1 Umieszczanie skryptów w dokumencie

ZAGADNIENIA

- **Czym jest JavaScript?**
- **Jakie są sposoby na umieszczenie skryptu JavaScript?**
- **Jak umieszczać skrypt w dokumencie HTML?**
- **Jak korzystać ze skryptów zewnętrznych?**

JavaScript jest językiem skryptowym pozwalającym na rozszerzenie standardowych dokumentów HTML m.in. o możliwość interakcji z użytkownikiem oraz na sprawdzanie poprawności informacji wprowadzanych przez użytkowników. Powstał w połowie lat 90. XX wieku w firmie Netscape jako język skryptowy LiveScript. W 1995 roku w wyniku współpracy z firmą Sun Microsystems ugruntował się jako język JavaScript. Jego głównym autorem jest Brendan Eich.

Program napisany w języku programowania wysokiego poziomu (C, C++, JAVA) przed zastosowaniem **kodu źródłowego**, którym jest uporządkowany ciąg instrukcji, wymaga skompilowania. Procesem tym zajmuje się **kompilator**, program którego zadaniem jest przetłumaczenie całego kodu źródłowego na **kod maszynowy**, ciąg liczb w systemie dwójkowym zrozumiały dla procesora. Podobne zadanie wykonuje **interpreter**. Jego sposób działania opiera się na pobraniu pojedynczej linii kodu, przetłumaczeniu jej, przekazaniu do procesora i przejściu do kolejnej linii kodu. Proces ten powtarza się do czasu przetłumaczenia całego programu.

JavaScript jest interpretowanym językiem programowania. Żeby zobaczyć efekty działania programu, nie trzeba go kompilować do kodu maszynowego. Wystarczy przeglądarka internetowa mająca włączoną obsługę języka JavaScript.

Jest to jeden z popularniejszych języków programowania działających po stronie klienta. To jedna z największych jego zalet, ponieważ wszystkie przeprowadzane operacje nie obciążają serwera. Zapewnia on również odpowiedni poziom bezpieczeństwa, ponieważ nie ma dostępu do systemu plików znajdujących się na komputerze użytkownika. Nieco trudności sprawia jedynie jego interpretacja przez różne przeglądarki. Może się zdarzyć, że prawidłowo napisany program będzie różnie działał w zależności od obsługującej go przeglądarki.

Sam język jest łatwy do opanowania. Do stworzenia dowolnego programu wystarczy zwykły edytor tekstu (np. Notatnik) lub dowolnie wybrane narzędzie wspomagające tworzenie strony WWW (Pajączek, Zajączek, Notepad++ i wiele innych). Wyniki działania programu obserwować można w dowolnie wybranej przeglądarce. Warto jednak pamiętać, żeby testować dany program w kilku dostępnych przeglądarkach (istnieją różnice w interpretacji).Wymagana jest również podstawowa wiedza z zakresu języka HTML.

Pierwszym sposobem na umieszczenie skryptu w dokumencie HTML jest wprowadzenie go bezpośrednio do kodu przez wykorzystanie znacznika **<script>** ... **</script>**. Można go wprowadzić w nagłówku **head** lub w głównej części dokumentu **body**.

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

Parametr **type** określa rodzaj języka skryptowego. Nie jest on jednak wymagany. Większość współczesnych przeglądarek zaakceptuje sam znacznik **script**, warto jednak stosować zapis **script** w całości. Natomiast całą treść skryptu (kodu) wprowadza się pomiędzy znaczniki.

Istnieje możliwość umieszczenia skryptu w osobnym pliku. Plik taki nazywamy skryptem zewnętrznym. Może on mieć dowolną nazwę oraz charakterystyczne rozszerzenie **.js** . Plik zewnętrzny zostaje powiązany z dokumentem HTML za pomocą znacznika **script** z dodatkowym oznaczeniem lokalizacji i nazwy pliku wraz z rozszerzeniem.

```
<script type="text/javascript" src="lokalizacja/nazwa.js">
</script>
```

W jednym dokumencie można umieścić kilka skryptów, zarówno osadzonych, jak i zewnętrznych. W przykładzie (list. 1.1) w nagłówku **head** wprowadzono skrypt zewnętrzny, a w głównej części dokumentu **body** – skrypt osadzony.

Listing 1.1

```
<html>
   <head>
   <title>
   Strona ze skryptem JavaScript
   </title>
   <meta http-equiv="Content-Type" content="text/html; charset=
iso-8859-2">
   <meta http-equiv="Content-Language" content="pl">
   <script type="text/javascript" src="lokalizacja/nazwa.js">
   </script>
   </head>
   <body>
<script type="text/javascript">
</script>
   </body>
</html>
```

Obecnie w internecie wiele stron oferuje gotowe przykłady skryptów, które można wykorzystać na stronie, nawet nie znając języka. Mogą to być elementy takie, jak: zegary, kalendarze, kalkulatory, efekty tła, przyciski i wiele innych.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przeszukaj dostępne w internecie strony internetowe oferujące gotowe skrypty. Wybierz jeden. Umieść go na stronie i sprawdź jego działanie na różnych przeglądarkach.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Co to jest JavaScript?
2. Jakie znasz sposoby włączenia skryptu do dokumentu HTML?
3. Co jest potrzebne do uruchomienia skryptu napisanego w języku JavaScript?
4. Porównaj działanie kompilatora i interpretera.

2 Instrukcja dokument.write

ZAGADNIENIA

- **Za co odpowiada instrukcja dokument.write?**
- **Jak zmienić wygląd tekstu?**
- **Jak wyświetlać tekst na stronie za pomocą języka JavaScript?**
- **Jak stosować elementy języka HTML wewnątrz skryptu?**

Kod języka JavaScript zbudowany jest z instrukcji. Instrukcją odpowiadającą za wyświetlenie tekstu na stronie jest **document.write**. Ciąg znaków, który ma zostać wyświetlony, należy umieścić w nawiasie okrągłym oraz w cudzysłowie. Na końcu każdej instrukcji znajduje się średnik:

```
document.write("Wyświetlany tekst");
```

Przed przystąpieniem do opisu instrukcji należy zapoznać się z zapisem ogólnym:

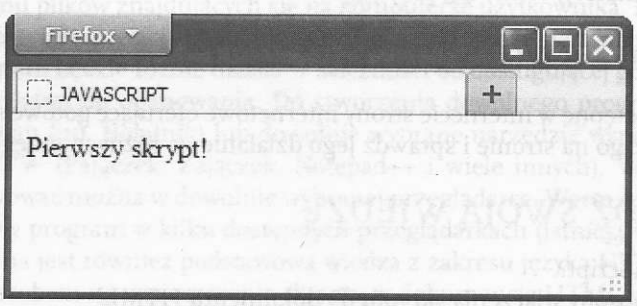
```
obiekt.metoda(argumenty metody);
```

Jak widać, **document** jest obiektem. Reprezentuje on aktualną stronę. Metoda **write** to funkcja działająca na obiekcie **document**, której zadaniem jest wyświetlenie argumentów (tekstu, wartości liczbowych) w oknie przeglądarki.

Pierwszy skrypt (list 2.1) ma za zadanie wyświetlić pojedynczą linijkę tekstu na stronie. Został on umieszczony w głównej części (**body**) dokumentu HTML.

