DO NOWEJ PODSTAWY PROGRAMOWEJ

Część 3

Kwalifikacja E.14
Tworzenie aplikacji internetowych

Podręcznik do nauki zawodu
technik informatyk

Jolanta Pokorska

Helion Edukacja

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 Szabłowskiego.

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

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

Printed in Poland.

Spis treści

Zadanie 3.1

Zmodyfikuj kod podany w przykładzie 3.31. Zamiast tworzenia zmiennej `kolor_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 `UpperCase()` 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
```

Zmiennej `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 JĘZYKA 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.

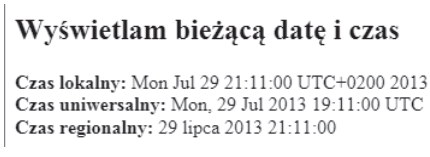
Jednym z najczęściej spotykanych przykładów wykorzystania JavaScript na stronie WWW jest wyświetlanie daty i czasu jako elementu strony internetowej.

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 wyświetlą się informacje o czasie lokalnym, uniwersalnym i regionalnym (rysunek 3.3).



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.

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.

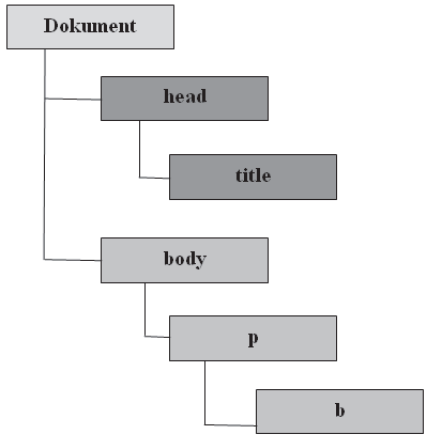
Gdy kod strony jest wczytywany do przeglądarki, przeglądarka zamienia ciąg znaków na stronę internetową. Informacje na temat interpretacji kodu HTML przeglądarka przechowuje w elementach będących obiektami (są to np. informacje o tym, które elementy przedstawić w postaci nagłówków, paragrafów itp.).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

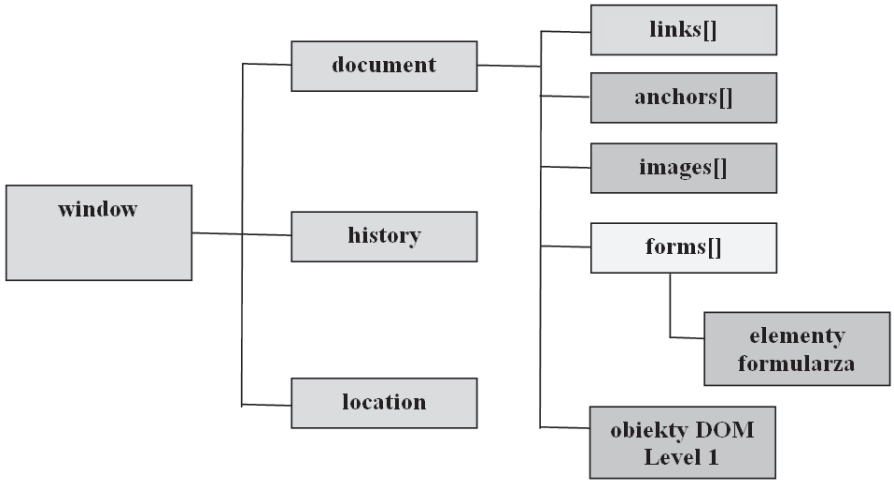
3.8.1. Hierarchia obiektów DOM

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.bgColor` — 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 `getElementByTagName()`. Metoda `getElementById()` używana jest, gdy element, do którego się odwołujemy, posiada atrybut `id`, natomiast metodę `getElementByTagName()` 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.getElementByTagName(p)` — odwołanie do kolekcji akapitów,
- `document.getElementByTagName(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).

Rysunek 3.7. Możliwość zmiany tekstu wyświetlanego na stronie

Przykład 3.64

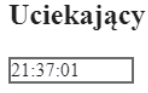
W przykładzie 3.49 zostało zdefiniowane wyświetlanie daty i czasu. Zmiana czasu następowała po odświeżeniu strony. Zmodyfikujemy powstały kod tak, aby czas wyświetlany na stronie był zawsze aktualny.

```
<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.



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 właściwości `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`.

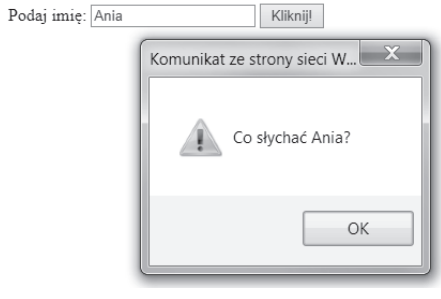
Każdy element formularza jest obiektem, więc posiada właściwości. Jedną z nich jest właściwość `value`, która przechowuje bieżącą wartość elementu.

Przykład 3.69

```
<html>
<head>
<title>Co słychać</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 słychać ' + 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
```

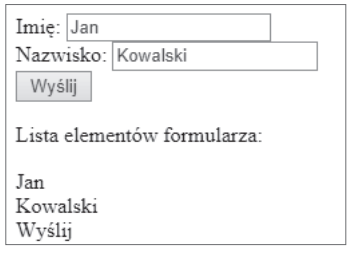
lub za pomocą metody `getElementById()`:

```
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

Skorowidz

F

G

H

I

J

K

L

M

N

O

P

R

S

Ś

T

U

V

W

X

Z

Ź

Kwalifikacja E.14 Część 3

Tworzenie aplikacji internetowych

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. Zaczną 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* wchodzą 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/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-246-8132-7

9 788324 681327

Informatyka w najlepszym wydaniu