wyborze autoinstalatora aplikacji określamy, jaką aplikację chcemy zainstalować.

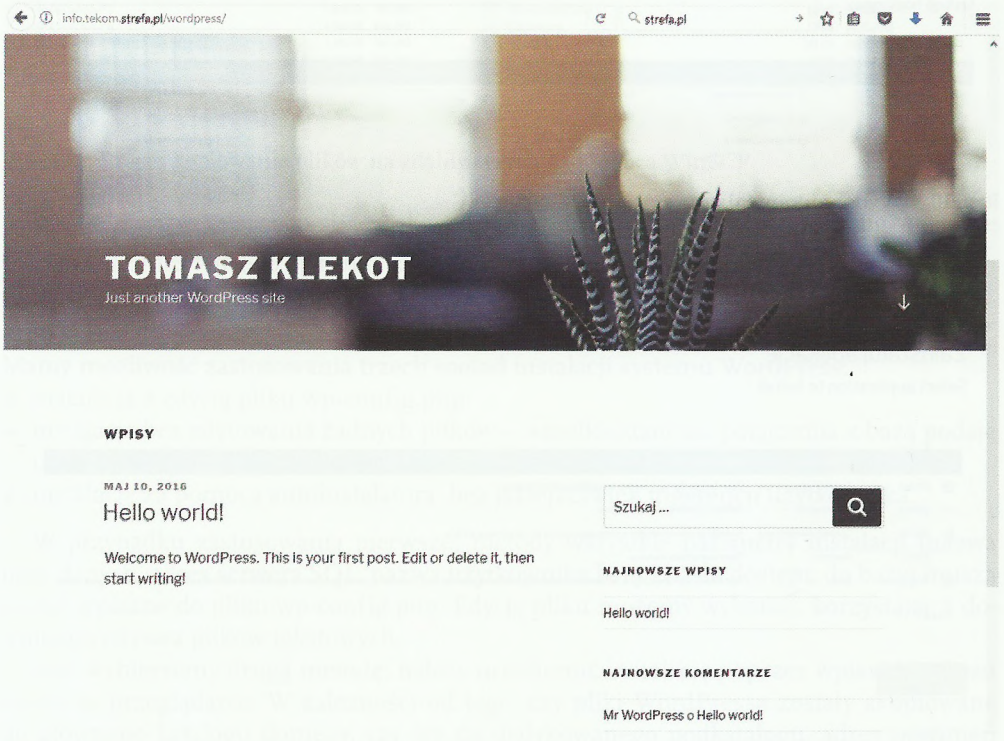


Rys. 10.3. Okno wyboru aplikacji do instalacji

Autoinstalator sam sprawdzi, czy parametry techniczne są spełnione, i w dalszej części procedury instalacyjnej poprosi o dane dotyczące nazwy katalogu do instalacji, domeny, z którą serwis ma być powiązany, oraz nazwy i hasła superadministratora.

Rys. 10.4. Weryfikacja wymagań instalacji systemu CMS

Po zatwierdzeniu wszystkich danych rozpocznie się proces instalacji. Ponieważ instalacja nie wymaga kopiowania plików i jest przeprowadzana na serwerze WWW, trwa kilkanaście sekund. Po jej zakończeniu wpisujemy adres naszej domeny, z którą powiązaliśmy serwis. W oknie przeglądarki pojawi się strona główna naszego serwisu internetowego w domyślnej konfiguracji.



Rys. 10.5. Domyślne okno systemu CMS po instalacji

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj proces instalacji systemu CMS.
2. Omów metody instalacji systemu CMS.

10.3 Personalizacja i zarządzanie systemem CMS

ZAGADNIENIA

- **Logowanie do panelu administratora**
- **Personalizacja witryny**
- **Edycja treści w CMS**

Po instalacji systemu WordPress uruchamia się witryna oparta na domyślnych ustawieniach i z domyślną templatką.

Każda struktura CMS składa się z dwóch części nazywanych **front end** i **back end**. Front end to widoczna dla użytkowników witryna internetowa, back end to warstwa administracyjna witryny, która jest dostępna dla zalogowanych do panelu administracyjnego.

