



Część 3

Kwalifikacja E.14 Tworzenie aplikacji internetowych



Podręcznik do nauki zawodu
technik informatyk

Jolanta Pokorska



Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych do kształcenia w zawodzie technik informatyk, na podstawie opinii rzeczoznawców: mgr Marii Dziurzyńskiej-Ścibior, mgr Elżbiety Leszczyńskiej, dr. inż. Stanisława Szablowskiego.

Nazwa kwalifikacji: Kwalifikacja E-14. Część 3. Tworzenie aplikacji internetowych.

Typ szkoły: technikum, szkoła policealna, kurs kwalifikacyjny.

Rok dopuszczenia 2014.

Numer ewidencyjny w wykazie: 13/2014

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Marcin Borecki

Projekt okładki: Maciej Pasek

Fotografia na okładce została wykorzystana za zgodą Shutterstock.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie?e14te3>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-8132-7

Copyright © Helion 2014

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wstęp	5
Rozdział 1. Podstawy programowania	9
1.1. Podstawowe pojęcia	9
1.2. Algorytmy	14
1.3. Narzędzia programistyczne	25
1.4. Etapy tworzenia programu	26
1.5. Dokumentacja programu	28
1.6. Istota programowania obiektowego	28
Rozdział 2. Aplikacje internetowe	37
2.1. Wprowadzenie	37
2.2. Narzędzia aplikacji internetowych	38
2.3. Struktura aplikacji internetowej	39
2.4. Projektowanie aplikacji internetowych	42
2.5. Framework — platforma programistyczna	44
Rozdział 3. Skrypty po stronie klienta — JavaScript	47
3.1. Wprowadzenie	47
3.2. Struktura języka JavaScript	48
3.3. Składnia języka JavaScript	50
3.4. Instrukcje sterujące	56
3.5. Funkcje	63
3.6. Obiekty	70
3.7. Obiekty wbudowane JavaScript	75
3.8. Obiekty DOM	83
3.9. Obsługa zdarzeń	95
3.10. Wykorzystanie skryptów na stronie internetowej	99
3.11. Walidacja formularzy	106
3.12. Testowanie skryptów	115
Rozdział 4. Biblioteka jQuery	119
4.1. Opis biblioteki	119
4.2. Zdarzenia biblioteki jQuery	128

4.3. Zastosowanie biblioteki jQuery na stronie internetowej	131
4.4. Walidacja formularzy	147
Rozdział 5. Ajax	153
5.1. Wprowadzenie	153
5.2. Obiekt XMLHttpRequest	154
5.3. Żądania Ajax	157
Rozdział 6. Serwery WWW	163
6.1. Serwer Apache	163
6.2. Serwer IIS	167
Rozdział 7. Język PHP	169
7.1. Wprowadzenie	169
7.2. Instalowanie PHP	171
7.3. Pakiet XAMPP	173
7.4. Struktura języka PHP	180
7.5. Składnia języka PHP	183
7.6. Instrukcje sterujące	198
7.7. Funkcje	207
7.8. Funkcje wbudowane	215
7.9. Pliki i katalogi	229
7.10. Obiekty	243
7.11. Formularze	259
7.12. Pliki cookies i sesje	271
7.13. Sieć	287
7.14. Bazy danych w PHP	290
Rozdział 8. Technologia .NET	311
8.1. Założenia platformy .NET	311
Rozdział 9. Systemy zarządzania treścią CMS	315
9.1. Joomla!	316
Rozdział 10. Testowanie i dokumentowanie aplikacji	337
10.1. Testowanie aplikacji	337
10.2. Zagrożenia	338
10.3. Dokumentowanie aplikacji	340
Skorowidz	342

Zadanie 3.1

Zmodyfikuj kod podany w przykładzie 3.31. Zamiast tworzenia zmiennej `color_t` zdefiniuj funkcję `zmiana_koloru` i wywołaj ją w kodzie, tak aby uzyskać efekt podobny do pokazanego w przykładzie.

3.6. Obiekty

Język JavaScript jako język zorientowany obiektowo udostępnia wiele wbudowanych obiektów. Daje również możliwość odczytywania ich właściwości oraz korzystania z ich metod. Właściwości obiektu reprezentują jego cechy (np. liczba znaków łańcucha, wymiary okna) lub pozwalają określić jego stan. Nazywane są również polami obiektu, zmiennymi lub zmiennymi składowymi. Metody to funkcje, które wykonują różne operacje na obiekcie.

Do właściwości i metod można się odwołać podobnie jak do zwykłych zmiennych i funkcji, trzeba tylko przed ich nazwą umieścić nazwę obiektu, którego są elementami (np. zmienną, która przechowuje dany obiekt), i kropkę.

Przykład 3.32

```
var napisz="Witaj w szkole";
document.write(napisz.toUpperCase());
document.write(napisz.length);
```

W podanym przykładzie zmienna `napisz` przechowuje obiekt. Funkcja `toUpperCase()` jest metodą obiektu, a `napisz.length` jego właściwością. Wynikiem będzie wypisanie tekstu: "WITAJ W SZKOLE15".

Każdy element strony internetowej jest traktowany jako obiekt umieszczony w obiekcie `document`. Obiekty języka JavaScript zawierają informacje opisujące stronę i jej środowisko.

3.6.1. Obiekty przeglądarki

Dla każdej strony internetowej zdefiniowane są następujące obiekty:

- `window`
- `navigator`
- `document`
- `location`
- `history`

window

Opisuje bieżące okno przeglądarki. Jest najważniejszym obiektem w hierarchii, dlatego odwołanie do jego właściwości lub metod nie wymaga podania nazwy obiektu. Tworzony jest automatycznie podczas otwierania okna przeglądarki. Posiada wiele właściwości przydatnych podczas tworzenia strony.

navigator

Pozwala na dostęp do informacji dotyczących przeglądarki. Służy do ustalenia wersji przeglądarki używanej przez użytkownika. Właściwości tego obiektu mogą być tylko odczytywane. Najczęściej używa się go do sprawdzenia, czy przeglądarka jest odpowiednia do zastosowanej wersji języka JavaScript.

Przykład 3.33

```
if ((navigator.appName=="Firefox") &&
    (parseInt(navigator.appVersion)>=3) )
    { kod skryptu }
else
    document.write("Masz niewłaściwą wersję przeglądarki!!!")
```

Właściwość `appName` zawiera nazwę przeglądarki, `appVersion` zawiera numer wersji przeglądarki, funkcja `parseInt()` zamienia dowolny łańcuch na liczbę.

document

Zawiera informacje o bieżącym dokumencie HTML. Poprzez jego właściwości mamy wpływ na elementy strony (np. kolor tła, kolor czcionki). Jego metody umożliwiają wyświetlenie np. tekstu w oknie przeglądarki.

location

Zawiera informacje o bieżącym adresie URL. Jego właściwości odpowiadają kolejnym członom adresu. Ogólna postać lokalizatora URL to:

```
protocol://host:port/path#fragment?query
```

Właściwość `protocol` to łańcuch znakowy określający protokół, zgodnie z którym należy nawiązać łączność z podanym serwerem (np. *http:*, *ftp:*, *file:*), `host` to łańcuch znakowy określający nazwę serwera z bieżącego adresu, `port` to łańcuch znakowy odpowiadający portowi, przez który należy połączyć się z serwerem, `pathname` to łańcuch znakowy określający ścieżkę dostępu do dokumentu na serwerze, `hash` to łańcuch znakowy odpowiadający nazwie zakotwiczenia (przypisanie tu jakiejś wartości spowoduje przewinięcie dokumentu do wskazanego punktu), `search` to człon lokalizatora URL.

Aby zmienić adres strony wyświetlanej w oknie, wystarczy zmienić lokalizator URL.

Przykład 3.34

```
window.location = "https://www.google.pl/search?q=warszawa+lotnisko"
```

history

Zawiera historię stron odwiedzanych w bieżącej sesji. Posiada dwie właściwości:

- `current` — zawiera bieżący lokalizator URL,
- `length` — zawiera długość listy historii.

Dodatkowo strona może zawierać następujące elementy DOM:

- `form` — formularz,
- `anchor` — zakotwiczenie,
- `link` — odsyłacz,
- `image` — obrazek,
- `embed` — dodatek,
- `applet` — aplet Javy,
- `frame` — ramka,
- `area` — mapa graficzna.

Predefiniowane obiekty w JavaScript

- `String` — łańcuch tekstowy. Posiada własność `length` i metody: `slice()`, `split()`.
- `Array` — tablica. Posiada własność `length` i metody: `concat()`, `pop()`, `push()`.
- `Date` — data. Posiada metody: `getMonth()`, `getDay()`.
- `Match` — obiekt matematyczny.

