JavaScript.
Ćwiczenia praktyczne

Autor: Marcin Lis
ISBN: 83-7197-725-5
Format: B5, stron: 106

Zapraszamy do lektury kolejnej książki z serii „Ćwiczenia praktyczne" dotyczących technologii tworzenia internetowych stron WWW. Tym razem prezentujemy JavaScript, czyli język skryptowy wzbogacający możliwości oferowane przez HTML.

Książka nie jest suchym omówieniem wszystkich konstrukcji i obiektów udostępnianych przez język, jest natomiast praktycznym wprowadzeniem do programowania w JavaScript. Składa się z szeregu ćwiczeń, które przedstawione są w postaci gotowej do uruchomienia w dowolnej przeglądarce obsługującej języki skryptowe.

JavaScript to już obecnie obowiązujący standard, jest to również jeden ze składników DHTML. Pozwala na tworzenie efektów niedostępnych w standardowym HTML, efektów które niegdyś trzeba było tworzyć pomocy technologii takich jak np. skrypty CGI. Przy czym równocześnie jest to prosty język skryptowy, którego zrozumienie i nauka nie powinna sprawić nikomu żadnego problemu. Niewątpliwie jest to więc technologia warta poznania.

Książka przeznaczona jest dla osób początkujących, dopiero zaczynających swoją przygodę z językami skryptowymi. Autor zakłada jednak, że czytelnik zna przynajmniej podstawy języka HTML i jest w stanie samodzielnie utworzyć proste strony. Nie jest natomiast potrzebna znajomość klasycznych języków programowania takich jak C, C++ czy Java.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.

Pierwsze skrypty

Na początku zajmijmy się klasycznym przykładem, od którego zaczyna się większość kursów programowania. Postarajmy się wyświetlić na ekranie dowolny napis np. `Jaki miły mamy dzień!`. Aby tego dokonać, wpierw musimy dowiedzieć się, w jaki sposób umieszczać skrypty JavaScript w kodzie HTML oraz jaka instrukcja JavaScript pozwala pisać na ekranie.

Znacznik <SCRIPT>

Kod JavaScript musi być umieszczony pomiędzy znacznikami HTML `<SCRIPT>` i `</SCRIPT>`. Znaczniki te można umieszczać w dowolnym miejscu dokumentu, jednak przyjmuje się, że jeżeli jest to tylko możliwe, należy umieścić je na początku pliku HTML przed znacznikiem `<BODY>`.

Znacznik ten powinien zawierać parametr `LANGUAGE`, który może przyjmować dwie wartości: `LiveScript` lub `JavaScript`. Wartość `LiveScript` jest pozostałością po wcześniejszych wersjach języka i służy zachowaniu kompatybilności. Powinniśmy użyć wartości `JavaScript`.

Ćwiczenie 2.1.

Umieść w standardowym kodzie HTML znacznik `<SCRIPT>`.

```
<HTML>
<HEAD>
</HEAD>
<SCRIPT language = "JavaScript">
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Na listingu znajduje się poprawny kod HTML z zawartym znacznikiem <SCRIPT>. Jednak po wczytaniu tego pliku do przeglądarki otrzymamy pustą stronę. Brakuje nam instrukcji pozwalającej wyświetlać tekst.

Instrukcja document.write

Instrukcja `document.write()` pozwala na wyprowadzenie tekstu na ekran przeglądarki. Tekst, który chcemy wyświetlić, należy ująć w nawiasy i cudzysłowy i podać zaraz za `document.write()` np.

```
document.write ("Jaki miły mamy dzień!")
```

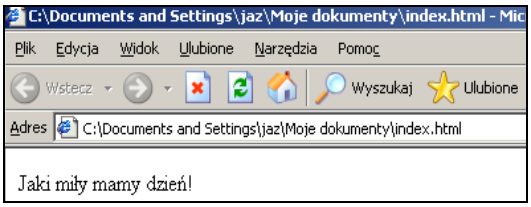
Ćwiczenie 2.2.

Napisz skrypt wyświetlający tekst "Jaki miły mamy dzień!" na ekranie przeglądarki.

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT language = "JavaScript">
document.write ("Jaki miły mamy dzień!")
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Tak przygotowany kod spowoduje, że na ekranie pojawi się pożądany napis (rysunek 2.1). Warto zwrócić uwagę, że w celu poprawnej interpretacji polskich liter przez przeglądarkę dodaliśmy w sekcji `HEAD` znacznik `<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">`

Rysunek 2.1. *Efekt działania instrukcji document.write()*



Przeanalizujmy nieco dokładniej fragment kodu odpowiedzialny za wyświetlanie tekstu na ekranie. Wszystkim, którzy mieli już wcześniej do czynienia z językiem C bądź C++, składnia wydaje się z pewnością znajoma:

```
document.write ("Jaki miły mamy dzień")
```

`document` to obiekt, który reprezentuje aktualną stronę. `write` to tzw. metoda, czyli pewna funkcja działająca na obiekcie `document` i, w tym przypadku, wyświetlająca na ekranie tekst. Tekst ten podajemy jako argument w nawiasach. Ogólnie można zapisać:

```
obiekt.metoda (argumenty metody)
```

Taki ciąg jest instrukcją i powinien zostać zakończony średnikiem. W JavaScript nie jest to jednak obligatoryjne, chyba że chcemy zapisać kilka instrukcji w jednej linii np.:

```
document.writeln ("Witamy");document.write ("na naszej stronie");
```

Wymieniona tutaj, nowa funkcja `writeln()` działa tak samo jak `write()`, z tym że na końcu wyświetlanego ciągu znaków dodaje znak przejścia do nowego wiersza. Niestety, nie zobaczymy tego efektu, jeżeli całość nie znajdzie się w bloku tekstu preformatowanego, tzn. pomiędzy znacznikami `<PRE>` i `</PRE>`.

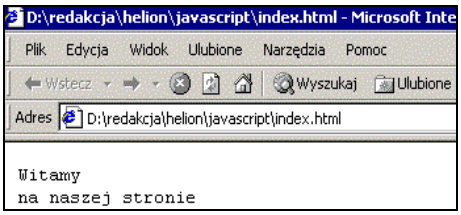
Ćwiczenie 2.3.

Użyj funkcji `write()` i `writeln()` do wyświetlenia tekstu w dwóch wierszach.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<PRE>
<SCRIPT>
document.writeln ("Witamy");document.write ("na naszej stronie");
</SCRIPT>
</PRE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Jak widać na rysunku 2.2, zadanie udało nam się wykonać znakomicie.

Rysunek 2.2. *Użycie instrukcji writeln() i znacznika <PRE>*



Komentarze

Komentarz HTML

Znacznik `<SCRIPT>`, niezbędny do umieszczania kodu JavaScript, niestety nie jest częścią specyfikacji HTML 2.0, ani wcześniejszych, więc niektóre przeglądarki mogą go nie rozpoznać. W takiej sytuacji mogą one wyświetlić tekst skryptu na stronie. Chcielibyśmy oczywiście tego uniknąć. Z pomocą przyjdą komentarze, które można umieszczać w kodzie HTML. Konstrukcja wygląda następująco:

```
<!--
Tekst komentarza
-->
```

Jeżeli zatem chcemy ukryć kod przed przeglądarkami nieobsługującymi JavaScript, powinniśmy ująć go w znaki komentarza, które są częścią standardu HTML.

Znacznik <SCRIPT>, niezbędny do umieszczania kodu JavaScript, niestety nie jest częścią specyfikacji HTML 2.0, ani wcześniejszych, więc niektóre przeglądarki mogą go nie rozpoznać. Co się stanie w takiej sytuacji? Otóż sam znacznik zostanie zignorowany, natomiast cały tekst skryptu znajdujący się między <SCRIPT> a </SCRIPT> zostanie wyświetlony na ekranie, zmieniając nam treść i strukturę strony. Chcielibyśmy oczywiście tego uniknąć. Z pomocą przyjdzie nam komentarz HTML, którego struktura wygląda następująco:

```
<!--
Tekst komentarza
-->
```

Jeżeli ujmiemy tekst skryptu w taką strukturę, przeglądarka nieobsługująca JavaScriptu pominie go, traktując właśnie jako zwykły komentarz.

Ćwiczenie 2.4.

Ukryj kod skryptu przed przeglądarkami nieobsługującymi JavaScript.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
document.write ("Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Powyższe ćwiczenie obrazuje użycie komentarzy znanych z języka HTML. W JavaScript mamy natomiast dwie nowe możliwości zastosowania komentarza. Obie są zapożyczone z języków programowania takich C, C++ czy Java. Pierwszy typ komentarza składa się z dwóch ukośników: // (komentarz ten został zastosowany w poprzednim przykładzie, bowiem wczesne wersje przeglądarki Netscape Navigator nie rozpoznawały poprawnie sekwencji --> umieszczonej między etykietami <SCRIPT>). Zaczyna się on wtedy od miejsca wystąpienia tych dwóch znaków i obowiązuje do końca danego wiersza.

Komentarz typu //

Ćwiczenie 2.5.

Użyj komentarza składającego się z dwóch ukośników do opisania kodu skryptu.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
// Wyświetlenie napisu w oknie przeglądarki
document.write ("Hello, Jaki miły mamy dzień!")
```

```
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Komentarz blokowy

Komentarz może się również zaczynać od sekwencji /* i kończyć */. W takim przypadku wszystko, co znajduje się pomiędzy tymi znakami, uznane zostanie za komentarz.

Ćwiczenie 2.6.

Użyj komentarza blokowego do opisania kodu skryptu.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nie osbługującymi JavaScript
/*
Komentarz blokowy
Wyświetlenie napisu w oknie przeglądarki
*/
document.write ("Hello, Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Znacznik <NOSCRIPT>

W jaki sposób jednak poinformować użytkownika przeglądarki nieobsługującej JavaScriptu, że strona taki skrypt zawiera, tylko nie został wykonany? Z pomocą przyjdą nam również komentarze.

Ćwiczenie 2.7.

Napisz kod, który po wczytaniu do przeglądarki nieobsługującej JavaScript wyświetli stosowny komunikat.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
// Twoja przeglądarka nie obsługuje JavaScript
// Sugerujemy użycie przeglądarki Netscape Navigator
// lub Microsoft Internet Explorer!
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
```

```
document.write ("Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Przeglądarka nieobsługująca skryptów po napotkaniu nieznanej sobie etykiety <SCRIPT> ignoruje ją, następnie wyświetla dwa kolejne wiersze, traktując je jako zwykły HTML. Następne wiersze są dla niej komentarzem, więc je pomija. Z kolei dla przeglądarki obsługującej skrypty komentarzem są dwa wiersze następujące po etykiecie <SCRIPT> i to one są pomijane, natomiast kod z piątego wiersza skryptu (`document.write ("Jaki miły mamy dzień!")`) jest interpretowany i wykonywany.

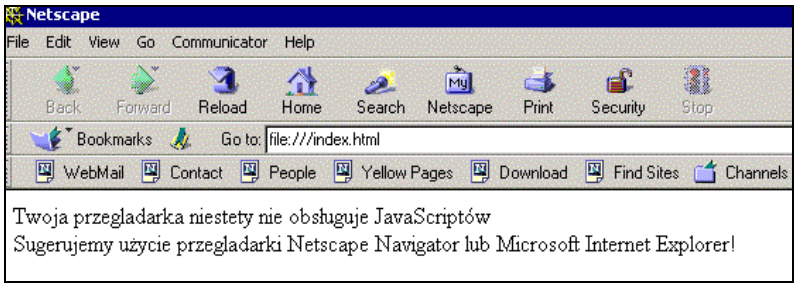
Jest też jeszcze inny sposób na wykonanie tego zadania. Przeglądarki Netscape Navigator oraz Internet Explorer, obie od wersji 3.0, akceptują dodatkowy znacznik <NOSCRIPT>. Dzięki niemu możemy osiągnąć podobny efekt. W tym przypadku tekst, który ma być wyświetlony, gdy wyłączymy skrypty w danej przeglądarce, umieszczamy pomiędzy znacznikami <NOSCRIPT> i </NOSCRIPT>.

Ćwiczenie 2.8.

Użyj znacznika <NOSCRIPT> do poinformowania użytkownika, że jego przeglądarka nie obsługuje JavaScriptu.