Listing 2.1

```
<script type="text/javascript">
  document.write("Pierwszy skrypt!");
</script>
```



Rys. 2.1. Wynik działania skryptu z list. 2.1

Instrukcja **document.write** pozwala również na wyświetlanie wartości liczbowych. W takim przypadku wystarczy wprowadzić określoną wartość w nawiasie okrągłym bez cudzysłowu:

```
document.write(31);
```

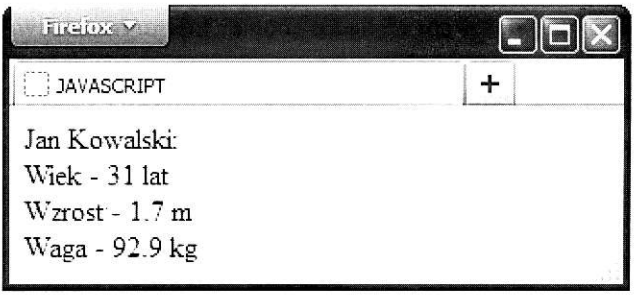
W jednej instrukcji można również umieścić bardziej rozbudowaną treść, składającą się zarówno z tekstu, jak i wartości liczbowych. Należy wówczas zastosować znak **+** do połączenia obu typów.

```
document.write("Wyświetlany tekst ma "+27+" znaków!");
```

Przykładowy skrypt (list. 2.2) prezentuje wyświetlanie tekstu oraz wartości liczbowych całkowitych i zmiennoprzecinkowych. W przypadku liczb ułamkowych wpisana w skrypcie wartość 1.70 po wyświetleniu na stronie została zaokrąglona do 1.7.

Listing 2.2

```
<script type="text/javascript">
document.write("Jan Kowalski: <br>");
document.write("Wiek - "+31+" lat <br>");
document.write("Wzrost - "+1.70+" m <br>");
document.write("Waga - "+92.9+" kg <br>");
</script>
```



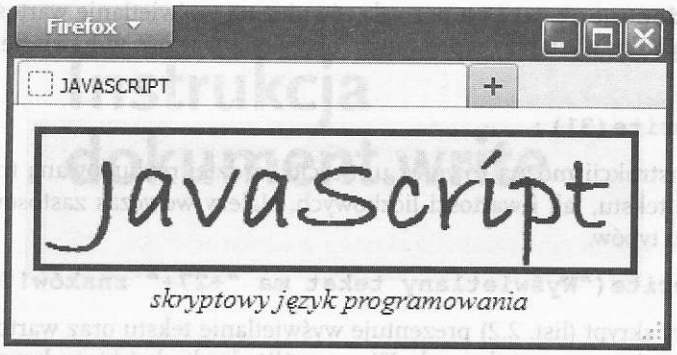
Rys. 2.2. Wynik działa skryptu z list. 2.2

Wprowadzenie kilku instrukcji **document.write** nie spowoduje wyświetlenia ich zawartości w nowej linii. Ponieważ wyświetlana zawartość traktowana jest jak kod HTML, w list. 2.2 wprowadzono dodatkowo znacznik **
**, który odpowiada za przejście do nowej linii.

Wykorzystanie zagnieżdżonych znaczników HTML pozwala w łatwy sposób sformatować informacje wyświetlane za pomocą instrukcji **dokument.write**, a nawet wstawić obrazek (list. 2.3).

Listing 2.3

```
<script type="text/javascript">
document.write("<img src=\"js.png\"<br>");
document.write("<center><i>skryptowy język programowania</center></i>");
</script>
```



Rys. 2.3. Wynik działania skryptu z list. 2.3

Podczas tworzenia skryptów zdarza się, że chcemy ukryć fragment programu lub dodać opis. W tym celu należy wprowadzić **komentarz**. JavaScript daje możliwość zastosowania dwóch rodzajów komentarzy. Pierwszy jest jednowierszowy (liniowy). Rozpoczyna się od znaków `//` i działa do końca danej linii skryptu.

```
// komentarz wierszowy
```

Komentarz wielowierszowy rozpoczyna się od znaków `/*` i kończy się `*/`. W takim komentarzu można umieścić komentarz liniowy, ale zabronione jest zagnieżdżanie w nim komentarzy wielowierszowych.

```
/*
komentarz
wielowierszowy
//z dodatkowym komentarzem liniowym
*/
```

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Zmodyfikuj skrypt z list. 2.2. Wprowadź swoje dane. Zmień formatowanie wyświetlanego tekstu i wprowadź zdjęcie.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Omów instrukcję `dokument.write`.
2. Jak można sformatować tekst wyświetlany przez skrypt?
3. Jak wprowadzić komentarz w skrypcie?
4. W jakim celu stosuje się komentarze w językach programowania?

3 Okno dialogowe

ZAGADNIENIA

- **Co to jest okno dialogowe?**
- **Jakie są rodzaje okien dialogowych?**
- **Jak wyświetlać okno informacyjne?**
- **Jak wyświetlać okno decyzyjne?**
- **Jak wyświetlać okno tekstowe?**

Okno dialogowe jest narzędziem pozwalającym na nawiązanie interakcji z użytkownikiem. JavaScript umożliwia wykorzystanie trzech rodzajów okien dialogowych: informacyjnego, decyzyjnego i tekstowego.

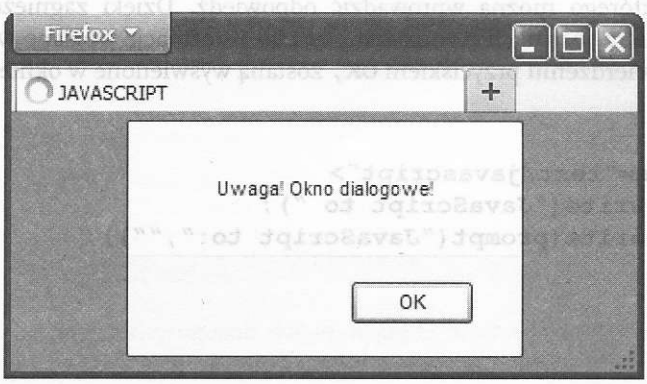
Zadaniem **okna informacyjnego** jest przekazanie określonej informacji. Nie ma ono wpływu na dalsze działanie skryptu. Jego budowa jest wyjątkowo prosta. Wyświetla ono tekst określony jako argument metody **alert** i ma jeden przycisk **OK**, powodujący zamknięcie okna.

```
alert("treść komunikatu");
```

Skrypt z list. 3.1 prezentuje okno dialogowe wyświetlające tekst **"Uwaga! Okno dialogowe!"**. Wygląd okna może nieco różnić się w zależności od przeglądarki internetowej.

Listing 3.1

```
<script type="text/javascript">
alert("Uwaga! Okno dialogowe!");
</script>
```



Rys. 3.1. Wynik działania skryptu z list. 3.1 – okno informacyjne

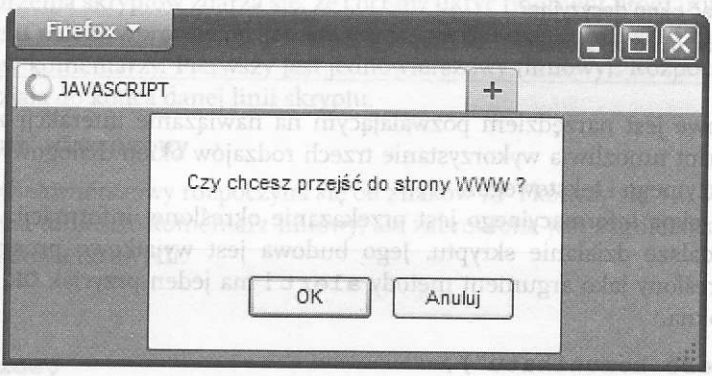
Okno decyzyjne odpowiada za wyświetlenie treści komunikatu stanowiącego argument metody **confirm**. Udostępnia dwa przyciski **OK** oraz **Anuluj**, które po wciśnięciu zwracają wartość logiczną **true** lub **false**.

```
confirm("treść komunikatu");
```

Skrypt z list. 3.2 prezentuje okno decyzyjne wyświetlające tekst **"Czy chcesz przejść do strony WWW?"**. Ponieważ skrypt nie ma żadnej funkcji podpiętej do okna decyzyjnego, wciśnięcie dowolnego klawisza nie wywoła żadnej reakcji.