3.6.2. Tworzenie obiektów

Aby utworzyć nowy obiekt w języku JavaScript, należy skorzystać z konstrukcji, która zdefiniuje nazwę obiektu oraz pozwoli utworzyć jego właściwości i metody.

Przykład 3.35

```
var obiekt_1 = {
  nazwisko: 'Nowacki',
  imie: 'Marek',
  zawod: 'informatyk',
  pokaz: function ()
  {
    document.write(this.nazwisko + ' ' + this.imie)
  }
}
```

Utworzony obiekt posiada trzy właściwości (`nazwisko`, `imie`, `zawod`) i jedną metodę, `pokaz`, która wyświetla nazwisko i imię. Metoda jest funkcją zdefiniowaną w obrębie obiektu. Przy definiowaniu obiektu deklarowane właściwości i funkcje muszą być oddzielone przecinkami. Słowo kluczowe `this` pozwala odwołać się do właściwości lub metod danego obiektu z jego wnętrza. W tym wypadku metoda `pokaz` odwołuje się do właściwości obiektu `obiekt_1`.

Dla istniejących obiektów można deklarować nowe właściwości i metody.

Przykład 3.36

```
var obiekt_1 = {
  nazwisko: 'Nowacki',
  imie: 'Marek',
  zawod: 'informatyk',
  pokaz: function ()
  {
    document.write(this.nazwisko + ' ' + this.imie)
  }
}

obiekt_1.wiek = 19;
obiekt_1 wypisz_wiek = function() {alert('Wiek: ' + this.wiek + 'lat')}
obiekt_1.wypisz();
```

Zadanie 3.2

Zdefiniuj w języku JavaScript obiekt opisujący wybraną markę samochodów. Zdefiniuj jego właściwości i metody. Wyświetl w przeglądarce informacje o utworzonym obiekcie.

3.6.3. Tworzenie obiektów z użyciem konstruktora

W języku JavaScript istnieje możliwość tworzenia wielu obiektów posiadających podobne właściwości. W tym celu można posłużyć się konstruktorem obiektu. Konstruktor przypomina zwykłą funkcję.

Przykład 3.37

```
function klient(nazwisko_k, imie_k, zawod_k)
{
  this.nazwisko=nazwisko_k;
  this.imie=imie_k;
  this.zawod=zawod_k;
  wypisz = function ()
  {
    alert(this.nazwisko + ' ' + this.imie)
  }
}
```


Został utworzony konstruktor o nazwie `klient` z właściwościami `nazwisko`, `imie`, `zawod` oraz metodą `wypisz()`. Właściwościom obiektu zostały przypisane wartości parametrów. Użyte słowo kluczowe `this` odnosi się do aktualnego obiektu i pozwala na przypisanie wartości parametru do odpowiedniego pola tego obiektu.

Właściwości obiektu istnieją w porządku, w jakim zostały zdefiniowane. Można się do nich odwoływać na dwa sposoby:

```
nazwa_obiektu.nazwa_właściwości
```

np. `klient.nazwisko`,

```
nazwa_obiektu["nazwa_właściwości"]
```

np. `klient["nazwisko"]`.

Słowo kluczowe `new`

Do utworzenia nowego obiektu na podstawie konstruktora stosowane jest słowo kluczowe `new`.

Przykład 3.38

```
var osoba1 = new klient('Kowalski', 'Jan', 'kierowca');
var osoba2 = new klient('Nowak', 'Anna', 'sekretarka');
```

Powstały dwa nowe obiekty `osoba1` i `osoba2` należące do klasy `klient`.

Zadanie 3.3

Sprawdź, jaki rezultat otrzymasz po wykonaniu poniższego kodu:

```
osoba1.wypisz();
osoba2.imie = 'Ewa';
osoba2.wypisz();
```

Właściwość `prototype`

Innym sposobem deklarowania metod i właściwości dla obiektu jest wykorzystanie właściwości `prototype`.

Przykład 3.39

```
function klient()
{
  this.nazwisko='Bielski';
  this.imie='Paweł';
}
klient.prototype.pisz_dane = function ()
```

```

    {
        document.write(this.nazwisko + ' ' + this.imie)
    }

    klient.prototype.zawod = 'kierowca';

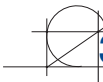
    var osoba1 = new klient();

    osoba1.pisz_dane();

```

W definicji konstruktora nie zostały zadeklarowane żadne metody i właściwości. Dopiero po użyciu właściwości `prototype` została dodana metoda `pisz_dane` oraz właściwość `zawod`. Od tej pory każdy nowo tworzony obiekt na podstawie konstruktora `klient` będzie posiadał tę dodatkową właściwość i metodę.

Właściwość `prototype` może być również wykorzystana do dodawania dodatkowych metod lub właściwości do istniejących obiektów.



3.7. Obiekty wbudowane JavaScript

3.7.1. Obiekt String

Obiektem zawsze występującym w języku JavaScript jest obiekt `String`. Posiada on jedną właściwość, `length`, określającą długość łańcucha.

Przykład 3.40

```

    tekst = "Obiekty języka JavaScript"
    dl = tekst.length

```

Zmienniej `dl` przypisana zostanie wartość 25, określająca długość tekstu.

Obiekt `String` posiada dwa typy metod.

Pierwszy typ metod odnosi się do utworzonego łańcucha, np. metoda `substring()`, która zwraca podzbiór tego łańcucha. Jej parametry określają położenie początku i końca podzbioru.

Przykład 3.41

```

    x = tekst.substring (15,19)

```

Zostanie zwrócony łańcuch `Java`.

Metodami, które można wykorzystać do zmiany wielkości liter, są: `toUpperCase()` i `toLowerCase()`. Metoda `toUpperCase()` zamienia wszystkie litery ciągu na duże, a metoda `toLowerCase()` zamienia wszystkie litery ciągu na małe.

Przykład 3.42

```
x = tekst.toLowerCase()
```

Zostanie zwrócony łańcuch obiekty języka javascript.

Przykład 3.43

```
x = tekst.toUpperCase()
```

Zostanie zwrócony łańcuch OBIEKTY JEZYKA JAVASCRIPT.

Do istniejących obiektów można dodawać nowe właściwości i można definiować dla nich nowe metody.

Do obiektu `String` dodamy dodatkową funkcjonalność, dzięki której pierwsza litera łańcucha będzie pisana dużą literą.

Przykład 3.44

```
String.prototype.duzaLitera = function() { return this.charAt(0).  
toUpperCase() + this.substr(1);}
```

Metoda `charAt()` zwraca znak z pierwszej pozycji łańcucha znaków. Metoda `substr()` zwraca podzbiór łańcucha znaków. Jako parametr tej metody został podany indeks pierwszego znaku podzbioru. Zadeklarowana metoda została dołączona do obiektu `String` i od tej pory każdy nowy tekst będzie posiadał metodę, która zamieni jego pierwszą literę na dużą.

Przykład 3.45

```
var tx1 = 'komputer';  
document.write(tx1.duzaLitera())
```

W wyniku wykonania skryptu wyświetli się tekst 'Komputer'.

3.7.2. Obiekt Date

Kolejnym obiektem języka JavaScript jest obiekt specjalny `Date`, który służy do przechowywania wartości daty i czasu. Przy jego pomocy można odczytać wartość daty i czasu, można też rozłożyć te wartości na części, odczytując oddzielnie dzień, miesiąc, rok itp. Można również te części niezależnie modyfikować.

Aby odczytać bieżącą datę i czas, należy utworzyć obiekt `Date` bez parametrów.

Przykład 3.46

```
var dat_cz = new Date();
```

Można utworzyć obiekt z określoną liczbą parametrów. Tych parametrów może być od dwóch do siedmiu (rok, miesiąc, dzień, godzina, minuty, sekundy, milisekundy). Taki obiekt będzie zawierał ściśle określoną wartość daty i godziny.

Przykład 3.47

```
var data = new Date(2013,2,27);
```

W języku JavaScript wartości daty i czasu są przechowywane w formacie `timestamp`, czyli jako liczba milisekund, które upłynęły od północy 1 stycznia 1970 roku.

Do konwersji obiektu `Date` na tekst służy kilka funkcji:

- `toString()` — zwraca datę, czas oraz informacje o strefie czasowej w języku angielskim,
- `toLocaleString()` — zwraca datę i czas dla bieżących ustawień regionalnych,
- `toUTCString()` — zwraca datę, czas oraz informacje o strefie czasowej dla formatu UTC (*Universal Coordinated Time*),
- `toGMTString()` — działa jak funkcja `toUTCString()`,
- `toDateString()` — zwraca tylko datę w języku angielskim,
- `toLocaleDateString()` — zwraca tylko datę dla bieżących ustawień regionalnych,
- `toTimeString()` — zwraca tylko czas w języku angielskim,
- `toLocaleTimeString()` — zwraca tylko czas dla bieżących ustawień regionalnych.