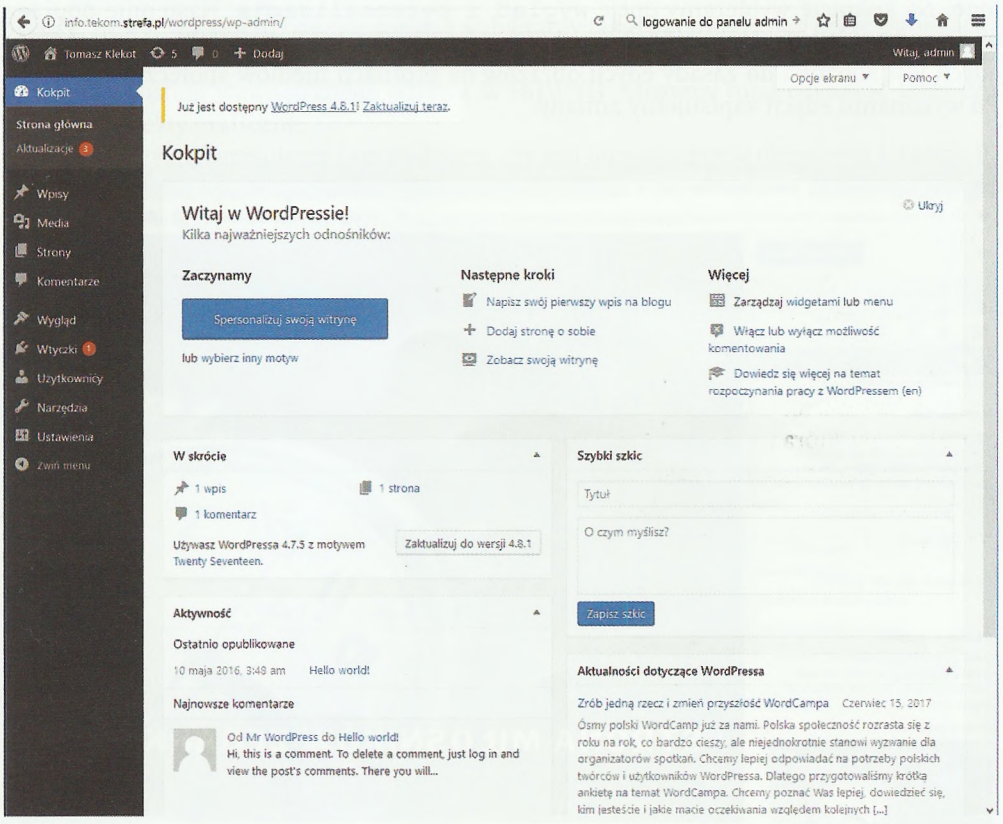
Dlatego aby zmienić wygląd, edytować treści czy wykonywać inne czynności, musimy zalogować się do panelu administratora. W systemie WordPress należy w pasku adresu przeglądarki internetowej wpisać adres według następującego szablonu:

```
http://www.nasza-domena.pl/nazwa-katalogu/wp-admin
```

Po zatwierdzeniu pojawi się okno logowania do panelu administratora. Tutaj należy wpisać nazwę użytkownika i hasło ustalone podczas instalacji systemu.

Rys. 10.6. Logowanie do panelu administratora

Po zalogowaniu otwiera się okno kokpitu użytkownika o uprawnieniach superadministratora (rys. 10.7). Uprawnienia administratorów mogą się różnić (mogą być większe lub mniejsze). Superadministrator ma najwyższe uprawnienia – dla całego zaplecza strony. Można również utworzyć profile administratorów – redaktorów z uprawnieniami do tworzenia artykułów, wstawiania grafik itp. Ułatwia to pracę po stronie back endu witryny.



Rys. 10.7. Kokpit systemu WordPress

Po rozpoczęciu pracy w panelu administracyjnym możemy skorzystać z krótkiego kursu obsługi lub wykonać samodzielnie personalizację witryny. Do najważniejszych czynności, które powinniśmy wykonać po instalacji systemu CMS jako superadministrator, należą:

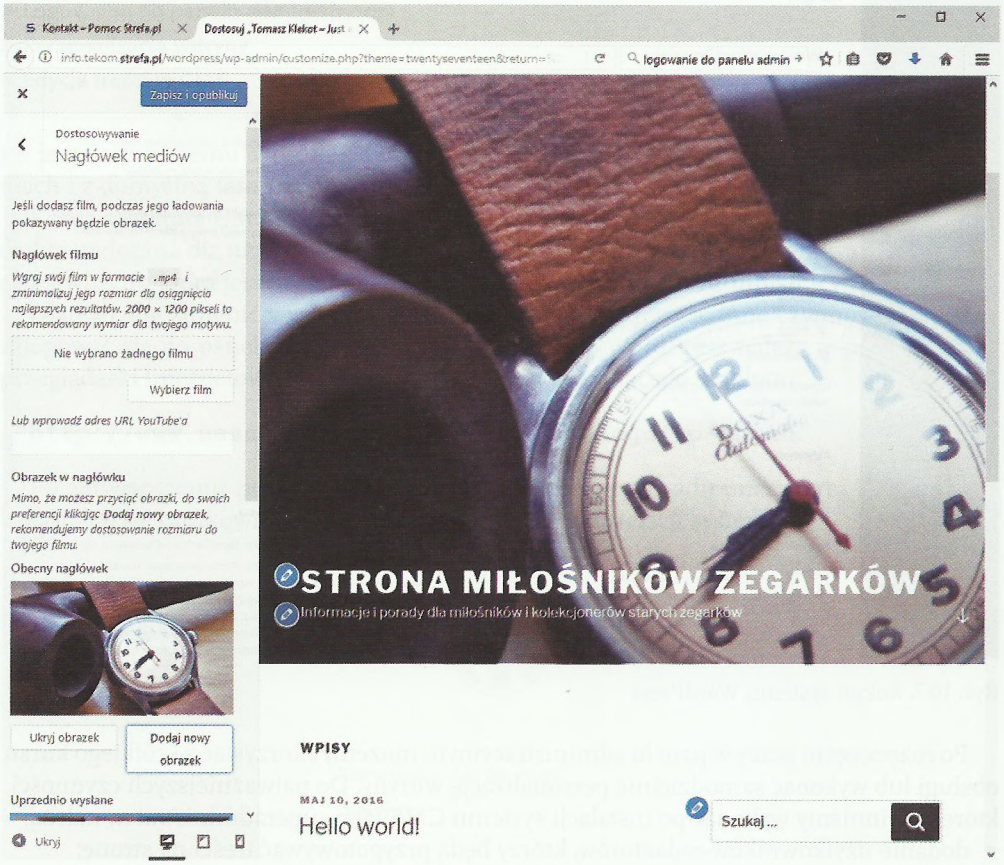
- dodanie użytkowników-redaktorów, którzy będą przygotowywać treści na stronę;
- personalizacja wyglądu witryny, czyli edycja lub zmiana motywu (templatki) oraz edycje jej ustawień;
- dodanie kluczowych elementów graficznych, jak ikona strony, logo itp.;
- ustalenie właściwego szablonu strony, menu i innych elementów;
- sprawdzenie aktualizacji i ewentualne wykonanie kopii zapasowej serwisu.

Nie będziemy opisywać wszystkich czynności związanych z zarządzaniem systemem CMS. Skupimy się tylko na dopasowaniu wyglądu motywu strony oraz redakcji artykułu na stronę. Każda grupa tworząca dany system zarządzania treścią oferuje na swoich stronach obszerne porady i samouczki, które pozwalają dokładnie poznać CMS.

Personalizacja wyglądu witryny

Aby zaprojektować oprawę graficzną witryny, musimy wcześniej przygotować odpowiednie pliki. Co prawda system umożliwia docinanie obrazów do określonego zastosowania w motywie, ale są to narzędzia proste i nie gwarantują poprawnej edycji obrazu.

Na naszą stronę przygotowaliśmy pliki graficzne do zastosowania jako ikona oraz logo. Na kokpicie wybieramy opcję **`wygląd i personalizacja`**. Następnie poprzez wybór odpowiedniej ikony ołówka wykonujemy edycję określonego elementu. Zasada edycji jest zbliżona do zasady edycji obrazów na profilach mediów społecznościowych. Po wykonaniu edycji zapisujemy zmiany.



Rys. 10.8. Personalizacja motywu witryny

Redagowanie artykułów

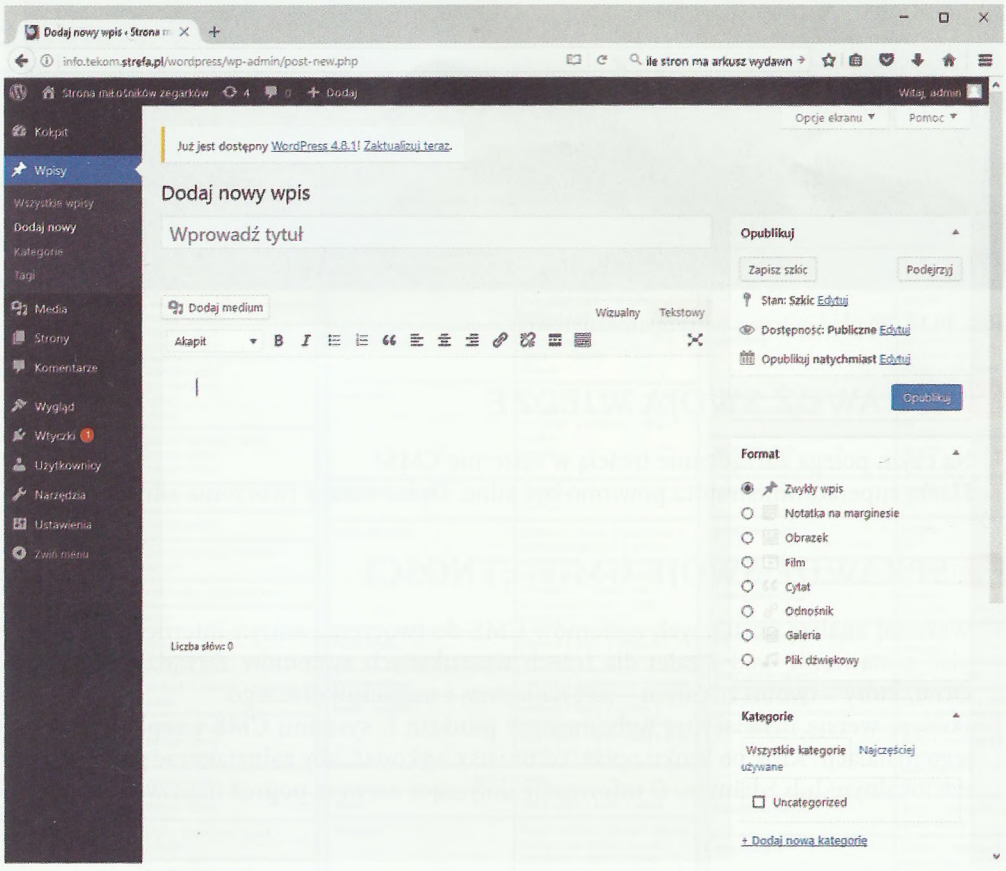
Zarządzanie treścią w systemie CMS sprowadza się głównie do dodawania artykułów oraz wpisów i zarządzania nimi. Wpisy mogą tworzyć stronę startową lub listę aktualności albo wyświetlać się w postaci pojedynczego artykułu. Aby prawidłowo zarządzać systemem CMS, musimy stworzyć sobie obraz struktury fizycznej witryny w postaci przyporządkowania artykułów do elementów menu.