```
<HTML>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
document.write ("Jaki miły mamy dzień!")
// Koniec kodu JavaScript -->
</SCRIPT>
<NOSCRIPT>
Twoja przeglądarka niestety nie obsługuje JavaScriptów.<BR>
Sugerujemy użycie przeglądarki Netscape Navigator lub Microsoft Internet Explorer!
</NOSCRIPT>
<BODY>
</BODY>
</HTML>
```

Na rysunku 2.3 widoczny jest efekt działania powyższego kodu w przeglądarce Netscape Navigator po wyłączeniu działania skryptów.



Rysunek 2.3. *Zastosowanie znacznika <NOSCRIPT> do poinformowania użytkownika, że jego przeglądarka nie obsługuje JavaScriptu*

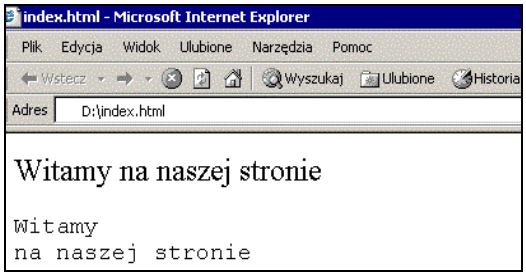
Formatowanie tekstu

Argumenty poznanych wyżej funkcji `write()` i `writeln()` są traktowane przez przeglądarkę jak tekst w HTML-u. Oznacza to, że możemy w łańcuchach wyświetlanych znaków wstawić praktycznie dowolne znaczniki formatujące tekst.

Ćwiczenie 2.9.

Użyj znaczników HTML formatujących tekst w argumentach funkcji `write()` i `writeln()`, tak by osiągnąć efekt jak na rysunku 2.4.

Rysunek 2.4. *Efekt użycia znaczników HTML w argumentach funkcji write() i writeln()*




```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptów
document.write ("<FONT SIZE=+2>Witamy ");
document.write ("na naszej stronie");
document.writeln ("<PRE>Witamy");
document.write ("na naszej stronie</PRE></FONT>");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Oprócz znaczników HTML w wyświetlanych łańcuchach znakowych mogą też pojawić się znaki specjalne, takie jak np. rozpoczęcie nowego wiersza. Jeśli chcemy wyświetlić znak specjalny, musimy zastosować sekwencję — ukośnik (backslash) plus litera symbolizująca dany znak. Sekwencje te przedstawione są w tabeli 2.1.

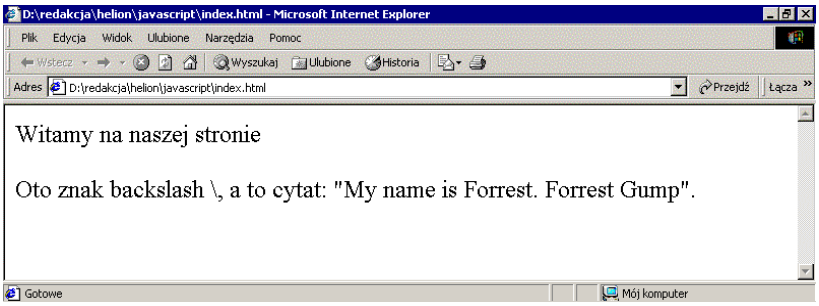
Tabela 2.1. *Sekwencje znaków specjalnych*

Sekwencja znaków specjalnych	Znaczenie
`\b`	backspace
`\f`	wysunięcie kartki (ang. *form feed*)
`\n`	nowy wiersz (ang. *new line character*)
`\r`	enter (ang. *carriage return*)
`\t`	tabulator (ang. *tab character*)

Podobnie, jeżeli chcemy wyświetlić cudzysłów lub sam ukośnik (backslash \), musimy go poprzedzić znakiem backslash.

Ćwiczenie 2.10.

Używając funkcji `write()` wyprowadź na ekran tekst zawierający znak cudzysłowu oraz ukośnik (rysunek 2.5).