UWAGA

Dla funkcji `toString()` i `toLocaleString()` różne przeglądarki zwracają wyniki w różny sposób.

Jednym z najczęściej spotykanych przykładów wykorzystania JavaScript na stronie WWW jest wyświetlanie daty i czasu jako elementu strony internetowej.

Przykład 3.48

```
<html>
<head>
<title>JavaScript - Data i czas</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<body>
<h2>Wyświetlam bieżącą datę i czas</h2>
<p>
<script type="text/javascript">
data_n = new Date();
data_l = data_n.toString();
data_u = data_n.toGMTString();
```

```

data_r = data_n.toLocaleString();
document.write("<b>Czas lokalny:</b> " + data_l + "<br>");
document.write("<b>Czas uniwersalny:</b> " + data_u + "<br>");
document.write("<b>Czas regionalny:</b> " + data_r + "<br>");

</script>
</p>
</body>
</html>

```

Utworzonej zmiennej o nazwie `data_n` przypisany został obiekt `Date`. Zmienne `data_l`, `data_u` i `data_r` zawierają tekstową postać czasu lokalnego, uniwersalnego i regionalnego, odpowiadającego wartości przechowywanej w zmiennej `data_n`. Do wyświetlenia wartości otrzymanych zmiennych została użyta funkcja `document.write`. W rezultacie na stronie wyświetlają się informacje o czasie lokalnym, uniwersalnym i regionalnym (rysunek 3.3).

Wyświetlam bieżącą datę i czas

Czas lokalny: Mon Jul 29 21:11:00 UTC+0200 2013
Czas uniwersalny: Mon, 29 Jul 2013 19:11:00 UTC
Czas regionalny: 29 lipca 2013 21:11:00

Rysunek 3.3. Wyświetlenie na stronie informacji o dacie i czasie

Zadanie 3.4

Przetestuj działanie skryptu z przykładu 3.49 w różnych przeglądarkach.

Zadanie 3.5

Zdefiniuj funkcję, która będzie wyświetlała datę i czas po polsku, na przykład tak jak na rysunku 3.4.

Dzisiejszy dzień to:
Środa, 12 grudnia 2012 roku,
Godzina:21:39:35

Rysunek 3.4. Data i czas wyświetlane na stronie internetowej

Zadanie 3.6

Utwórz stronę internetową, na której data i czas będą wyświetlane w postaci kartki z kalendarza.

3.7.3. Obiekt Array

Tabele służą do przechowywania wielu zmiennych. W języku JavaScript do pracy z tabelami można używać wbudowanego obiektu `Array`. Posiada on metody do manipulowania tablicami zmiennych.

Aby utworzyć nową tablicę, należy zadeklarować obiekt `Array` w postaci:

```
var NazwaTablicy = new Array()
```

lub

```
var NazwaTablicy = []
```

Jeżeli w nawiasach zostanie podana liczba `n`, to zostanie utworzona tablica zawierająca `n` pustych elementów.

Przykład 3.49

```
var Tab1 = new Array(10)
```

```
var Tab2 = [15]
```

Tablicę można również tworzyć, wstawiając do niej konkretne wartości.

Przykład 3.50

```
var Tab3 = new Array('Anna', 'Adam', 'Piotr', 'Ewa')
```

```
var Tab4 = ['Paweł', 'Marcin', 'Ela']
```

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

Przykład 3.51

```
document.write(Tab3[2])
```

W wyniku wyświetli się wartość `'Piotr'`.

Aby dodać nową wartość do tablicy, należy przypisać tę wartość do odpowiedniego indeksu tablicy.

Przykład 3.52

```
var Tab5 = new Array('kot', 'pies', 'koń')
```

```
Tab5[3] = 'mysz';
```

```
Tab5[4] = 'chomik';
```

```
document.write(Tab5[0] + ' i ' + Tab5[3])
```

W wyniku wyświetli się tekst `'kot i mysz'`.

Dzięki właściwości `length` można określić, z ilu elementów składa się tablica. Jest to bardzo przydatna właściwość, szczególnie gdy chcemy utworzyć pętlę odczytującą wszystkie elementy tablicy.

Przykład 3.53

```
var Tablica_N = new Array('Anna', 'Adam', 'Piotr', 'Ewa', 'Paweł', 'Marcin',
    'Ela')
for (i = 0; i < tablica_N.length; i++)
{
    document.write(Tablica_N[i] + "<br>");
}
```

W wyniku zostaną wyświetlone po kolei wszystkie elementy tablicy.

Zadanie 3.7

Zmień zapis skryptu podanego w przykładzie 3.54, tak aby wyświetlone zostały elementy tabeli od ostatniego do pierwszego. Pamiętaj, że indeksowanie elementów rozpoczyna się od zera.

Tablice wielowymiarowe

W języku JavaScript można również tworzyć tablice wielowymiarowe. Wtedy element tablicy jest opisywany za pomocą indeksu określającego jego położenie w wierszu i kolumnie.

Przykład 3.54

```
var Tablica_Z = [];
Tablica_Z[0] = ['Anna', 'Nowak'];
Tablica_Z[1] = ['Adam', 'Kowal'];
Tablica_Z[2] = ['Piotr', 'Ogórek'];
Tablica_Z[3] = ['Ewa', 'Lisowska'];
document.write('imię: ' + Tablica_Z[0][0] + 'nazwisko: ' + Tablica_Z[0]
    [1] + "<br>");
document.write('imię: ' + Tablica_Z[1][0] + 'nazwisko: ' + Tablica_Z[1]
    [1] + "<br>");
document.write('imię: ' + Tablica_Z[2][0] + 'nazwisko: ' + Tablica_Z[2]
    [1] + "<br>");
document.write('imię: ' + Tablica_Z[3][0] + 'nazwisko: ' + Tablica_Z[3][1]);
```

Łączenie elementów tablicy

Za pomocą metody `join()` można łączyć elementy tablicy w jeden tekst. W metodzie tej można opcjonalnie podać parametr, który określi znak oddzielający kolejne elementy tablicy. Jeżeli nie zostanie podana wartość tego parametru, domyślnym znakiem będzie przecinek.

Przykład 3.55

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');
document.write(Tablica.join() + "<br>");
document.write(Tablica.join(" - ") + "<br>");
```

Odwracanie kolejności elementów tablicy

Za pomocą metody `reverse()` można odwrócić kolejność elementów tablicy.

Przykład 3.56

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');
document.write(Tablica.join() + "<br>");
Tablica.reverse();
document.write(Tablica.join() + "<br>");
```

Sortowanie

Do sortowania elementów tablicy służy metoda `sort()`.

Przykład 3.57

```
var Tablica = new Array('Paweł', 'Anna', 'Maria', 'Adam', 'Piotr');
Tablica.sort();
document.write(Tablica.join());
```

Domyślnie tablica jest sortowana leksykograficznie. Powoduje to, że liczba 12459 będzie mniejsza od 4567, ponieważ cyfra na pierwszej pozycji jest mniejsza. Aby temu zaradzić, można sortować tablicę według własnych kryteriów. Należy skorzystać z dodatkowego parametru metody `sort()`. Parametrem będzie własna funkcja sortująca. Tworząc taką funkcję, należy pamiętać o trzech zasadach, które muszą być spełnione:

- jeżeli funkcja (a, b) zwróci wartość mniejszą od 0, to wartości a zostanie nadany indeks mniejszy od indeksu przyznanego wartości b ,
- jeżeli funkcja (a, b) zwróci wartość równą 0, to wartości indeksów pozostaną bez zmian,
- jeżeli funkcja (a, b) zwróci wartość większą od 0, to wartości a zostanie nadany indeks większy od indeksu przyznanego wartości b .

Stosując się do tych zasad, można tworzyć własne metody sortowania w tablicy.

Przykład 3.58

```
function porownaj(a,b)
{
return a - b
}

var Tablica = new Array(27, 100, 10, 450, 1654, 320);
document.write('Bez sortowania: ' + Tablica.join() );

document.write('Sortowanie domyślne: ');
Tablica.sort()
document.write(Tablica.join());

document.write('Sortowanie poprawne: ');
Tablica.sort(porownaj)
document.write(Tablica.join() );
```

W podanym przykładzie została zdefiniowana funkcja `porownaj(a,b)`, która zwróci wartość mniejszą od zera, równą zero lub większą od zera. W zależności od zwróconej wartości elementy tablicy zostaną uporządkowane od wartości najmniejszej do największej.