Podczas tworzenia menu i jego elementów w postaci odnośników określamy, które będą otwierać pojedynczy artykuł, które listę artykułów. Kluczowe dla przyporządkowania artykułu do elementów strony jest nadanie nazwy oraz dołączenie go do określonej kategorii.

Dlatego też przed przystąpieniem do edycji wpisów należy utworzyć określone ich kategorie (np. aktualności, archiwum, dla_ucznia itp.).

Po utworzeniu kategorii możemy przystąpić do edycji artykułów. Po wyborze opcji **`Dodaj nowy`**, wpisujemy nazwę wpisu i wybieramy takie parametry, jak opcje publikacji, format wpisu czy kategorię. Sama edycja wpisu jest intuicyjna i prosta, ponieważ sprowadza się do obsługi wewnętrznego edytora WYSIWYG. Możemy tutaj edytować tekst czy wstawiać obiekty graficzne.

Gotowy artykuł zapisujemy i sprawdzamy, czy jest on widoczny w określonej lokalizacji. Jeżeli artykuł ma być bezpośrednio powiązany z odsyłaczem elementu menu, należy go wybrać jako związany w oknie edycji danego elementu menu.



Rys. 10.9. Okno edycji wpisu

Po wykonaniu personalizacji witryny oraz uzupełnieniu treści możemy sprawdzić ostateczny wygląd naszego CMS od strony front end – wpisać adres domenowy.

Cascading Style Sheets (CSS 3)

TEXT	
direction	ltr | rtl | inherit
hanging-punctuation	none | [start | end | end-edge]
letter-spacing	normal *length* %
punctuation-trim	none | [start | end | adjacent]
text-align	start | end | left | right | center | justify
text-align-last	start | end | left | right | center | justify
text-decoration	none | underline | overline | line-through | blink
text-emphasis	none | [[accent | dot | circle | disc] [before | after]?]
text-indent	*length* %
text-justify	auto | inter-word | inter-ideograph | inter-cluster | distribute | kashida | tibetan
text-outline	none *color* *length*
text-shadow	none *color* *length*
text-transform	none | capitalize | uppercase | lowercase
text-wrap	normal | unrestricted | none | suppress
unicode-bidi	normal | embed | bidi-override
white-space	normal | pre | nowrap | pre-wrap | pre-line
white-space-collapse	preserve | collapse | preserve-breaks | discard
word-break	normal | keep-all | loose | break-strict | break-all
word-spacing	normal *length* %
word-wrap	normal | break-word

COLUMN	
column-count	auto *number*
column-fill	auto | balance
column-gap	normal *length*
column-rule	*column-rule-width* *column-rule-style* *column-rule-color*
column-rule-color	*color*
column-rule-style	*border-style*
column-rule-width	thin | medium | thick *length*
columns	*column-width* *column-count*
column-span	1 | all
column-width	auto *length*

COLOR	
color	inherit *color*
opacity	inherit *number*

TEMPLATE LAYOUT	
box-align	start | end | center | base-
box-direction	normal | reverse
box-flex	*number*
box-flex-group	*integer*
box-lines	single | multiple
box-orient	horizontal | vertical | inline-axis | block-axis
box-pack	start | end | center | justify
box-sizing	content-box | padding-box | border-box | margin-box
tab-side	top | bottom | left | right

TABLE	
border-collapse	collapse | separate
border-spacing	*length length*
caption-side	top | bottom | left | right
empty-cells	show | hide
table-layout	auto | fixed