Listing 3.2

```
<script type="text/javascript">
  confirm("Czy chcesz przejść do strony WWW ?");
</script>
```



Rys. 3.2. Wynik działania skryptu z list. 3.2 – okno decyzyjne

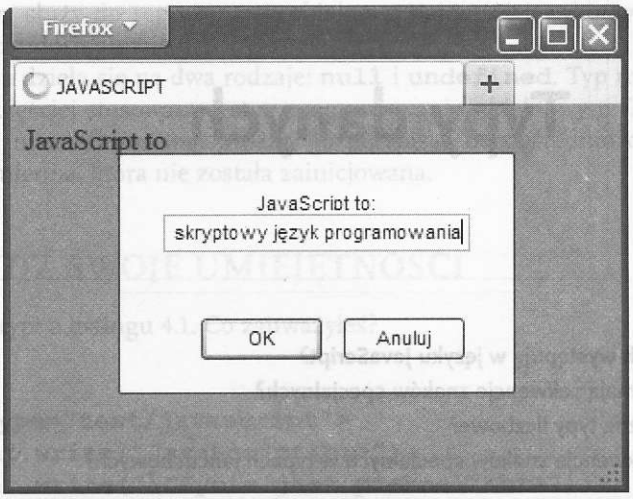
Okno tekstowe wyświetla treść komunikatu stanowiącego argument metody **prompt** oraz pole umożliwiające wprowadzenie danych przez użytkownika. W trakcie wywoływania okna tekstowego w polu może pojawić się tekst domyślny.

```
prompt("treść komunikatu","tekst domyślny");
```

Skrypt list. 3.3 prezentuje okno tekstowe wyświetlające treść **"JavaScript to:"** oraz pole, do którego można wprowadzić odpowiedź. Dzięki zagnieżdżeniu metody **prompt** wewnątrz instrukcji **document.write** informacje wpisane w polu okna tekstowego, po zatwierdzeniu przyciskiem **OK**, zostaną wyświetlone w oknie przeglądarki.

Listing 3.3.

```
<script type="text/javascript">
  document.write("JavaScript to ");
  document.write(prompt("JavaScript to:",""));
</script>
```



Rys. 3.3. Wynik działa skryptu z list. 3.3 – okno tekstowe

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Utwórz skrypt wyświetlający trzy różne okienka dialogowe. Tekst wprowadzony w okienku tekstowym wyświetl w oknie przeglądarki.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj różne okna dialogowe.
2. Jak wyświetlić okno dialogowe na stronie?
3. Podaj praktyczne przykłady zastosowania okien dialogowych.

4 Typy danych

ZAGADNIENIA

- **Jakie typy danych występują w języku JavaScript?**
- **Jakie znaczenie mają sekwencje znaków specjalnych?**
- **Jak stosować różne typy liczbowe?**
- **Jak stosować sekwencje znaków specjalnych w typach łańcuchowych?**

Język JavaScript udostępnia kilka typów danych. Typ danych to zbiór wartości, jakie mogą przyjmować dane. Należą do nich: typ liczbowy, łańcuchowy, logiczny, obiektowy oraz typy specjalne.

Typ liczbowy reprezentuje różnego rodzaju liczby. W porównaniu z innymi językami programowania nie uwzględnia on podziału na liczby całkowite i zmiennoprzecinkowe. Umożliwia wprowadzanie liczb w postaci dziesiętnej (np. 12 lub 14), ósemkowej (np. 012) lub szesnastkowej (np. 0xBD). Dozwolona jest również notacja wykładnicza w postaci X.YeZ, gdzie X stanowi część całkowitą, Y – część dziesiętną, a Z jest wykładnikiem potęgi liczby 10 (np. 0.1e2).

Typ łańcuchowy to dowolne ciągi znaków. Należy umieścić je w cudzysłowie lub pomiędzy znakami apostrofów. Mogą dodatkowo zawierać sekwencje znaków specjalnych (tab. 4.1).

Tabela 4.1. Sekwencje znaków specjalnych

Sekwencja	Znaczenie
\b	backspace
\n	nowy wiersz
\r	powrót karetki
\f	nowa strona
\t	tabulacja pozioma
\"	cudzysłów
\'	apostrof
\\	lewy ukośnik

Typ logiczny może przyjmować jedną z dwóch dostępnych wartości: `true` (prawda) oraz `false` (fałsz). Stosowany jest głównie przy budowaniu wyrażeń logicznych lub do porównywania danych.

Typ obiektowy służy do reprezentacji obiektów. Najczęściej wykorzystuje się obiekty wbudowane oraz udostępniane przez przeglądarkę.

Typy specjalne dzielą się na dwa rodzaje: **null** i **undefined**. Typ **null** określa wartość pustą. Najczęściej stosowany jest w programowaniu obiektowym. Typ **undefined** określa wartość niezdefiniowaną. Można go przypisać bezpośrednio do zmiennej lub przyjmuje go zmienna, która nie została zainicjowana.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przetestuj skrypt z listingu 4.1. Co zauważyłeś?

Listing 4.1

```
<script type="text/javascript">
   document.write("JavaScript<br>");
   document.write("\"Życie jest piękne\"<br>");
   document.write(15+"<br>");
   document.write(100.4+"<br>");
   document.write(-26+"<br>");
   document.write(0.1e2+"<br>");
   document.write(0xAA+"<br>");
   document.write(-0xCD);
</script>
```

SPRAWDŹ SWOJĄ WIEDZĘ

1. Scharakteryzuj typy danych dostępne w języku JavaScript.
2. Jakie znasz sekwencje znaków specjalnych?

5 Zmienne

ZAGADNIENIA

- **Co to jest zmienna?**
- **Jakie są zasady deklaracji zmiennej w języku JavaScript?**
- **Jak deklarować zmienne?**
- **Jak przypisać wartość do zmiennej?**

Zmienna jest to element programu pozwalający na przechowywanie danych różnych typów. W języku JavaScript, w odróżnieniu od innych języków programowania, nie wymaga się podania typu zmiennej podczas jej deklaracji. Ponadto typ zmiennej może ulec modyfikacji w trakcie wykonywania skryptu – np. zmiennej typu łańcuchowego możemy przypisać wartość liczbową.

Deklaracja zmiennej odbywa się przez nadanie jej jednoznacznej nazwy, przez którą jest identyfikowana. Przed nazwą należy wprowadzić instrukcję **var**:

```
var nazwa_zmiennej;
```

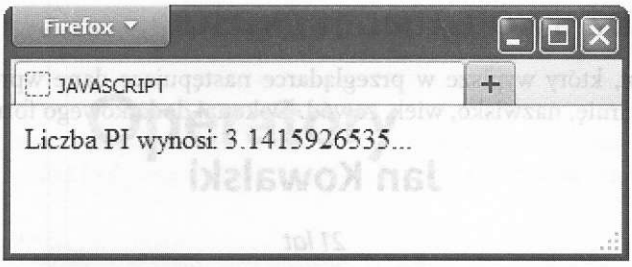
Nazwa zmiennej musi zaczynać się od litery. Wewnątrz jej składni dopuszcza się stosowanie liter, cyfr i znaku podkreślenia (_).

Przykład z list. 5.1 prezentuje deklarację dwóch zmiennych. Zmiennej **zmienna_1** przypisano ciąg znaków umieszczonych w cudzysłowie. Oznacza to, że **zmienna_1** jest typu łańcuchowego. **Zmienna_2** jest typu liczbowego, przypisano jej wartość zmiennoprzecinkową. Za pomocą instrukcji **document.write** wypisano wartości obu zmiennych w oknie przeglądarki (rys. 5.1).