Zadanie 3.8

Utwórz tablicę, która będzie zawierała elementy zawierające cyfry i litery. Zdefiniuj funkcję, która pozwoli uporządkować elementy tabeli w ten sposób, że najpierw będą umieszczone litery w kolejności alfabetycznej, a następnie cyfry w kolejności od wartości najmniejszej do największej.

Zadanie 3.9

Utwórz tabelę „trzy na trzy”.

```
var tablica = new Array(1, 2, 3);
document.write(tablica[1] + "<br>");
tablica[1] = 123;
document.write(tablica[1] + "<br>");
```

Umieść w niej obrazki. Zdefiniuj kod, który spowoduje zmianę wyświetlanych na stronie obrazków co 2 sekundy.

3.8. Obiekty DOM

DOM (ang. *Document Object Model*) to sposób reprezentacji złożonych dokumentów XML i HTML w postaci modelu obiektowego.

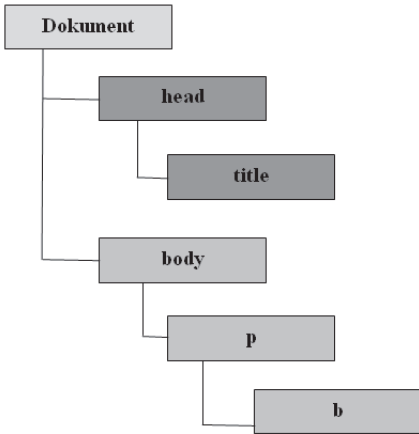
Gdy kod strony jest wczytywany do przeglądarki, przeglądarka zamienia ciąg znaków na stronę internetową. Informacje na temat interpretacji kodu HTML przeglądarka przechowuje w elementach będących obiektami (są to np. informacje o tym, które elementy przedstawić w postaci nagłówków, paragrafów itp.). Obiekty te tworzą obiektowy model dokumentu.

DOM opisuje hierarchię obiektów na stronie oraz udostępnia metody i właściwości, które umożliwiają manipulowanie nimi. W tej hierarchii na samej górze znajduje się okno przeglądarki, czyli obiekt `window`. Zawiera on wszystkie inne obiekty, funkcje i właściwości strony. W oknie znajduje się obiekt `document`, czyli otwarta strona internetowa. W obiekcie `document` znajdują się obiekty strony. W skryptach definiujemy różne działania związane z istniejącymi obiektami, czyli manipulujemy przez skrypty obiektami strony internetowej. Dzięki skryptom można wczytać nową stronę do przeglądarki, zmienić elementy dokumentu, otwierać okna lub modyfikować tekst na stronie. Dzięki DOM język JavaScript staje się narzędziem tworzenia dynamicznych stron internetowych.

Przykład 3.59

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tytuł strony</title>
</head>
<body>
<p>Moja strona <b>internetowa</b></p>
</body>
</html>
```

Podany w przykładzie dokument można rozrysować w postaci drzewa (rysunek 3.5). Na samej górze jest dokument HTML, niżej znajdują się węzły (`nodes`).



Rysunek 3.5. Hierarchia obiektów przykładowej strony internetowej

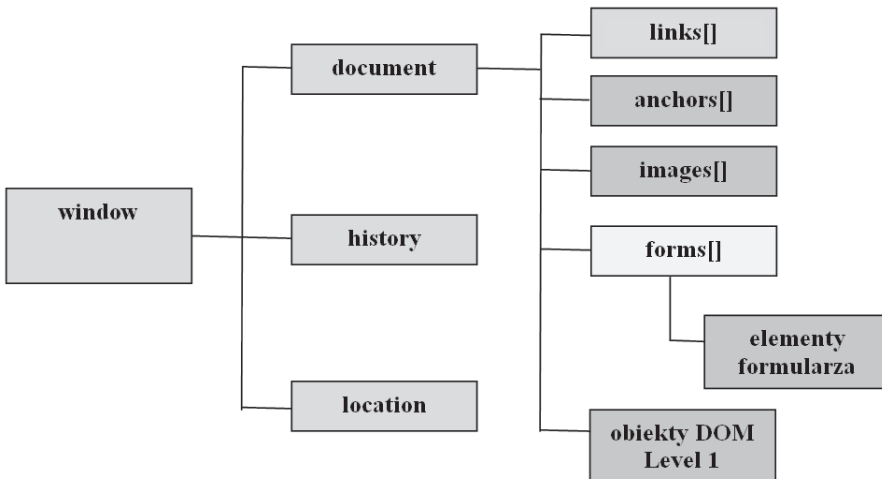
3.8.1. Hierarchia obiektów DOM

Odwołując się do obiektu, należy używać nazw obiektów nadrzędnych oddzielonych kropkami, po których następuje nazwa wybranego obiektu.

Przykład 3.60

```
window.document.link1
```

Wycinek hierarchii DOM z najważniejszymi obiektami strony internetowej został pokazany na rysunku 3.6.



Rysunek 3.6. Hierarchia obiektów DOM

3.8.2. Obiekty przeglądarki

Obiekt window

Obiektem nadrzędnym dla wszystkich obiektów jest obiekt `window`, który zawiera okno przeglądarki. W danej chwili może istnieć wiele obiektów `window`. Każdy z nich reprezentuje otwarte okno przeglądarki. Odwołanie do jego właściwości lub metod nie wymaga podania nazwy obiektu. Tworzony jest automatycznie podczas otwierania okna przeglądarki. Do otwierania nowego okna używamy metody `open()`. Jako parametry metody występują adres URL otwieranej strony oraz nazwa wewnętrzna okna (nie mylić z nazwą wyświetlaną przez przeglądarkę zdefiniowaną metatagiem `<title></title>`):

```
window.open('http://helion.pl', 'Wydawnictwo');
```

Do określania rozmiaru okna mogą być używane właściwości:

- `window.innerHeight` — wysokość okna przeglądarki,
- `window.innerWidth` — szerokość okna przeglądarki.

Do zamknięcia okna używana jest metoda `close()`.

Obiekt document

Obiekt `document` reprezentuje stronę internetową (dokument HTML). Jest on potomkiem obiektu `window`. Za pomocą polecenia `window.document` można odwołać się do bieżącego dokumentu. Można to zrobić również za pomocą polecenia `document`. Odwołanie nastąpi do bieżącego dokumentu w bieżącym oknie. Jeżeli zostało otwarte kilka okien, to aby określić, do którego dokumentu powinno nastąpić odwołanie, należy podać nazwę okna i nazwę dokumentu.

Informacje o bieżącym dokumencie otrzymamy, odwołując się do właściwości i metod obiektu `document`.

- `document.URL` — zwraca adres URL dokumentu jako ciąg tekstu,
- `document.title` — zwraca tytuł strony zdefiniowany w znaczniku `<title>`,
- `document.lastModified` — zwraca datę ostatniej modyfikacji strony,
- `document.backgroundColor` — określa kolor tła dokumentu ustawianego atrybutem `bgcolor` znacznika `<body>`,
- `document.fgColor` — określa kolor pierwszego planu dokumentu ustawianego atrybutem `text` znacznika `<body>`,
- `document.linkColor` — określa kolor łącza w dokumencie ustawianego atrybutem `link`,
- `document.alinkColor` — określa kolor łącza w dokumencie ustawianego atrybutem `alink`,
- `document.vlinkColor` — określa kolor łącza w dokumencie ustawianego atrybutem `vlink`,
- `document.cookie` — ustawia lub odczytuje `cookie` dla dokumentu.

Do odwołania się do elementu strony służą metody `getElementById()` oraz `getElementsByTagName()`. Metoda `getElementById()` używana jest, gdy element, do którego się odwołujemy, posiada atrybut `id`, natomiast metodę `getElementsByTagName()` wykorzystujemy do pobrania kolekcji zawierającej elementy danego typu.

Przykład 3.61

```
<p id="tekst1">Skrypty języka JavaScript</p>
<p>Dokument HTML</p>
<h2>Model dokumentu DOM</h2>
```

Odwołanie w skrypcie do tak zdefiniowanych elementów może mieć postać:

- `document.getElementById("tekst1")` — odwołanie do akapitu z atrybutem `id="tekst1"`,
- `document.getElementsByTagName(p)` — odwołanie do kolekcji akapitów,
- `document.getElementsByTagName(p)[0]` — odwołanie do pierwszego akapitu w kolekcji.

Używając metody `getElementById()`, należy pamiętać, że jest to metoda obiektu `document`, dlatego dostęp do niej jest możliwy tylko za pomocą tego obiektu. Odwołanie do elementu strony będzie możliwe tylko wtedy, gdy temu elementowi zostanie nadany atrybut `id`.

Przykład 3.62