Rysunek 2.5. *Wyprowadzenie na ekran znaków specjalnych*

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScript
document.write ("<FONT SIZE=+2>Witamy ");document.write ("na naszej stronie<BR><BR>");
document.write ("Oto znak backslash \\");document.write (", a to cytat: \"My name is
Forrest. Forrest Gump\".");
document.write ("</FONT>");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

W ten sam sposób możemy również pokusić się o wyświetlenie grafiki. Jeżeli argumentem funkcji `write()` będzie znacznik `<IMG>` z odpowiednim URL-em jako parametrem, przeglądarka wyświetli na stronie wskazany w ten sposób obrazek np.

```
document.write ("<IMG SRC = /javasc/gfx/grafika1.gif>");
```

Oczywiście, plik o lokalizacji */javasc/gfx/grafika1.gif* musi istnieć, abyśmy mogli zobaczyć efekt w oknie przeglądarki. Formalnie rzecz biorąc, powinniśmy wartość argumentu `SRC` ująć w cudzysłów, zatem zgodnie z tym, co wiemy już o znakach specjalnych, konstrukcja powinna wyglądać następująco:

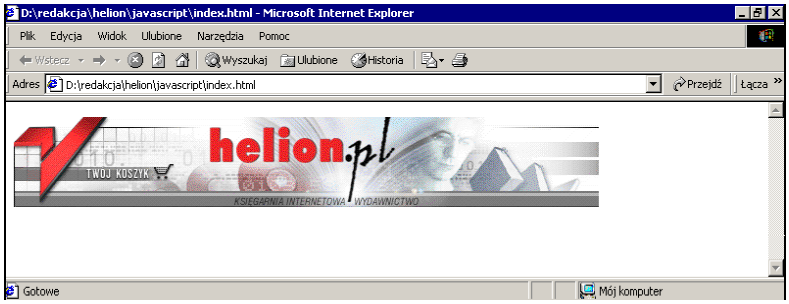
```
document.write ("<IMG SRC = \"/javasc/gfx/grafika1.gif\">");
```

Ćwiczenie 2.11.

Użyj funkcji `write()` do wyświetlenia na ekranie pliku graficznego (rysunek 2.6).

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

Rysunek 2.6. *Przykład użycia funkcji write() do wyświetlenia pliku graficznego*




```
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
document.write ("<IMG SRC = \"/javasc/gfx/grafika1.gif\">");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

Okno dialogowe

Nauczymy się teraz, jak wyświetlić na ekranie najprostsze okienko dialogowe. Okno takie służy zwykle do poinformowania użytkownika o wystąpieniu jakiegoś zdarzenia. Najczęściej chodzi o sytuacje, w której wystąpił błąd. Na taki charakter prezentowanej metody wskazuje już sama nazwa: `alert()`. Może ona przyjmować jako parametr ciąg znaków, który zostanie wyświetlony na ekranie.

Ćwiczenie 2.12.

Wyświetl na ekranie okno dialogowe z dowolnym napisem.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
alert("To jest okno dialogowe");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

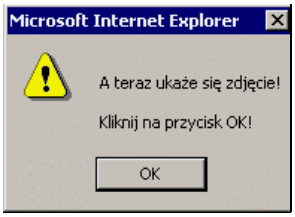
Nasze okno wygląda jak na rysunku 2.7. Wykonywanie kodu jest wstrzymane do czasu, kiedy użytkownik kliknie przycisk *OK*. Dokładniej rzecz biorąc, w taki sposób powinna się zachować większość współczesnych przeglądarek. Tekst wyświetlany w oknie dialogowym możemy formatować, używając do tego celu znaków specjalnych (tabela 2.1), podobnie jak w przypadku funkcji `write()`.

Rysunek 2.7.
Użycie funkcji alert() do wyświetlenia okna dialogowego

Ćwiczenie 2.13.

Wyświetl na ekranie okno dialogowe z tekstem w dwóch wierszach (jak na rysunku 2.8).

Rysunek 2.8.
Użycie znaków specjalnych formatujących tekst w oknie dialogowym



Spowoduj, aby po kliknięciu przez użytkownika przycisku *OK* na ekranie pojawił się plik graficzny.

```
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
</HEAD>
<SCRIPT LANGUAGE = "JavaScript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptów
alert ("\nA teraz ukaże się zdjęcie!\n\nKliknij na przycisk OK!")
document.write ("<IMG SRC = \"obrazek1.gif\">");
// Koniec kodu JavaScript -->
</SCRIPT>
<BODY>
</BODY>
</HTML>
```