Listing 5.1

```
<script type="text/javascript">
   var zmienna_1="Liczba PI wynosi: ";
   var zmienna_2=3.1415926535;
   document.write(zmienna_1+zmienna_2+"...");
</script>
```

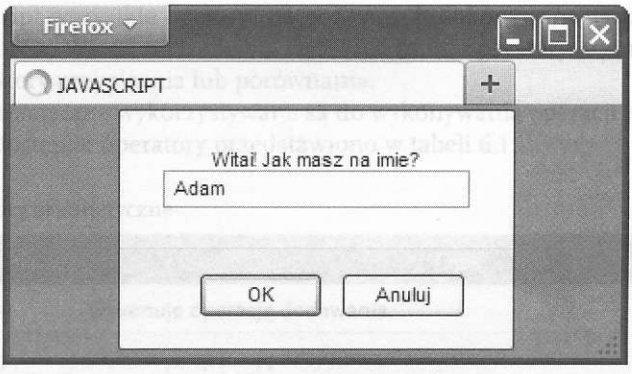
Przykład z list. 5.2 prezentuje, jak przy wykorzystaniu metody **prompt** i okna tekstowego (rys. 5.2) przypisać do zmiennej wartości podane przez użytkownika. Za pomocą instrukcji **document.write** wypisano wartość zmiennej **imie** (wprowadzonej przez użytkownika) w oknie przeglądarki (rys. 5.3). Jeżeli użytkownik nie wprowadzi imienia i wciśnie przycisk **Anuluj**, do zmiennej **imie** zostanie przypisana wartość **null**. Wówczas na ekranie przeglądarki pojawi się napis „Cześć null!".



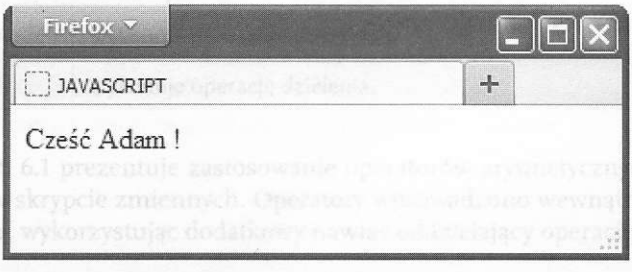
Rys. 5.1. Wynik działaniania skryptu z list. 5.1 – zmienne

Listing 5.2

```
<script type="text/javascript">
  var imie=prompt("Witaj! Jak masz na imię?","");
  document.write("Cześć "+imie+" !");
</script>
```



Rys. 5.2. Wynik działania skryptu z list. 5.2 – okno tekstowe



Rys. 5.3. Wynik działania skryptu z list. 5.2

6 Operatory

ZAGADNIENIA

- **Jakie operatory występują w języku JavaScript?**
- **Co to jest inkrementacja i dekrementacja?**
- **Co to jest konkatencja?**
- **Jak wykonywać operację przypisania?**
- **Jak stosować operatory arytmetyczne, logiczne, bitowe?**
- **Jak wykorzystywać w skrypcie inkrementację i dekrementację?**

W języku JavaScript wszystkie operacje na zmiennych dokonywane są za pomocą odpowiednich operatorów. Wśród nich można wyróżnić m.in. operatory arytmetyczne, logiczne, bitowe, operatory przypisania lub porównania.

Operatory arytmetyczne wykorzystywane są do wykonywania operacji matematycznych na zmiennych. Dostępne operatory przedstawiono w tabeli 6.1.

Tabela 6.1. Operatory arytmetyczne

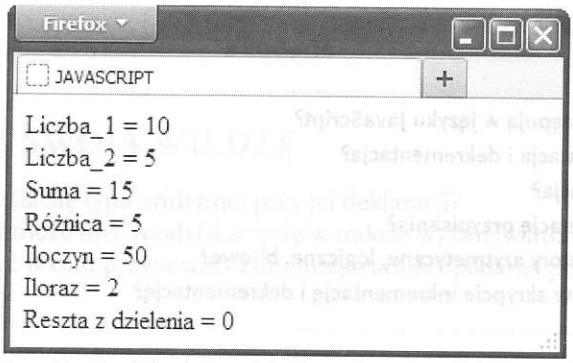
Symbol	Składnia	Opis
+	x+y	Wykonuje operację dodawania.
-	x-y	Wykonuje operację odejmowania.
-	-x	Wykonuje negację zmiennej.
%	x%y	Zwraca resztę z dzielenia pierwszej zmiennej przez drugą (dzielenie modulo).
*	x*y	Wykonuje operację mnożenia.
/	x/y	Wykonuje operację dzielenia.

Przykład z list. 6.1 prezentuje zastosowanie operatorów arytmetycznych na dwóch zadeklarowanych w skrypcie zmiennych. Operatory wprowadzono wewnątrz instrukcji **document.write**, wykorzystując dodatkowy nawias oddzielający operacje matematyczne.

Listing 6.1

```
<script type="text/javascript">
var liczba_1=10;
var liczba_2=5;
document.write("Liczba_1 = "+liczba_1+"<br>");
```

```
document.write("Liczba_2 = "+liczba_2+"<br>");
document.write("Suma = "+(liczba_1+liczba_2)+"<br>");
document.write("Różnica = "+(liczba_1-liczba_2)+"<br>");
document.write("Iloczyn = "+(liczba_1*liczba_2)+"<br>");
document.write("Iloraz = "+(liczba_1/liczba_2)+"<br>");
document.write("Reszta z dzielenia = "+(liczba_1%liczba_2)+"<br>");
</script>
```



Rys. 6.1. Wynik działania skryptu z list. 6.1

Operator łańcuchowy pozwala na złączenie dwóch ciągów znaków w jeden. W języku programowania takie połączenie nazywane jest **konkatencją**.

Tabela 6.2. Operator łańcuchowy (konkatencja)

Symbol	Składnia	Opis
+	"text1"+"text2"	Łączy dwa ciągi znaków w jeden.

Operatory bitowe związane są z wykonywaniem operacji na bitach. Na odpowiednich bitach zmiennych wykonywane są operacje algebry logicznej. Dostępne operatory przedstawiono w tabeli 6.3.

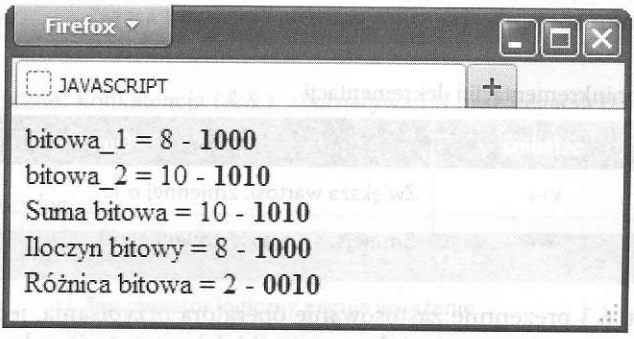
Tabela 6.3. Operatory bitowe

Symbol	Składnia	Opis
&	x&y	Wykonuje bitową operację AND, która wyświetla 1, jeśli obie zmienne wynoszą 1.
^	x^y	Wykonuje bitową operację XOR, która wyświetla 1, jeśli jedna ze zmiennych (ale nie obie jednocześnie) wynosi 1.
|	x|y	Wykonuje bitową operację OR, która wyświetla 1, jeśli jedna ze zmiennych wynosi 1.
<<	x<<y	Wykonuje przesunięcie bitów w lewo o podaną liczbę miejsc.
>>	x>>y	Wykonuje przesunięcie bitów w prawo o podaną liczbę miejsc.

Przykład z list. 6.2 prezentuje działanie sumy bitowej, iloczynu bitowego oraz różnicy bitowej. Zmienne zapisane są w postaci dziesiętnej, ale wykonywane na nich operacje przeprowadzane są na ich odpowiednikach w systemie dwójkowym. Wynik operacji również przedstawiony jest w systemie dziesiętnym. Dla ułatwienia wypisano również wszystkie wartości w systemie dwójkowym wytłuszczoną czcionką.

Listing 6.2

```
<script type="text/javascript">
var bitowa_1=8;
var bitowa_2=10;
document.write("bitowa_1 = "+bitowa_1+" - <b>1000</b><br>");
document.write("bitowa_2 = "+bitowa_2+" - <b>1010</b><br>");
document.write("Suma bitowa = "+(bitowa_1|bitowa_2)+" -
<b>1010</b><br>");
document.write("Iloczyn bitowy = "+(bitowa_1&bitowa_2)+" -
<b>1000</b><br>");
document.write("Różnica bitowa = "+(bitowa_1^bitowa_2)+" -
<b>0010</b><br>");
</script>
```



Rys. 6.2. Wynik działania skryptu z list. 6.2

Podstawowym **operatorem przypisania** jest znak =. Odpowiada on za przypisanie wartości argumentu prawostronnego argumentowi lewostronnemu. Argumentem prawostronnym może być zmienna lub wyrażenie, natomiast argument prawostronny stanowi zmienna, której zadaniem jest przyjęcie nowej wartości. JavaScript oferuje dodatkowo wiele operatorów łączonych zaprezentowanych w tabeli 6.4.