```
<input type="button" id="klik" value="Kliknij!"/>
<script type="text/javascript">

var b = document.getElementById("klik");
alert(b.value);

</script>
```

Możliwość odwołania się do elementu strony jest wykorzystywana do zmiany jej zawartości. Zawartość elementu strony można odczytać i zmienić, używając właściwości `innerHTML`. Właściwość tę posiada każdy element strony internetowej. Określa ona wartość przypisaną elementowi. Właściwość `innerHTML` może być użyta tylko razem z metodą `getElementById()` i tylko dla elementów, dla których został zdefiniowany identyfikator (`id`).

Przykład 3.63

```
<html>
<head>
<title>Tekst</title>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h2>Zmień tekst</h2>
<script type='text/javascript'>

function zmien_Tekst ()
{
    document.getElementById('blok').innerHTML = 'Nie ma czasu!';
}
</script>

<p>Czas to pieniądz. <b id='blok'>Nie ma pieniędzy!</b></p>
<input type="button" onclick="zmien_Tekst()" value = 'Zmień tekst' />

</body>
</html>

```

Po kliknięciu przycisku efektem wykonania kodu będzie zmiana wyświetlanego tekstu (rysunek 3.7).

Zmień tekst

Czas to pieniądz. **Nie ma pieniędzy!**

Zmień tekst

Rysunek 3.7. Możliwość zmiany tekstu wyświetlanego na stronie

Przykład 3.64

W przykładzie 3.49 zostało zdefiniowane wyświetlanie daty i czasu. Zmiana czasu następowała po odświeżeniu strony. Zmodyfikujemy powstały kod tak, aby czas wyświetlany na stronie był zawsze aktualny.

```

<html>
<head>
<title> Mój zegar</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h2>Uciekający czas</h2>

```

```
<div id='zegar' style='width: 100px; border: solid 2px #1d330a;'></div>

<script type='text/javascript'>

var timerID = null;
var timerRunning = false;

function stopclock()
{
    if(timerRunning)
        clearTimeout(timerID)
        timerRunning = false;
}

function startclock()
{
    stopclock();
    showtime();
}

function showtime()
{
    var data_n = new Date();
    var godz = data_n.getHours();
    var min = data_n.getMinutes();
    var sek = data_n.getSeconds();

    var czas = "" + ( godz);
    czas += ((min < 10) ? ":0" : ":") + min;
    czas += ((sek < 10) ? ":0" : ":") + sek;

    document.getElementById('zegar').innerHTML = czas;
    timerID = setTimeout("showtime()",1000);
    timerRunning = true;
}
```

```

</script>

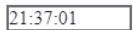
<script>startclock();</script>

</script>
</body>
</html>

```

Metoda `getElementById()` odwołuje się do wcześniej zdefiniowanego identyfikatora 'zegar'. W przykładzie za pomocą właściwości `innerHTML` identyfikatorowi 'zegar' przypisana została wartość zmiennej 'czas', która zawiera bieżący czas. Użyty operator warunkowy `((min < 10) ? ":0" : "") + min`; ma trzy argumenty: *test* po którym występuje znak `?` oraz *wyrażenie1* i *wyrażenie2* oddzielone znakiem `:`. Jeżeli wartość testu jest *true* to wynikiem jest *wyrażenie1*, w przeciwnym razie wynikiem jest *wyrażenie2*. Wynik interpretacji kodu został pokazany na rysunku 3.8.

Uciekający czas



Rysunek 3.8. Wyświetlanie na stronie aktualnego czasu

Metodą obiektu `document` jest również metoda `document.write()`, która wyświetla na stronie internetowej w oknie dokumentu podany tekst.

Zadanie 3.10

Wykorzystując arkusze CSS oraz skrypty z poprzednich przykładów, zdefiniuj kod HTML, który na stronie internetowej wyświetli bieżący czas w postaci zegara cyfrowego.

Zadanie 3.11

Utwórz i dodaj do dokumentu HTML skrypt, który wyświetli datę ostatniej modyfikacji tego dokumentu.

Obiekt history

Drugim obiektem potomnym w stosunku do obiektu `window` jest obiekt `history`. Zawiera on informację o odwiedzanych adresach URL. Posiada zdefiniowane metody, które pozwalają na przejście do wcześniej odwiedzanych stron.

- `history.go()` — otwiera określony adres URL z listy historii. W nawiasach należy podać liczbę dodatnią lub ujemną, określającą, o ile do przodu lub do tyłu należy przemieścić się, aby otworzyć określony adres, np. `history.go(3)`.
- `history.back()` — otwiera poprzedni adres URL z listy historii.
- `history.forward()` — otwiera następny adres URL z listy historii, jeżeli taki istnieje.

Obiekt `history` posiada jedną właściwość, `history.length`, która zawiera informację o długości listy historii.

Wykorzystując metody `back()` i `forward()`, można utworzyć skrypty, które wyświetlą na stronie przyciski *Wstecz* oraz *Dalej*, umożliwiające poruszanie się w przeglądarce po odwiedzanych stronach (rysunek 3.9).



Rysunek 3.9. Wyświetlenie za pomocą metody `back()` i `forward()` przycisków *Wstecz* i *Dalej*

Przykład 3.65

```
<html>
<head>
<title>Przyciski</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1>Firma komputerowa BIT</h1><br>
<input type="button" onclick="history.back()" value="Wstecz">
<input type="button" onclick="history.forward()" value="Dalej">
</body>
</html>
```

Zadanie 3.12

Dla dokumentu HTML utwórz skrypt, dzięki któremu po kliknięciu przycisku *Data modyfikacji* będzie możliwe wyświetlenie daty modyfikacji bieżącej strony internetowej.

Obiekt `location`

Trzecim obiektem potomnym w stosunku do obiektu `window` jest obiekt `location`. Zawiera on informację o bieżącym adresie dokumentu otwartego w oknie. Za pomocą właściwości tego obiektu można uzyskać pełną informację o adresie URL, można też uzyskać dostęp do jego fragmentów.

- `location.href` — zawiera cały adres URL,
- `location.protocol` — zawiera protokół,
- `location.hostname` — zawiera nazwę hosta,
- `location.port` — zawiera numer portu,

- `location.pathname` — zawiera nazwę pliku ze ścieżką,
- `location.search` — zawiera zapytanie, jeżeli znajduje się ono w adresie,
- `location.hash` — zawiera nazwę kotwicy, jeżeli kotwica występuje w adresie.

Przykład 3.66

```
window.location.href = "http://www.helion.pl"
```

UWAGA

Właściwość `location.href` zawiera ten sam adres co właściwość `document.URL`. Jednak właściwość `document.URL` nie można modyfikować. W celu otwarcia nowej strony należy posługiwać się właściwością `location.href`.

Obiekt `location` posiada dwie metody:

- `location.reload()` — odświeża (ponownie wczytuje) bieżący dokument. Jeżeli dodany zostanie parametr `true`, odświeżanie odbędzie się niezależnie od tego, czy dokument uległ zmianie, czy nie.
- `location.replace()` — zastępuje bieżący adres URL nowym.

Zadanie 3.13

Dla dokumentu HTML utwórz skrypt, który wyświetli nazwę pliku oraz ścieżkę dostępu do bieżącej strony internetowej.

Obiekt link

Obiektem potomnym w stosunku do obiektu `document` jest obiekt `link`. Zawiera on informację o łączu do określonego adresu. Obiekty `link` są zapisane w tablicy `links`. W dokumencie może wystąpić wiele obiektów `link`. Każdy z nich jest zapisany jako oddzielny element tablicy.

Właściwość tablicy `document.links.length` określa liczbę linków na stronie.

Każdy obiekt `link` zapisany w tablicy ma listę właściwości określających adres URL. Są to właściwości takie same jak dla obiektu `location`. Można się do nich odwoływać, podając numer w tablicy i nazwę właściwości.

Przykład 3.67

```
link1 = links[0].href;
```

Obiekt anchor

Kolejnym obiektem potomnym w stosunku do obiektu `document` jest obiekt `anchor`. Reprezentuje on kotwicę w bieżącym dokumencie.

UWAGA

Kotwica określa zdefiniowaną lokalizację w dokumencie HTML, do której można się przenieść.

Podobnie jak łącza, kotwice są zapisywane w tablicy o nazwie `anchors`. Każdy jej element jest obiektem `anchor`.

Właściwość tablicy `document.anchors.length` określa liczbę elementów kotwicy na stronie.

Obiekt form

Obiektem potomnym w stosunku do obiektu `document` jest również obiekt `form`. Zawiera on informacje dotyczące formularzy występujących w dokumencie HTML. Obiekty `form` są zapisane w tablicy `forms`. Ponieważ w dokumencie może wystąpić wiele formularzy, każdy z nich jest zapisany jako oddzielny element tablicy. Do wybranego formularza można odwoływać się przez indeks lub przez nazwę, wpisując w kodzie polecenie `document.forms[0]` lub `document.forms['Form1']`. Lepszą metodą odwołania do formularza jest wykorzystanie metody `getElementById()`, np. `document.getElementById('form1')`.

Przykład 3.68