SPEECH	
cue	*cue-before* *cue-after*
cue-before	*uri* [silent | x-soft | soft | medium | loud | x-loud] | none | inherit] *number* %
cue-after	*uri* [silent | x-soft | soft | medium | loud | x-loud] | none | inherit] *number* %
mark	*mark-before* *mark-after*
mark-before	*string*
mark-after	*string*
pause	*pause-before* *pause-after*
pause-before	none | x-weak | weak | medium | strong | x-strong | inherit *time*
pause-after	none | x-weak | weak | medium | strong | x-strong | inherit *time*
phonemes	*string*
rest	*rest-before* *rest-after*
rest-before	none | x-weak | weak | medium | strong | x-strong | inherit *time*
rest-after	none | x-weak | weak | medium | strong | x-strong | inherit *time*
speak	none | normal | spell-out | digits | literal-punctuation | no-punctuation | inherit
voice-balance	left | center | right | leftwards | rightwards | inherit *number*
voice-duration	*time*

SPEECH	
voice-family	inherit | [<specific-voice, age, generic-voice, number>]
voice-rate	x-slow | slow | medium | fast | x-fast | inherit %
voice-pitch	x-low | low | medium | high | x-high | inherit *number* %
voice-pitch-range	x-low | low | medium | high | x-high | inherit *number*
voice-stress	strong | moderate | none | reduced | inherit
voice-volume	silent | x-soft | soft | medium | loud | x-loud | inherit *number* %

LIST & MARKERS	
list-style	*list-style-type* *list-style-position* *list-style-image*
list-style-image	none *url*
list-style-position	Inside | outside
list-style-type	none | asterisks | box | check | circle | diamond | disc | hyphen | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-alpha | upper-alpha | lower-greek | lower-latin | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha | footnotes
marker-offset	auto *length*

ANIMATIONS	
animation	*animation-name* *animation-duration* *animation-timing-function* *animation-delay* *animation-iteration-count* *animation-direction*
animation-delay	*time*
animation-direction	normal | alternate
animation-duration	*time*
animation-iteration-count	inherit *number*
animation-name	none | IDENT
animation-play-state	running | paused
animation-timing-function	ease | linear | ease-in | ease-out | ease-in-out | cubic-Bezier (*number, number, number, number*)

TRANSITIONS	
transition	*transition-property* *transition-duration* *transition-timing-function* *transition-delay*
transition-delay	*time*
transition-duration	*time*
transition-property	none | all
transition-timing-function	ease | linear | ease-in | ease-out | ease-in-out | cubic-Bezier (*number, number, number, number*)

Cascading Style Sheets (CSS 3)

GRID POSITIONING	
grid-columns	none | inherit *[length percentage relative length]*
grid-rows	none | inherit *[length percentage relative length]*

OUTLINE	
outline	*outline-color* *outline-style* *outline-width*
outline-color	*color* invert
outline-offset	inherit *length*
outline-style	None | dotted | dashed | solid | double | groove | ridge | inset | outset
outline-width	thin | medium | thick *length*

3D / 2D TRANSFORM	
backface-visibility	visible | hidden
perspective	none *number*
perspective-origin	*[[[percentage> | <length> | left | center | right] [<percentage> | <length> | top | center | bottom]?] <length>] | [[[left | center | right] || [top | center | bottom]] <length>]*
transform	none | matrix | matrix3d | translate3d | translateX | translateY | translateZ | scale | scale3d | scaleX | scaleY | scaleZ | rotate | rotate3d | rotateX | rotateY | rotateZ | skewX | skewY | skew | perspective
transform-origin	*[[[<percentage> | <length> | left | center | right] [<percentage> | <length> | top | center | bottom]?] <length>] | [[[left | center | right] || [top | center | bottom]] <length>]*
transform-style	flat | preserve-3d

GENERATED CONTENT	
bookmark-label	content *attr* *string*
bookmark-level	none *integer*
bookmark-target	self *uri* *attr*
border-length	auto *length*
content	normal | none | inhibit *uri*
counter-increment	none *identifier number*
counter-reset	none *identifier number*
crop	auto *shape*
display	normal | none | list-item
float-offset	*length length*

GENERATED CONTENT	
hyphenate-after	auto *integer*
hyphenate-before	auto *integer*
hyphenate-character	auto *string*
hyphenate-lines	no-limit *integer*
hyphenate-resource	none *uri*
hyphens	none | manual | auto
image-resolution	normal | auto *dpi*
marks	[crop || cross] | none
move-to	normal | here *identifier*
page-policy	start | first | last
quotes	none *string string string string*
string-set	*identifier* *content-list*
text-replace	none *[<string> <string>]+*