Tabela 6.4. Operatory przypisania

Symbol	Składnia	Opis
=	x=y	Przypisuje wartość y do zmiennej x.
+=	x+=y	Wykonuje przypisanie x=x+y.
-=	x-=y	Wykonuje przypisanie x=x-y.

Symbol	Składnia	Opis
*=	x*=y	Wykonuje przypisanie x=x*y.
/=	x/=y	Wykonuje przypisanie x=x/y.
%=	x%=y	Wykonuje przypisanie x=x%y.
<=	x<=y	Wykonuje przypisanie x=x<y.
^=	x^=y	Wykonuje przypisanie x=x^y.
|=	x|=y	Wykonuje przypisanie x=x|y.
<<=	x<<=y	Wykonuje przypisanie x=x<<y.
>>=	x>>=y	Wykonuje przypisanie x=x>>y.
>>>=	x>>>==y	Wykonuje przypisanie x=x>>>y.

Często stosowanym operatorem jest **inkrementacja** odpowiadająca zwiększeniu danej wartości o jeden. Zapisywana jest za pomocą dwóch plusów „++". Odwrotne działanie daje operator **dekrementacji** zmniejszający wartość danej zmiennej o jeden. Zapisywany jest za pomocą dwóch minusów „--".

Tabela 6.5. Operator inkrementacji i dekrementacji

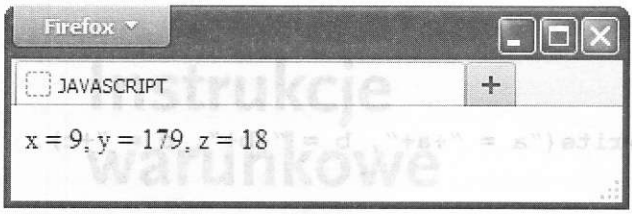
Symbol	Składnia	Opis
++	x++	Zwiększa wartość zmiennej o 1.
--	x--	Zmniejsza wartość zmiennej o 1.

Przykład z list. 6.3 prezentuje zastosowanie operatora przypisania, jednego z operatorów łączonych (*=) oraz operatorów inkrementacji i dekrementacji wobec trzech zadeklarowanych w skrypcie zmiennych.

Listing 6.3

```
<script type="text/javascript">
var x=8;
var y=10;
var z;
z=x+y;
x++;
y*=z;
y--;
document.write("x = "+x+", y = "+y+", z = "+z);
</script>
```

Operatory porównania wykorzystywane są do porównania dwóch argumentów. W wyniku podawana jest wartość **true**, jeżeli zależność jest prawdziwa, lub wartość **false**, jeżeli warunek nie został spełniony.



Rys. 6.3. Wynik działania skryptu z list. 6.3

Tabela 6.6. Operatory porównania

Symbol	Składnia	Opis
!=	x!=y	Zwraca **true**, jeśli zmienne nie są równe.
<	x<y	Zwraca **true**, jeśli pierwsza zmienna jest mniejsza niż druga.
<=	x<=y	Zwraca **true**, jeśli pierwsza zmienna jest mniejsza niż druga lub jej równa.
==	x==y	Zwraca **true**, jeśli zmienne są równe.
>	x>y	Zwraca **true**, jeśli pierwsza zmienna jest większa niż druga.
>=	x>=y	Zwraca **true**, jeśli pierwsza zmienna jest większa niż druga lub jej równa.

Operatory logiczne: konkatencja (&&), alternatywa (||) oraz negacja (!) zwracają wartość **true** (prawda) lub **false** (fałsz) według zależności przedstawionych w tabeli 6.7.

Tabela 6.7. Operatory logiczne

Symbol	Składnia	Opis
!	!x	Ten operator logiczny neguje wyrażenie.
&&	x&&y	Operator logiczny AND zwraca **true**, jeśli obie zmienne są prawdziwe (**true**).
||	x||y	Operator logiczny OR zwraca **true**, jeśli co najmniej jedna ze zmiennych jest prawdziwa (**true**).

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Przeanalizuj skrypt z list. 6.4 i podaj, jakie wartości przyjmują zmienne **a**, **b** i **c** w trakcie działania programu.

   Listing 6.4

```
<script type="text/javascript">
var a=7;
var b=10;
var c;
a--;
```

7 Instrukcje warunkowe

ZAGADNIENIA

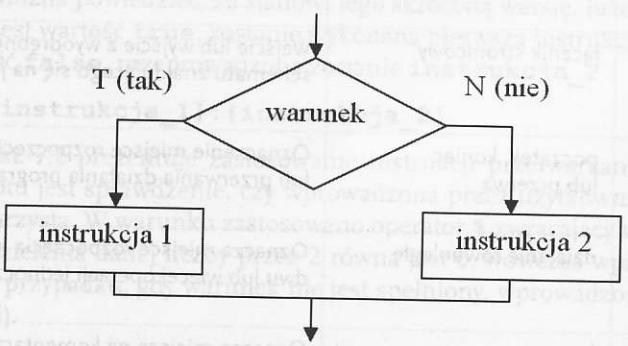
- **Jak wygląda ogólny zapis instrukcji warunkowej?**
- **Co to jest instrukcja przetwarzania warunkowego?**
- **Jak stosować instrukcję warunkową?**
- **Jak wykorzystywać operatory porównania w instrukcji warunkowej?**

Instrukcja warunkowa określa, który z fragmentów skryptu zostanie wykonany w zależności od spełnienia określonych warunków. Do wyboru mamy uproszczony zapis instrukcji warunkowej, dla której **instrukcje** zostaną wykonane w przypadku, gdy warunek przyjmie wartość **true**:

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

Bardziej rozbudowana forma instrukcji warunkowej ma dodatkowo element **else**. W tym przypadku **instrukcja_1** zostanie wykonana, gdy warunek przyjmie wartość **true**. W przeciwnym wypadku zostanie wykonana **instrukcja_2**, gdy warunek przyjmie wartość **false**.

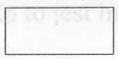
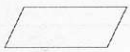
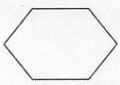
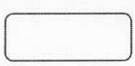
```
if (warunek) {
instrukcja_1;
}
else {
instrukcja_2;
}
```



Rys. 7.1. Schemat blokowy instrukcji warunkowej

Instrukcję warunkową można przedstawić również w postaci schematu blokowego (rys. 7.1). **Schemat blokowy** jest graficzną formą reprezentacji algorytmów. Algorytm stanowi zestaw instrukcji (zadań), których wykonanie prowadzi do osiągnięcia wyznaczonego celu. **Algorytm** przedstawiony w postaci schematu blokowego zbudowany jest z bloków zawierających takie informacje jak: dane wejściowe, wykonywane operacje oraz dane wyjściowe. Do stworzenia schematu blokowego wykorzystuje się odpowiednie bloki opisane w tabeli 7.1.

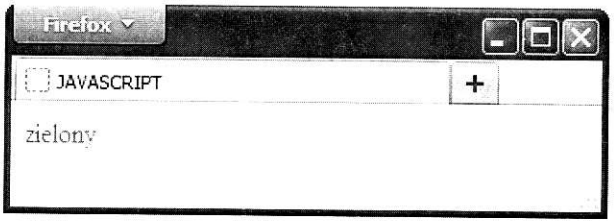
Tabela 7.1. Symbole graficzne w schematach blokowych programów według PN-75/E-01226

Symbol	Nazwa operacji	Wyjaśnienie
	przetwarzanie	Operacja lub grupa operacji, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych.
	wprowadzanie lub wyprowadzanie	Wprowadzanie lub wyprowadzanie danych.
	decyzja	Operacja określająca wybór jednej z alternatywnych dróg działania.
	przygotowanie	Modyfikacja rozkazu lub grupy rozkazów powodująca zmianę w przebiegu procesu przetwarzania.
	droga przepływu danych	Więź operacyjna pomiędzy poszczególnymi operacjami procesu przetwarzania.
	skrzyżowanie dróg przepływu danych bez powiązania między nimi	
	łączenie dróg przepływu danych	
	łącznik międzystronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdującego się na różnych stronach.
	łącznik stronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdującego się na jednej stronie.
	początek, koniec lub przerwa	Oznaczenie miejsce rozpoczęcia, zakończenia lub przerwania działania programu.
	działanie równoległe	Oznacza miejsce rozpoczęcia lub zakończenia dwu lub więcej operacji jednoczesnych.
	komentarz	Oznacza miejsce na komentarz.