```
<body>
<form id="form1" name="form1" action="mailto:nauka@gmail.com" method="post">
...
</form>
<script type="text/javascript">
document.forms['form1']
...
</script>
</body>
```

Jeżeli został zastosowany atrybut `name` (jak w przykładzie 3.69), to do formularza można odwołać się również w ten sposób: `document.form1`.

Każdy element formularza jest obiektem, więc posiada właściwości. Jedną z nich jest właściwość `value`, która przechowuje bieżącą wartość elementu.

Przykład 3.69

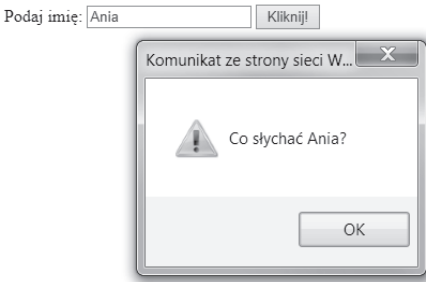
```
<html>
<head>
<title>Co słyhać</title>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<form id="form1" name="form1" action="mailto:nauka@gmail.com" method="post">
Podaj imię: <input type="text" name="imie"/>
<button onclick="Witaj ()"> Kliknij!</button>
</form>
<script type="text/javascript">
function Witaj ()
{
var imie = document.forms['form1'].imie.value;
alert('Co słyhać ' + imie + '?');
}
</script>
</body>
</html>

```

Podany kod definiuje formularz z polem *Imie* i przyciskiem *Kliknij*. Gdy dla przycisku wystąpi zdarzenie `onclick` (kliknięcie przycisku), zostanie uruchomiona funkcja `Witaj ()`, która pobierze wartość elementu `imie` (`var imie = document.forms['form1'].imie.value;`) i wyświetli ją w oknie z komunikatem (rysunek 3.10).



Rysunek 3.10. Pobrane z formularza imię zostało wyświetlone w oknie z komunikatem

Elementy formularza tworzą tablicę. Dostęp do nich jest możliwy przez odwołanie się do kolejnych elementów tej tablicy (`elements[i]`). Podobnie jak do całego formularza, do jego elementów można odwoływać się przez indeks lub przez nazwę:

```

document.forms['form1'].elements[0]
document.forms['form1'].elements['nazwa']
document.forms['form1'].nazwa

```

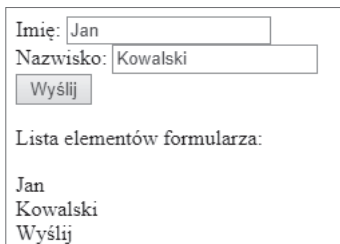
lub za pomocą metody `getElementById()`:

```
document.getElementById('Imie').value
```

Przykład 3.70

```
<html>
<head>
<title>Lista formularza</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<form id="form1" action="mailto:nauka@gmail.com" method="post">
Imię: <input type="text" name="imie" value="Jan"/><br />
Nazwisko: <input type="text" name="nazwisko" value="Kowalski" /><br />
<input type="submit" value="Wyślij" />
</form>
<p>Lista elementów formularza:</p>
<script type="text/javascript">
var x=document.getElementById("form1");
for (var i=0;i<x.length;i++){
document.write(x.elements[i].value);
document.write("<br />");
}
</script>
</body>
</html>
```

Wynikiem wykonania kodu będzie wyświetlenie formularza z polami *Imię* i *Nazwisko* oraz przycisku *Wyślij* wraz z listą elementów formularza (rysunek 3.11).



Rysunek 3.11. Wyświetlenie na stronie formularza wraz z listą jego elementów

A

abstrakcja, 34
ADO.NET, 313
adres IP, 287
Ajax, 38, 153
algorytm, 14
 cechy, 18
 implementacja, 21
 iteracyjny, 20
 klasyfikacja, 19–21
 liniowy, 19
 matematyczny, 21
 porządkujący, 21
 probabilistyczny, 21
 przeszukujący, 21
 rekurencyjny, 20–21
 sekwencyjny, 20
 szyfrujący, 21
 warunkowy, 20
analiza
 leksykalna, 12
 semantyczna, 12
 składniowa, 12
 syntaktyczna, 12
 znaczeniowa, 12
animowanie
 grafiki, 102–106
 tekstu, 100–102
Apache, 163–164
 instalowanie serwera
 w systemach Linux,
 164–165
 instalowanie serwera
 w systemie Windows,
 165–166
aplet, 10
aplety Java, 38
aplikacja, 10
aplikacja internetowa,
 10, 37

aplikacja webowa, *Patrz*
aplikacja internetowa
architektura
 4-warstwowa, 40–41
 Model-View-Controller
 (MVC), 41
argumenty funkcji
 w JavaScript, 64–65
 w PHP, 212–215
argumenty
 przekazywane przez
 wartość, 212–213
 przekazywane za pomocą
 referencji, 213
ASP.NET, 39, 44, 167
ataki w aplikacjach
 internetowych, 340–342
audyt bezpieczeństwa, 340

B

baner animowany, 104–106
baner (przesuwany tekst),
 100–101
baner (tekst pływający),
 101–102
bazy danych w PHP,
 290–306
biblioteka jQuery,
 119–121
biblioteki
 języka JavaScript, 119
 standardowe, 13–14
błędy uruchomienia, 27

C

C#, 313
CGI, 39
CIL, 311
CLI, 311

CLR, 311
CMS, 315
cookies, 271, 273–276
CSRF, 339

D

debuger, 25–26
definiowanie funkcji
 w JavaScript, 63–65
 w PHP, 207–208
definiowanie klasy w PHP,
 243–244
destruktor, 252, 257
długość drogi, 22
dokumentacja
 programu, 28
 techniczna, 28,
 340–341
 użytkownika, 28,
 340–341
dokumentowanie aplikacji,
 340–341
dołączanie plików w PHP,
 229–231
DOM, 47, 83–84
domyślne argumenty funkcji
 w PHP, 214–215
drzewa, 35
drzewo algorytmu,
 15–16, 22
DSN, 306
dziedziczenie, 35
 konstruktorów
 i destruktorów, 257
 w PHP, 252–254
dziel i zwyciężaj, 19

E

edytory, 25

F

- filtry biblioteki jQuery, 124–128
- Flash, 38
- formaty przesyłanych danych w technologii Ajax, 161
- formularz HTML, 260–261
- formularze
 - służące do wysyłania plików, 268–270
 - w PHP, 259–271
- framework, 44
- funkcja, 63, 207
 - \$(), 121
 - alert(), 68
 - array_merge(tablica1, tablica2, ...), 285
 - array_search(), 217
 - array_unique(tablica), 285
 - arsort(), 216
 - asort(), 216
 - closedir(), 238
 - count(), 215
 - date(), 219–220
 - define(), 197
 - die(), 242
 - each(), 216
 - empty(), 263–268
 - error, 159
 - eval(), 68
 - exit(), 242
 - fclose(), 233
 - feof(), 234
 - fgetc(), 235–236
 - fgets(), 234
 - file(), 237
 - file_exists(), 231
 - file_get_contents(), 237
 - filesize(), 232
 - fopen(), 232
 - foreach(), 216–217
 - fread(), 236–237
 - ftp_close(), 289
 - ftp_connect(), 288
 - ftp_fget(), 288–289
 - ftp_login(), 288
 - fwrite(), 233–234
 - getdate(), 218–219
 - include(), 229–231
 - is_file(), 231
 - is_uploaded_file(), 270
 - isFinite(), 67–68
 - isNaN(), 67
 - isSet(), 262–263, 278
 - krsort(), 216
 - ksort(), 216
 - list(), 216
 - ltrim(), 225
 - mkdir(), 237–238
 - mktime(), 220–221
 - move_uploaded_file(), 270
 - mysql_affected_rows(), 297
 - mysql_close(), 291
 - mysql_connect(), 290–291
 - mysql_fetch_array(), 293–294
 - mysql_fetch_row(), 293
 - mysql_get_server_info(), 291
 - mysql_num_rows(), 293
 - mysql_query(), 293
 - mysql_select_db(), 292
 - nl2br(), 222
 - opendir(), 238
 - parseFloat(), 67
 - parseInt(), 66–67
 - print(), 181
 - readdir(), 238
 - readfile(), 237
 - require(), 229–231
 - rmdir(), 238
 - rsort(), 216
 - rtrim(), 225
 - scandir(), 239
 - serialize(), 278
 - session_destroy(), 277
 - session_id(), 277
 - session_start(), 277
 - setcookie(), 271–272
 - settype(), 192
 - sizeof(), 215
 - sort(), 216
 - strcasecmp(), 229
 - strcmp(), 229
 - strlen(), 225
 - strpos(), 227
 - strstr(), 226
 - strtok(), 228
 - strtolower(), 224
 - strtoupper(), 224
 - substr(), 227
 - success(), 159
 - time(), 217–218
 - toDateString(), 77
 - toGMTString(), 77
 - toLocaleDateString(), 77
 - toLocaleString(), 77
 - toLocaleTimeString(), 77
 - toString(), 77
 - toTimeString(), 77
 - touch(), 232
 - toUTCString(), 77
 - trim(), 225
 - ucfirst(), 224
 - ucwords(), 224
 - unlink(), 232
 - unserialize(łańcuch), 285
 - wordwrap(), 222–224
- funkcje
 - analizowania ciągów znaków, 225–229
 - daty i czasu, 217–221
 - formatowania ciągów, 221–225
 - tablic, 215–217
 - usuwania ciągu znaków, 225
 - wyjścia, 242–243
 - wykonujące operacje na plikach, 231–237
 - zmiany wielkości liter, 224
 - zwrotne żądania Ajax, 158–160

funkcje wbudowane
 w JavaScript, 66–69
 w PHP, 215–229

G

Google hacking, 340

H

hermetyzacja, 34–35
 w PHP, 248–250
 heurystyka, 19
 hierarchia klas, 35
 hierarchia obiektów DOM,
 84
 hover, 129

I

IDE, *Patrz* zintegrowane
 środowisko
 programistyczne
 IIS, 167–168
 implementacja algorytmu, 14
 indeksowanie ciągu znaków,
 225–226
 indeksy tablicy zwracanej
 przez funkcję `getdate()`, 218
 inkapsulacja, *Patrz*
 hermetyzacja
 instalacja XAMPP,
 174–179
 instalowanie PHP,
 171–173
 instrukcja
 break
 w JavaScript, 62
 w PHP, 204–205
 continue
 w JavaScript, 62
 w PHP, 205
 echo, 181
 function, 207
 global, 209–210
 if ... else, 57
 new, 74

prototype, 74–75
 return
 w JavaScript, 63
 w PHP, 208
 SQL CREATE DATABASE,
 305
 SQL CREATE TABLE, 305
 sterująca
 w JavaScript, 56–62
 w PHP, 198–207
 switch
 w JavaScript, 58
 w PHP, 200–201
 throw, 258
 try...catch, 258
 UPDATE, 301
 var, 65–66
 warunkowa
 w JavaScript, 57
 w PHP, 198–200
 interpreter, 10
 ISAPI, 39

J

JavaScript, informacje ogólne
 38, 48–50
 język
 hybrydowy, 30
 pośredni, 311
 programowania, 9–11
 skryptowy, 13, 47
 Joomla!, 315–321
 grafika 328–329
 komponenty 333–334
 moduły 330–333
 szablony 334–335
 tworzenie artykułów
 323–326
 tworzenie kategorii,
 321–323
 tworzenie menu 326–328
 tworzenie treści portalu,
 321
 JSON, 161
 JSP, 39

K

klasa, 31, 33, 73, 243
 klasa
 bazowa, 35
 Exception, 257
 pochodna, 35
 klasy znaków, 111
 klauzula WHERE,
 296–297
 klucz REMOTE_ADDR,
 287
 kolejność wykonywania
 skryptów, 99
 komentarze
 w JavaScript, 50–51
 w PHP, 183–184
 kompilacja, 11–13, 27
 kompilator, 9
 komunikacja między
 serwerem a użytkownikiem
 w technologii Ajax, 154
 konsolidacja, 10, 27
 konsolidator, 10
 konstruktor
 w JavaScript, 73
 w PHP, 250–252, 257
 kontroler, 43–44
 kotwica, 92

L

linker, 10
 lista kroków, 15

M

magiczne stałe, 197–198
 menu nawigacyjne, 132
 metoda, 70
 addClass(), 127
 animate(), 137–144
 append(), 147
 bind(), 128
 CSS, 127
 document.write(), 89
 exec(), 308–309

fadeIn(), 136
 fadeOut(), 136
 getElementById(), 86
 getElementByTagName(), 86
 hide(), 136
 history.back(), 89
 history.forward(), 89
 history.go(), 89
 join(), 81
 klasy w PHP, 244
 location.reload(), 91
 location.replace(), 91
 open(), 157–158
 query(), 307–308
 ready(), 121
 reset (), 114
 reverse(), 81
 send(), 158
 show(), 136
 slideDown(), 136
 slideToggle(), 136
 slideUp(), 136
 sort(), 81
 substring(), 75, 102
 toggle(), 123
 toLowerCase(), 75
 toUpperCase(), 75
 window.setTimeout(), 102
 zachłanna, 19
 metoda GET
 w technologii Ajax, 158
 w PHP, 261–262
 metoda POST
 w technologii Ajax, 158
 w PHP, 263–264
 model, 41–43
 modelowanie obiektowe, 33–34
 moduł, 10
 modyfikatory dostępu
 w PHP, 248
 MSIL, 311

N

naprzemienne bloki kodu
 PHP i HTML, 205–207
 narzędzia
 aplikacji internetowych,
 38–39
 programistyczne, 25,
 44–45

O

obiekt, 31, 243
 anchor, 91–92
 Array, 79–82
 Date, 76–78
 document, 71, 85–89
 form, 92–94
 history, 71–72, 89–90
 klasy PDO, 306
 link, 91
 location, 71, 90–91
 navigator, 71
 RegExp (), 110
 String, 75–76
 window, 70, 85
 XMLHttpRequest,
 153–157
 obiektowość, 21, 34
 obiekty
 w JavaScript, 70, 75–82
 w PHP, 246–248
 obiekty
 predefiniowane
 w JavaScript, 72
 przeglądarki, 70–72
 obsługa protokołu FTP
 w PHP, 288
 obsługa zdarzeń
 w JavaScript, 96–99
 w jQuery, 128–131
 opcje żądania Ajax, 157
 operacje na katalogach,
 237–239
 operatory arytmetyczne