LINE BOX	
alignment-adjust	auto | baseline | before-edge | text-before-edge | middle | central | after-edge | text-after-edge | ideographic | alphabetic | hanging | mathematical *length* %
alignment-baseline	baseline | use-script | before-edge | text-before-edge | after-edge | text-after-edge | central | middle | ideographic | alphabetic | hanging | mathematical
baseline-shift	baseline | sub | super *length* %
dominant-baseline	auto | use-script | no-change | reset-size | alphabetic | hanging | ideographic | mathematical | central | middle | text-after-edge | text-before-edge
drop-initial-after-align	*alignment-baseline*
drop-initial-after-adjust	central | middle | after-edge | text-after-edge | ideographic | alphabetic | mathematical *length* %
drop-initial-before-align	caps-height *alignment-baseline*
drop-initial-before-adjust	before-edge | text-before-edge | central | middle | hanging | mathematical *length* %
drop-initial-value	initial *integer*
drop-initial-size	auto *integer* % *line*
inline-box-align	initial | last *integer*

LINE BOX	
line-height	normal *number* *length* %
line-stacking	*line-stacking-strategy* *line-stacking-ruby* *line-stacking-shift*
line-stacking-strategy	inline-line-height | block-line-height | max-height | grid-height
line-stacking-ruby	exclude-ruby | include-ruby
line-stacking-shift	consider-shifts | disregard-shifts
text-height	auto | font-size | text-size | max-size
vertical-align	Baseline | sub | super | top | text-top | middle | bottom | text-bottom *length* %

HYPERLINK	
target	*target-name* *target-new* *target-position*
target-name	current | root | parent | new | modal *string*
target-new	window | tab | none
target-position	above | behind | front | back

POSITIONING	
bottom	auto % *length*
clip	*shape* auto
left	auto % *length*
position	static | relative | absolute | fixed
right	auto % *length*
top	auto % *length*
z-index	auto *number*

RUBY	
ruby-align	auto | start | left | center | end | right | distribute-letter | distribute-space | line-edge
ruby-overhang	auto | start | end | none
ruby-position	before | after | right | inline
ruby-span	attr(x) | none

Cascading Style Sheets (CSS 3)

PAGED MEDIA	
fit	fill | hidden | meet | slice
fit-position	[top | center | bottom] || [left | center | right] *length* %
image-orientation	auto *angle*
orphans	*integer*
page	auto *identifier*
page-break-after	auto | always | avoid | left | right
page-break-before	auto | always | avoid | left | right
page-break-inside	auto | avoid
size	auto | landscape | portrait *length*
windows	*integer*
UI	
appearance	normal | inherit | [icon | window | desktop | work-space | document | tooltip | dialog | button | push-button | hyperlink | radio-button | checkbox | menu-item | tab | menu | menubar | pull-down-menu | pop-up-menu | list-menu | radio-group | checkbox-group | outline-tree | range | field | combo-box | signature | password]
cursor	auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help *url*
icon	auto | inherit *url*
nav-index	auto | inherit *number*
nav-up	auto | inherit *<id> [current | root | <target-name>]*
nav-right	auto | inherit *<id> [current | root | <target-name>]*
nav-down	auto | inherit *<id> [current | root | <target-name>]*
nav-left	auto | inherit *<id> [current | root | <target-name>]*
resize	none | both | horizontal | vertical | inherit

- Values in italics are place holders for an actual value (like 1px, 1em, 1%), values in normal text are values that can be used as the actual value
- CSS Properties in **Dark Red** are shorthand properties and each value must be defined. The exception is where the property can define from one to four of the sides of a box element property (Top-Right-Bottom-Left) - i.e. border-width

Cascading Style Sheets (CSS 3)

PSEUDO-CLASS	
:active	an activated element
:focus	an element while the element has focus
:visited	a visited link
:hover	an element when you mouse over it
:link	an unvisited link
:disabled	an element while the element is disabled
:enabled	an element while the element is enabled
:checked	an element (form element control) that is checked
:selection	an element that is currently selected of highlighted by the user
:lang	Allows the author to specify a language to use in a specified element
:nth-child(n)	an element that is the n-th sibling
:nth-last-child(n)	an element that is the n-th sibling counting from the last sibling
:first-child	an element that is the first sibling
:last-child	an element that is the last sibling
:only-child	an element that is the only child
:nth-of-type(n)	an element that is the n-th sibling of its type.
:nth-last-of-type(n)	an element that is the n-th sibling of its type counting from the last sibling
:last-of-type	an element that is the first sibling of its type
:first-of-type	an element that is the last sibling of its type
:only-of-type	an element that is the only child of that type
:empty	an element that has no children
:root	root element within the document
:not(x)	an element not represented by the argument 'x'
:target	a target element as specified by a target in a URI
PSEUDO-ELEMENT	
::first-letter	Adds special style to the first letter of a text
::first-line	Adds special style to the first line of a text
::before	Inserts some content before an element
::after	Inserts some content after an element