Przykład z list. 7.1 prezentuje zastosowanie instrukcji warunkowej. Skrypt w pierwszej kolejności wyświetla okno tekstowe z informacją o wpisaniu koloru. Pierwszy warunek sprawdza, czy wprowadzony napis w oknie tekstowym to **zielony** lub 1. Jeżeli tak, na ekranie przeglądarki wyświetli się napis **zielony** w kolorze zielonym. W przeciwnym wypadku następuje przejście do sprawdzania drugiego warunku. Jeżeli wprowadzono tekst **czerwony** lub liczbę 2, na ekranie przeglądarki wyświetli się napis **czerwony** w kolorze czerwonym. W przypadku gdy żaden z warunków nie zostanie spełniony lub użytkownik wybierze przycisk **Anuluj**, na ekranie przeglądarki wyświetli się napis **"Podałeś nieprawidłową wartość!!"**.

Listing 7.1

```
<script type="text/javascript">
var kolor=prompt("Podaj kolor: 1-zielony lub 2-czerwony","");
if((kolor==1|kolor=="zielony")){
document.write("<font color=green>zielony</font>");
}
else if((kolor==2|kolor=="czerwony")){
document.write("<font color=red>czerwony</font>");
}
else{
document.write("Podałeś nieprawidłową wartość!!");
}
</script>
```



Rys. 7.2. Wynik działania skryptu z list. 7.1 po wpisaniu w oknie tekstowym wartości 1

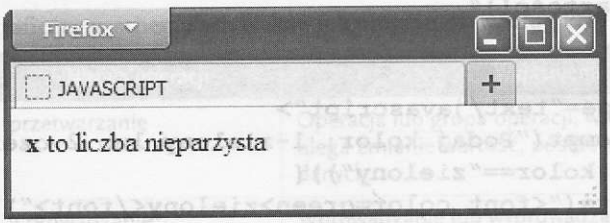
Instrukcja przetwarzania warunkowego pozwala na łatwe i szybkie zastąpienie bloku **if ... else**. Można powiedzieć, że stanowi jego skróconą wersję. Jeżeli wynikiem działania warunku jest wartość **true**, zostanie wykonana pierwsza instrukcja. Jeżeli warunek przyjmie wartość **false**, przeprowadzona zostanie **instrukcja_2**

```
(warunek)?{instrukcja_1}:{instrukcja_2}
```

Przykład z list. 7.2 prezentuje zastosowanie instrukcji przetwarzania warunkowego. Zadaniem skryptu jest sprawdzenie, czy wprowadzona przez użytkownika liczba jest parzysta, czy nieparzysta. W warunku zastosowano operator % zwracający resztę z dzielenia. Jeżeli reszta z dzielenia danej liczby przez 2 równa jest 0, wówczas wprowadzona liczba jest parzysta. W przypadku, gdy warunek nie jest spełniony, wprowadzona liczba jest nieparzysta (rys. 7.3).

Listing 7.2

```
<script type="text/javascript">
var x=prompt("Podaj liczbę","");
var x=(x%2==0)?"parzysta":"nieparzysta";
document.write("<b>x</b> to liczba "+x);
</script>
```



Rys. 7.3. Wynik działania skryptu z list. 7.2 po wpisaniu w oknie tekstowym wartości 15

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz skrypt sprawdzający, czy wprowadzona przez użytkownika wartość jest większa niż 100 i podzielna przez 3.
2. Opracuj skrypt, który wypisze 3 liczby wprowadzone przez użytkownika w kolejności malejącej.
3. Napisz skrypt, którego zadaniem jest pobranie od użytkownika informacji na temat jego wzrostu. Na podstawie wprowadzonych danych wypisz odpowiedni komunikat: NISKI < 150 CM < ŚREDNI < 180 CM < WYSOKI.

SPRAWDŹ SWOJĄ WIEDZĘ

1. Do czego wykorzystywana jest instrukcja warunkowa?
2. Podaj ogólny zapis instrukcji przetwarzania warunkowego.
3. W jakim celu stosuje się schematy blokowe?
4. Omów symbole graficzne wykorzystywane do budowy schematu blokowego

8 Pętle

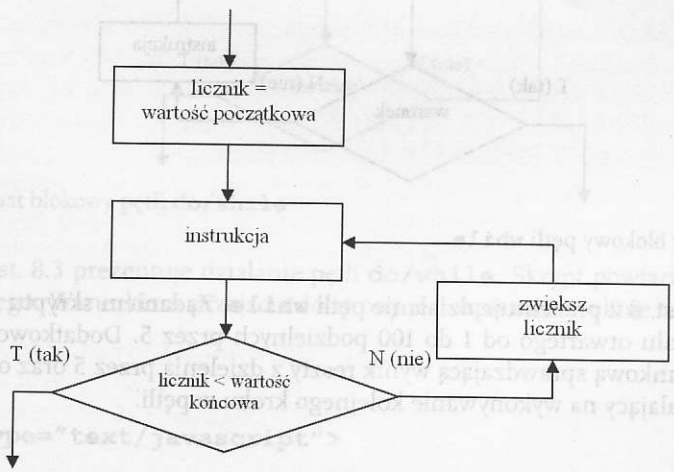
ZAGADNIENIA

- **Jakie zadanie mają pętle?**
- **Jakie pętle są dostępne w języku JavaScript?**
- **Jak stosować pętle w skrypcie?**
- **Jak stosować instrukcję warunkową wewnątrz pętli?**

Język JavaScript oferuje trzy rodzaje pętli (instrukcji iteracyjnych), których zadaniem jest powtarzanie instrukcji określoną liczbę razy. Pętla **for** jest najczęściej stosowanym rodzajem pętli. Składnia pętli jest następująca:

```
for (wyrażenie początkowe; warunek; wyrażenie modyfikujące){
instrukcje;
}
```

Wyrażenie początkowe odpowiada za zainicjowanie zmiennej używanej jako licznik przebiegu pętli. Spełnienie warunku umożliwia wykonanie kolejnego przejścia pętli. Wyrażenie modyfikujące dostosowuje zmienną będącą licznikiem pętli. Pętlę **for** można przedstawić również w postaci schematu blokowego (rys. 8.1).

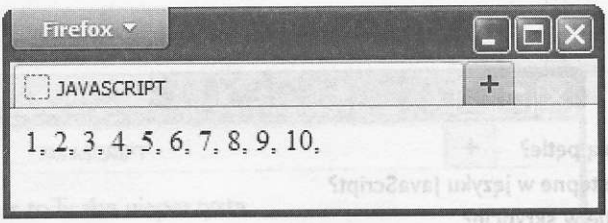


Rys. 8.1. Schemat blokowy pętli **for**

Przykład z list. 8.1 prezentuje wykorzystanie pętli **for**. Za jej pomocą dziesięciokrotnie wykonana została instrukcja **document.write**. Końcowym efektem jest wypisanie w oknie przeglądarki (rys. 8.2) cyfr od 1 do 10.