 w JavaScript, 54
 w PHP, 193

operatory bitowe
 w JavaScript, 55
 w PHP, 194
 operatory dekrementacji
 w JavaScript, 54
 w PHP, 196–197
 operatory inkrementacji
 w JavaScript, 54
 w PHP, 196
 operatory i wyrażenia w PHP,
 193–197
 operatory konkatencji,
 196
 operatory logiczne
 w JavaScript, 56
 w PHP, 194–195
 operatory porównania
 w JavaScript, 55
 w PHP, 194
 operatory przypisania
 w JavaScript, 56
 w PHP, 195
 operatory warunkowe, 201
 opis słowny algorytmu,
 14–15
 optymalizacja
 kodu wynikowego, 12
 programu, 28

P

paradygmat programowania,
 28
 Pascal, 29
 PDO, 306–309
 pętla, 59, 202
 pętla do ... while
 w JavaScript, 61
 w PHP, 203
 pętla for
 w JavaScript, 59–60
 w PHP, 202
 pętla foreach, 203–204
 pętla while
 w JavaScript, 60–61
 w PHP, 202–203

PHP, informacje ogólne, 39,
169–170
planowanie programu
komputerowego, 26
platforma
.NET Compact
Framework, 313
Eclipse, 44
NetBeans IDE, 45
Zend, 45
platforma programistyczna,
44
.NET, 311–313
plik php.ini, 180
pliki cookies, 271, 273–276
podręcznik
administratora systemu,
341
użytkownika, 341
pole
input typu file, 269
typu SELECT, 263
polecenie INSERT INTO,
297
polimorfizm, 35
połączenie z bazą danych
w PHP, 290–292
porównania ciągów,
228–229
poszukiwanie i wyliczanie,
19
praca
rownoległa, 21
sekwencyjna, 21
wielowątkowa, 21
private, 248–249
proceduralność, 21
proces przetwarzania kodu
PHP, 170
program, 9
program komputerowy, 30
programowanie, 9
dynamiczne, 19
obiektywne, 30–32, 243
proceduralne, 30
strukturalne, 29

uogólnione, 30
projektowanie aplikacji
internetowych, 42–43
protected, 248–249
przekazywanie danych
z formularza, 261–268
przesłanie składowych,
254–257
pseudokod, 15
public, 248–249

R

rekurencja, 21
reprezentacja algorytmów,
14
rzutowanie typów, 192

S

schemat blokowy, 16–18
selektory, 121–124
semantyka, 11
serwer aplikacji, 40
serwery WWW, 163
sesja, 277
sesje w PHP, 277–287
składnia, 10
heredoc, 187–188
języka JavaScript,
50–56
języka PHP, 183–197
nowdoc, 188
składowe klasy, 243
skrypt, 13
skrypty PHP, 170, 180
slider, 104–106
słowa kluczowe, 9
słowa kluczowe this, 73
sortowanie, 216
bąbelkowe, 23
liczb, 23
sprawdzanie wypełnienia pól
formularza, 106–109
SQL injection, 339
stałe, 197
stałe predefiniowane, 197

struktura
aplikacji internetowej,
39–42
języka PHP, 180–183
SVG, 47
symbole
flagi, 111
wzorca, 110
systemy zarządzania treścią,
315
szablon Beez3, 330

Ś

środowisko sieciowe
w PHP, 287

T

tabela, 79
tablica, 188
\$_FILES, 269
\$_SERVER, 287
\$_SESSION, 277–278
\$GLOBALS, 210
asocjacyjna, 188,
190–191
indeksowana, 188–190
test bezpieczeństwa, 340
testowanie
aplikacji, 337
programu
komputerowego, 27–28
skryptów, 115–117
testy
automatyczne, 338
bezpieczeństwa, 337
czarnej skrzynki, 338
funkcjonalne, 337
jednostkowe, 338
kompatybilności, 337
użyteczności, 337
wydajności, 337
wydajnościowe, 338
z udziałem użytkownika,
338
tłumacz, 9

- tworzenie
- bazy danych, 305
 - obiektów
 - w JavaScript, 72–73
 - w PHP, 246–248
 - z użyciem konstruktora, 73–75
 - pliku cookie w PHP, 271–272
 - programu komputerowego, 26–28
 - tabeli w bazie danych, 305
 - tablicy, 79–80
 - typy danych, 11
 - array
 - w JavaScript, 53
 - w PHP, 188
 - boolean
 - w JavaScript, 53
 - w PHP, 186
 - float, 186
 - integer, 186
 - liczbowy, 52
 - string
 - w JavaScript, 52–53
 - w PHP, 187
 - null
 - w JavaScript, 53
 - w PHP, 191
 - object
 - w JavaScript, 53
 - w PHP, 191
 - resource, 191
 - undefined, 53
 - w JavaScript, 52–53
 - w PHP, 185–193
- U**
- usuwanie pliku cookie, 273
- V**
- Visual Studio Express 2012 for Web, 314
- Visual Studio Express 2012 for Windows Desktop, 314
- Visual Studio, 313
- W**
- walidacja formularzy przy zastosowaniu biblioteki jQuery, 147–150
- w JavaScript, 106–115
- warstwa
 - aplikacji, 40
 - danych, 40
 - klienta, 40
- WEB.PI, 167
- widok, 43
- Windows Forms, 313
- właściwość
 - onreadystatechange, 155
 - readyState, 155
 - responseText, 156
 - responseXML, 157
 - status, 156
- właściwości
 - klasy w PHP, 244
 - obiektu, 70
- wstępne przetwarzanie kodu, 11–12
- wyjątki w PHP, 257–259
- wrażenia regularne, 110
- wysokość drzewa, 22
- wysyłanie żądania Ajax do serwera, 157
- wzorce, 110–114
- wzorzec projektowy, 42
 - MVC, 43–44
- X**
- XAMPP, 173–179
- XML, 153
- XSS, 339
- Z**
- zapytania do bazy danych, 293–305
- zasięg zmiennej, 65, 208
- zastosowanie biblioteki jQuery na stronie internetowej, 131–147
 - animacje, 136–144
 - pokaz zdjęć, 144–147
 - pokazywanie i ukrywanie treści, 132–135
- zdarzenia, 95
 - biblioteki jQuery, 129–131
 - dokumentu, 99
 - elementów formularza, 99
 - formularza
 - w JavaScript, 98
 - w jQuery, 131
 - klawiatury, 98
 - myszy
 - w JavaScript, 96–98
 - w jQuery, 129
- zerowanie pól formularza, 114–115
- zgłaszanie wyjątków w PHP, 257–258
- zintegrowane środowisko programistyczne, 26
- złożoność
 - czasowa algorytmu, 22
 - obliczeniowa algorytmu, 21–22
 - oczekiwana, 22
 - optymistyczna, 22
 - pamięciowa algorytmu, 22
 - pesymistyczna, 22
- zmiana grafiki w JavaScript, 102–104
- zmiana typu zmiennej, 192–193

zmienne, 51, 184
 predefiniowane 185
 sesji, 278
 statyczne, 211–212
 superglobalne, 185
 w JavaScript, 51–52
 w PHP, 184–185
zmienne globalne
 w JavaScript, 65
 w PHP, 209–210
zmienne lokalne
 w JavaScript, 65
 w PHP, 185, 210

znaczniki
 bloku PHP, 180
 formatujące funkcji
 date(), 219–220
znajdowanie
 największego
 elementu w zbiorze
 nieuporządkowanym,
 24
 podciągów, 226–228

Ż

źródło danych dla MySQL,
306

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**



Podręcznik do nauki zawodu **technik informatyk**

Technik informatyk nie jest zwykłym użytkownikiem komputerów. Jeśli uczeń wybiera szkołę o takim profilu, z czasem staje się prawdziwym komputerowym ekspertem.

Kwalifikacje wyodrębnione w zawodzie technik informatyk dotyczą między innymi tworzenia aplikacji internetowych i baz danych oraz administrowania tymi bazami. Trzecia część podręcznika, omawiająca kwalifikację E.14, składa się z dziesięciu rozdziałów, których budowa pozwala na realizację treści programowych w sposób wybrany przez nauczyciela. Przygotowane materiały obejmują nie tylko zagadnienia teoretyczne, ale także projekty różnych zadań oraz – co bardzo istotne – sposoby ich praktycznej realizacji, dzięki czemu prowadzą do uzyskania efektów kształcenia wymienionych w podstawie programowej. Podczas zajęć uczniowie zdobędą niezbędną wiedzę na temat aplikacji internetowych. Nie tylko będą samodzielnie programować w wybranych technologiach, ale też staną się świadomi ich indywidualnych cech, wad i zalet. Poznają działanie serwerów stron internetowych oraz zasady umieszczania opracowanych aplikacji w internecie. Zaczynają wykorzystywać zasoby biblioteki jQuery do tworzenia własnych aplikacji. Będą korzystać z systemów zarządzania treścią CMS przy tworzeniu menu, zamieszczaniu grafiki i artykułów oraz dostosowywaniu aplikacji do potrzeb użytkownika. Na koniec przeprowadzą testy aplikacji oraz przeanalizują zagrożenia. Tak skonstruowany podręcznik pomaga także w samodzielnym poszerzaniu umiejętności.

Technik informatyk to doskonały, charakteryzujący się wysoką jakością, kompletny zestaw edukacyjny, przygotowany przez dysponującego ogromnym doświadczeniem lidera na rynku książek informatycznych – wydawnictwo Helion.

W skład zestawu *Technik Informatyk* wchodzi także:

Kwalifikacja E.12. Montaż i eksploatacja komputerów osobistych oraz urządzeń peryferyjnych.

Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.13. Projektowanie lokalnych sieci komputerowych i administrowanie sieciami.

Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.14. Część 1. Tworzenie stron internetowych.

Podręcznik do nauki zawodu technik informatyk

Kwalifikacja E.14. Część 2. Tworzenie baz danych oraz administrowanie bazami.

Podręcznik do nauki zawodu technik informatyk

Podręczniki oraz inne pomoce naukowe należące do tej serii zostały opracowane z myślą o wykształceniu kompetentnych techników, którzy bez trudu poradzą sobie z wyzwaniami w świecie współczesnej informatyki. Według nowych przepisów, aby otrzymać tytuł technika informatyka, należy potwierdzić trzy kwalifikacje wyodrębnione w tym zawodzie – to niewątpliwe wyzwanie i dla adeptów nauki o komputerach, i dla ich pedagogów. Ta książka pozwoli zarówno przygotować się do egzaminów, jak i uzyskać wiedzę i umiejętności przydatne w przyszłej pracy.

helion.pl
księgarnia
internetowa

Nr katalogowy: **15976**



Księgarnia internetowa:
http://helion.pl



Zamówienia telefoniczne:

0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

🔗 <http://helion.pl/nowości>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po **WIECEJ**



KOD KORZYŚCI

ISBN 978-83-246-8132-7



9 788324 681327