UNITS	
ABSOLUTE MEASUREMENT	
%	percentage
cm	centimeter
in	inch
mm	millimeter
pc	pica (1p = 12 points)
pt	point (1pt = 1/72 inch)
RELATIVE MEASUREMENT	
ch	width of the "0" glyph found in the font for the font size used to render
em	1em = current font size of current element
ex	x-height of the element's font
gd	the grid defined by 'layout-grid'
px	pixel of the viewing device
rem	the font size of the root element
vh	the viewport's height
vw	the viewport's width
vm	viewport's height or width, whichever is smaller of the two
ANGLES	
deg	degrees
grad	grads
rad	radians
turn	turns
TIME	
ms	milli-seconds
s	seconds
FREQUENCY	
Hz	hertz
kHz	kilo-hertz
COLORS	
color name	red, blue, green, dark green
rgb(x,y,z)	red = rgb(255,0,0)
rgb(x%,y%,z%)	red = rgb(100%,0,0)
rgba(x,y,z, alpha)	red = rgba(255,0,0)
#rrggbb	red = #ff0000 (or shorthand = #f00)
hsl(hue, saturation, lightness)	red = hsl(0, 100%, 50%)
hsla(hue, saturation, lightness, alpha)	red = hsl(0, 100%, 50%)
flavor	An accent color (typically chosen by the user) to customize the user interface of the user agent itsel
currentColor	computed value of the 'currentColor' keyword is the computed value of the 'color' property

SELECTOR TYPES		
Name	**Info**	**Example**
Universal	Any element	* { font: 10Px Arial; }
Type	Any element of that type	h1 { text-decoration: underline; }
Grouping	Multiple elements of different types	h1, h2, h3 { font-family: Verdana; }
Class	Multiple elements of different types when you don't want to affect all instances of that type	.sampleclass { text-decoration: underline; }
Id	A single element type when you don't want to affect all instances of that type	#sampleid { text-decoration: underline; }
Descendant	An element that is below (in the document tree) another element—no matter how many levels below	#gallery h1 { text-decoration: underline; }
Child	An element that is directly below (in the document tree) another element	#title > p { font-weight: bold; }
Adjacent Sibling	All elements that share the same parent and elements are in the same immediate sequence	h1 + p { font-style: italic; }
General Sibling	All elements that share the same parent and elements are in the same sequence (not necessarily immediate)	h1 ~ p { font-style: italic; }
Attribute	An element with that matches the attribute listed	E[selected] - att whatever the value E[att="val"] - att with a specific value E[rel~="next"] - att with a value is a whitespace separated list *[lang|="en"] - att value either being exactly "val" or beginning with "val" immediately followed by "-" E[att^="val"] - att value that begins with the prefix "val" E[att$="val"] - att value that end with the suffix "val" E[att*="val"] - att value contains at least one instance of the substring "val"

WYKAZ PODSTAWOWYCH POJĘĆ W JĘZYKACH POLSKIM, ANGIELSKIM I NIEMIECKIM

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
animacja poklatkowa	stop-motion animation	Stop – Motion
argument	argument	das Argument / der Operand
atrybut	attribute	das Attribut
blok preformatowany	preformatted block	der <pre>-Block / der vorformatierte Block
blok tekstu	block of text	der Textblock
deklaracja	declaration	die Deklaration
dokument hipertekstowy	hypertext document	der Hypertext
dynamiczna strona internetowa	dynamic website	die dynamische Webseite
dźwięk	sound	der Klang
element liniowy	linear element	das Inline-Element
filtr	filter	der Filter
formularz	form	das Formular
fotomontaż	photomontage	die Fotomontage
funkcja	function	die Funktion
głębia barwy	color depth	die Farbtiefe
grafika rastrowa	raster graphics	die Bitmapgrafik / die Rastergrafik / die Pixelgrafik
grafika wektorowa	vector graphics	die Vektorgrafik
hipertekstowy język znaczników	hypertext markup language	die Hypertext-Auszeichnungssprache / die Web-Programmiersprache
implementacja	implementation	die Implementierung
instrukcja sterująca	control instruction	die Steuerungsanweisung
internet wszechrzeczy	internet of all things	das Internet der Dinge
kadrowanie	cropping	das Cropping / das Abschneiden
kaskadowy arkusz stylów	cascading style sheet	der Kaskadenstilentwurf
kodek	codec	der Codec
kodowanie polskich znaków	Polish character encoding	die Codierung der polnischen Sonderzeichen
koło kolorów	color wheel	der Farbenkreis