Listing 8.1

```
<script type="text/javascript">
for (x=1;x<=10;x++)
document.write(x+", ")
</script>
```

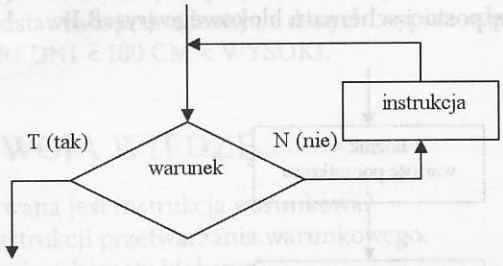


Rys. 8.2. Wynik działania skryptu z list. 8.1 – pętla **for**

Pętla **while** przed wykonaniem instrukcji sprawdza warunek logiczny. Jeżeli warunek przyjmuje wartość **true**, pętla będzie wykonywana do czasu osiągnięcia przez warunek wartości **false**. Może zdarzyć się sytuacja, że pętla nie wykona się ani razu, gdy za pierwszym razem warunek przyjmie wartość **false**. Składnia pętli jest następująca:

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

Pętlę **while** w postaci schematu blokowego prezentuje rys. 8.3.



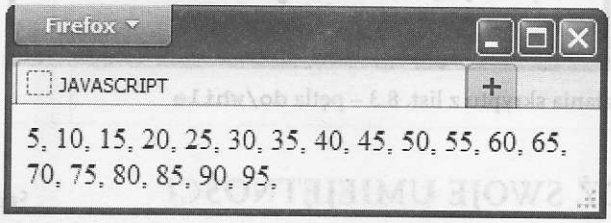
Rys. 8.3.Schemat blokowy pętli **while**

Przykład z list. 8.2 prezentuje działanie pętli **while**. Zadaniem skryptu jest wypisanie liczb z przedziału otwartego od 1 do 100 podzielnych przez 5. Dodatkowo zastosowano instrukcję warunkową sprawdzającą wynik reszty z dzielenia przez 5 oraz operator inkrementacji pozwalający na wykonywanie kolejnego kroku w pętli.

Listing 8.2

```
<script type="text/javascript">
var x=1;
while(x<100)
{
if(x%5==0)
```

```
document.write(x+", ");
x++;
}
</script>
```

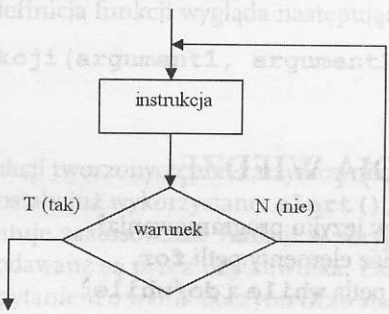


Rys. 8.4. Wynik działania skryptu z list. 8.2 – pętla **while**

Pętla **do/while** pozwala na wykonanie instrukcji przynajmniej raz, zanim zostanie sprawdzony warunek logiczny. Składnia pętli jest następująca:

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

Pętlę **do/while** w postaci schematu blokowego prezentuje rys. 8.5.



Rys. 8.5. Schemat blokowy pętli **do/while**

Przykład list. 8.3 prezentuje działanie pętli **do/while**. Skrypt powtarza wyświetlanie okna tekstowego. Warunkiem przejścia do strony jest wpisanie w oknie tekstowym hasła JavaScript.

Listing 8.3

```
<script type="text/javascript">
do{
var haslo=prompt("Podaj hasło", "")
}
while(haslo!="JavaScript")
document.write("Witaj na tajnej stronie!")
</script>
```

9 Funkcje

ZAGADNIENIA

- **Co to jest funkcja?**
- **Co to są argumenty funkcji?**
- **Jak definiować funkcję?**
- **Jak wywoływać funkcję?**

Funkcja to zamknięty fragment skryptu oznaczony odpowiednią nazwą, który można wywołać, wielokrotnie odwołując się do tej nazwy. Nazwa funkcji powinna zaczynać się od litery, a kolejne znaki nazwy mogą stanowić litery, cyfry lub znak podkreślenia (_). Funkcja może posiadać argumenty – jeden lub kilka oddzielonych przecinkami. Wywołując funkcję, należy o nich pamiętać. Lista argumentów może być pusta, wówczas pomiędzy nawiasami nic nie zostaje wpisane. W ciele funkcji można wprowadzić dowolną liczbę instrukcji. Należy tylko pamiętać, że instrukcje obowiązkowo należy umieścić w nawiasach klamrowych { }. Ogólnie definicja funkcji wygląda następująco:

```
function nazwa_funkcji(argument1, argument2, ... ,argument n){
instrukcje;
}
```

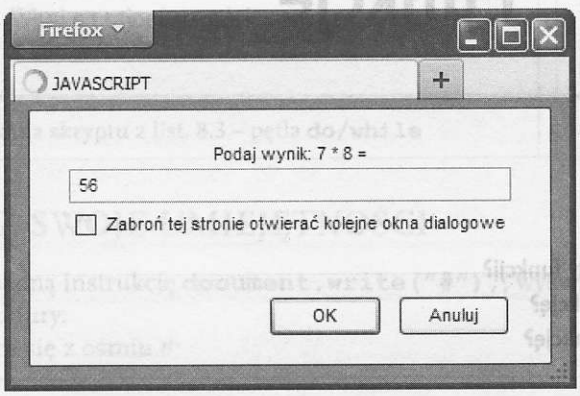
W JavaScript oprócz funkcji tworzonych przez użytkownika istnieją także funkcje wbudowane. Niektóre z nich zostały już wykorzystane: **alert()**, **confirm()**, **prompt()**.

Przykład list. 9.1 prezentuje zastosowanie funkcji w skrypcie. Funkcja ma dwa argumenty, których wartości podawane są przez użytkownika. Zadaniem funkcji jest wyświetlenie okna dialogowego z pytaniem o wynik iloczynu liczb stanowiących argumenty funkcji. Jeżeli wprowadzony wynik jest poprawny, na ekranie przeglądarki pojawi się napis: **Brawo! Wynik poprawny**. W przypadku wprowadzenia błędnej odpowiedzi, wypisana zostanie informacja o błędzie i, dodatkowo, odpowiedź poprawna.

Listing 9.1

```
<script type="text/javascript">
function iloczyn(a,b){
var x=prompt("Podaj wynik: "+a+" * "+b+" ","");
if(x==a*b)
document.write("Brawo! Wynik poprawny.");
else
document.write("Błąd! "+a+" * "+b+" = "+(a*b));
}
x=prompt("Podaj pierwszą liczbę: ");
```

```
y=prompt("Podaj drugą liczbę: ");
iloczyn(x,y);
</script>
```



Rys. 9.1. Wynik działania skryptu z list. 9.1 – funkcja

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

1. Napisz funkcję pobierającą dwa argumenty typu całkowitego x i y (x<y) oraz zwracającą wartość sumy wszystkich elementów przedziału otwartego (x,y).

SPRAWDŹ SWOJĄ WIEDZĘ

1. Podaj przykład funkcji wbudowanej.
2. Jak wywołać funkcję z argumentem? Podaj przykład.
3. W jakim celu stosujemy funkcje?

10 Obiekty

ZAGADNIENIA

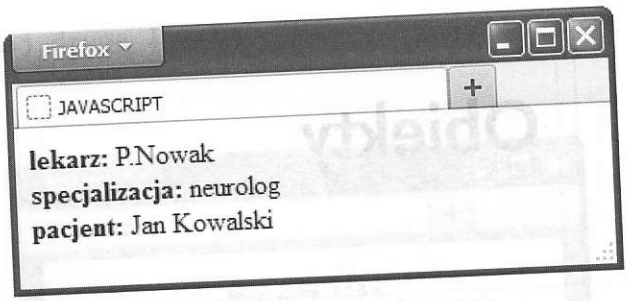
- **Co to jest obiekt?**
- **Za co odpowiada konstruktor?**
- **Jak tworzyć własne obiekty?**
- **Jak korzystać z obiektów wbudowanych?**

Obiekt to konstrukcja programistyczna mająca swoje cechy charakterystyczne (właściwości), którymi mogą być zmienne lub inne obiekty. Dodatkowo obiekt ma możliwość wykonywania różnych funkcji, które nazywamy metodami. Obiekt definiujemy, tworząc specjalną funkcję zwaną **konstruktorem**, a następnie tworzymy, stosując operator **new**.