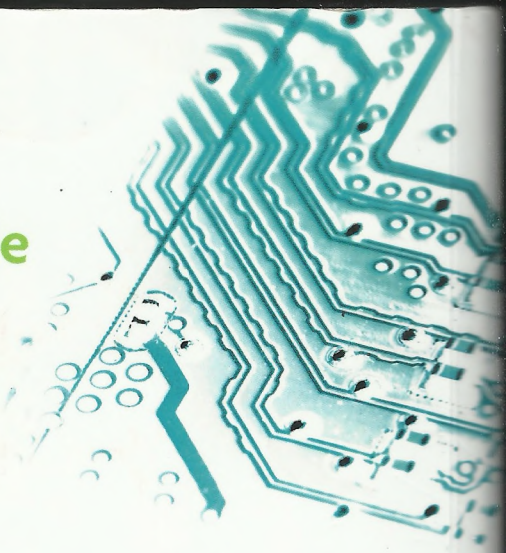
JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
kompresja obrazu	image compression	die Bildkompression / die Bildkomprimierung
konkatenacja	concatenation	die Verkettung
konstruktor	constructor	der Konstruktor
kontener grafiki	graphic container	der Grafikcontainer
konwersja formatu	format conversion	die Formatkonvertierung
kwantyzacja	quantization	die Quantisierung
mapa odsyłaczy	map of links	die Imagemap / die verweissensitive Grafik
media strumieniowe	streaming media	die Streaming Medien
menu nawigacyjne	navigation menu	das Navigationsmenü
metoda	method	die Methode
model barw	color model	das Farbmodell
model blokowy	block model	das Boxmodell
motyw	theme	das Motiv
obiekt	object	das Objekt
obraz cyfrowy	digital image	das digitale Bild
obszar kompozycji	area of the composition	die Zeichenfläche
odnośnik	reference	der Link / der Verweis
okno decyzyjne	decision window	das Confirm-Fenster / das Bestätigungsfeld
okno dialogowe	dialogue window	das Dialogfeld / das Dialogfenster / die Dialogbox
okno informacyjne	information window	das Informationsfenster
okno tekstowe	text window	das Textfeld / das Textfenster
operator	operator	der Operator
pętla	loop	die Schleife
piksel	pixel	das Pixel / das Bildelement/ der Bildpunkt / der Rasterpunkt
plaster	plaster	der Bildteil
pozycjonowanie	positioning	die Positionierung

JĘZYK POLSKI	JĘZYK ANGIELSKI	JĘZYK NIEMIECKI
protokół sieciowy	network protocol	das Netzwerkübertragungs-protokoll / das Netzwerk-protokoll
próbkowanie	sampling	die Abtastung
przełamanie wiersza	break the line	der Zeilenumbruch
ramka kluczowa	key frame	das Schlüsselbild
retusz	retouch	die Retusche
rozdzielczość	resolution	die Auflösung
ruch po ścieżce	movement along the path	die Pfad-Animation
selektor	selector	der Selektor
skalowanie	calibration	das Skalieren / die Skalierung
statyczna strona internetowa	static website	die statische Webseite
struktura fizyczna	physical structure	die physikalische Topologie
System Zarządzania Treścią	content Management System	das Inhaltsverwaltungssystem
technologia webowa	web technology	die Webtechnologie
templatka	template	das Webtemplate / die Webvorlage
testowanie	testing	der Softwaretest
urządzenie mobilne	mobile device	das Mobilgerät
walidacja	validation	die Validierung
warstwa	layer	die Schicht
weryfikacja	verification	die Verifizierung / die Nachprüfung
witryna internetowa	website	die Website
wskaźnik fokusa	focus indicator	das aktive Fenster/ das fokussierte Fenster
wykaz	list	die Liste
zdarzenie	happening	das Ereignis
zmienna	variable	die Variable

REFORMA 2017

Programowanie i tworzenie stron internetowych oraz baz danych i administrowanie nimi

Część 2



Kwalifikacja EE.09

Podręcznik opracowany **zgodnie z nową podstawą programową** kształcenia w zawodzie **technik informatyk**.

Podręcznik do nauki zawodu technik informatyk realizuje treści z zakresu **kwalifikacji EE.09 Programowanie, tworzenie i administrowanie stronami internetowymi i bazami danych**.

W publikacji znalazły się wiadomości dotyczące projektowania i tworzenia stron WWW w języku HTML, edytorów HTML spełniających założenia WYSIWYG, kaskadowych arkuszy stylów CSS, grafiki stron internetowych, walidacji i testowania stron internetowych a także pozycjonowania i optymalizacji stron internetowych, systemów zarządzania treścią CMS, skryptów po stronie klienta – JavaScript oraz skryptów po stronie serwera – PHP. Autorzy wsparli wiadomości teoretyczne licznymi przykładami, ćwiczeniami, bogatym materiałem ilustracyjnym oraz schematami i zrzutami ekranowymi.

Kwalifikacje w ramach zawodu **technik informatyk**

Kwalifikacja **EE.08**	Kwalifikacja **EE.09**
Montaż i eksploatacja systemów komputerowych, urządzeń peryferyjnych i sieci	Programowanie, tworzenie i administrowanie stronami internetowymi i bazami danych

WSiP WYDAWNICTWA SZKOLNE i PEDAGOGICZNE

wsip.pl
sklep.wsip.pl
infolinia: 801 220 555

ISBN 978-83-02-17362-2