Najłatwiej przedstawić to na przykładzie list. 10.1. W skrypcie zdefiniowano za pomocą konstruktora obiekt **osoba** zawierający właściwości: **imie** i **nazwisko** oraz obiekt **szpital** zawierający właściwości: **lekarz**, **specjalizacja** oraz **pacjent** (obiekt). Następnie utworzono oba obiekty za pomocą operatora **new**. Ostatnim krokiem jest wyświetlenie informacji poprzez odwołanie się do konkretnej właściwości obiektu (**nazwa_obiektu.nazwa_właściwości**).

Listing 10.1

```
<script type="text/javascript">
function szpital(lekarz,specjalizacja,pacjent) {
this.lekarz=lekarz;
this.specjalizacja=specjalizacja;
this.pacjent=pacjent;
}
function osoba(imie,nazwisko) {
this.nazwisko=nazwisko;
this.imie=imie;
}
pacjent=new osoba("Jan","Kowalski");
oddzial=new szpital("P.Nowak","neurolog",pacjent);
document.write("<b>lekarz:</b> "+oddzial.lekarz+"<br>");
document.write("<b>specjalizacja:</b> "+oddzial.specjalizacja-
+"<br>");
document.write("<b>pacjent:</b> "+ oddzial.pacjent.imie +"
"+oddzial.pacjent.nazwisko);
</script>
```



Rys. 10.1. Wynik działania skryptu z list. 10.1 – obiekty własne

Oprócz możliwości tworzenia własnych obiektów JavaScrip udostępnia spory zbiór obiektów własnych (wbudowanych), posiadających własne właściwości i metody.

Obiekt window reprezentuje okno przeglądarki i stoi na szczycie hierarchii obiektów. Jest to obiekt domyślny, co oznacza, że do większości jego metod i właściwości można odwołać się bezpośrednio, pomijając jego nazwę. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.1 oraz 10.2.

Tabela 10.1. Właściwości obiektu window

Właściwości	Opis
`frames[]`	Macierz ramek potomnych w oknie.
`frames.length`	Liczba zdefiniowanych ramek.
`self`	Bieżące okno.
`parent`	Okno rodzicielskie ramki potomnej w zestawie zdefiniowanym znacznikiem `<frameset>`.
`top`	Okno najwyższego rzędu, które jest właścicielem wszystkich widocznych ramek.
`status`	Komunikat pojawiający się w pasku stanu okna przeglądarki.
`defaultStatus`	Komunikat pojawiający się w pasku stanu okna przeglądarki standardowo, kiedy oczekuje ona na wprowadzenie jakichś danych przez użytkownika.
`name`	Wewnętrzny identyfikator okna otwartego metodą `window.open()` (może być niezdefiniowany).

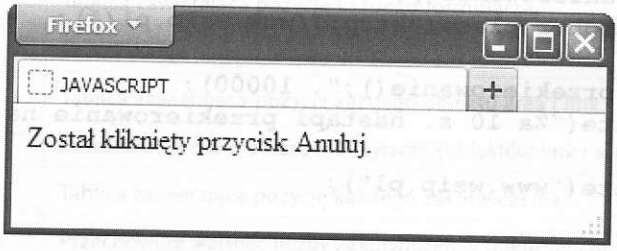
Tabela 10.2. Metody obiektu window

Metody	Opis
`alert(„komunikat")`	Wyświetla okno dialogowe.
`confirm(„komunikat")`	Wyświetla okno decyzyjne.
`prompt(„komunikat")`	Wyświetla okno tekstowe.
`open(„URL", „nazwa")`	Otwiera na ekranie nowe okno, nadaje mu wewnętrzny identyfikator „`nazwa`" i ściąga do niego dokument wskazany lokalizatorem „`URL`".
`close()`	Zamyka okno z dokumentem.

Przykład z list. 10.2 prezentuje działanie przycisków okna decyzyjnego. Instrukcja warunkowa sprawdza, jaka wartość logiczna została przypisana do zmiennej **test** po wciśnięciu wybranego przycisku w oknie decyzyjnym. Jeżeli zostanie wciśnięty przycisk **OK**, **test** przyjmie wartość **true** i w oknie przeglądarki wyświetli się informacja **Został kliknięty przycisk OK**. Gdy zostanie wciśnięty przycisk **Anuluj**, **test** przyjmie wartość **false** i w oknie przeglądarki pojawi się napis **Został kliknięty przycisk Anuluj** (rys. 10.2).

Listing 10.2

```
<script type="text/javascript">
var test=confirm("Test okienka decyzyjnego:");
if(test==true){
  document.write("Został kliknięty przycisk OK.");
}
else{
  document.write("Został kliknięty przycisk Anuluj.");
}
</script>
```



Rys. 10.2. Wynik działania skryptu z list. 10.2 – wciśnięcie **Anuluj** w oknie decyzyjnym

Obiekt location posiada informacje dotyczące aktualnego adresu URL dokumentu oraz metody pozwalające na operowanie tym adresem. Najpopularniejsze właściwości i metody przedstawiają odpowiednio tabela 10.3 i tabela 10.4.

Tabela 10.3. Właściwości obiektu location

Właściwości	Opis
href	Łańcuch zawierający cały adres URL dokumentu.
protocol	Łańcuch zawierający początek adresu URL wraz z pierwszym dwukropkiem.
host	Łańcuch zawierający nazwę serwera, nazwę domeny.
hostname	Łańcuch zawierający pełną nazwę serwera łącznie z numerem portu.
port	Łańcuch określający używany przez serwer port komunikacyjny.
pathname	Łańcuch zawierający część adresu URL.
hash	Łańcuch rozpoczynający się od znaku #, który określa nazwę zakotwiczenia w dokumencie.
search	Łańcuch rozpoczynający się znakiem ?, który określa zapytanie w adresie URL.

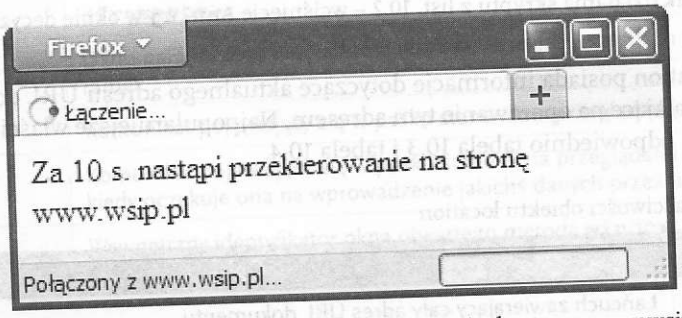
Tabela 10.4. Metody obiektu location

Metody	Opis
`assign(url)`	Wczytuje dokument o adresie wskazanym przez argument url.
`reload(force)`	Wymusza ponowne wczytanie bieżącej strony.
`replace(url)`	Zastępuje bieżący dokument przez wczytany spod adresu wskazanego przez URL.

Przykład z list 10.3 prezentuje metodę **replace** obiektu **location**. Skrypt po upływie 10 sekund powoduje przekierowanie do strony www.wsip.pl. Do opóźnienia czasowego zastosowano funkcję **setTimeout**, która przyjmuje dwa parametry: nazwę funkcji (**przekierowanie()**), która ma zostać wykonana po upływie określonego czasu, oraz czas zwłoki (**10000**) – okres opóźnienia wykonania funkcji podawany w milisekundach.

Listing 10.3

```
<script type="text/javascript">
function przekierowanie(){
window.location.replace("http://www.wsip.pl/");
}
setTimeout("przekierowanie();", 10000);
document.write("Za 10 s. nastąpi przekierowanie na stronę
<br>");
document.write("www.wsip.pl");
</script>
```



Rys. 10.3. Wynik działania skryptu z list. 10.3 – przekierowanie do strony www.wsip.pl

Obiekt document wykorzystuje dostępne metody i właściwości do modyfikacji dokumentu HTML aktualnie wczytanego przez przeglądarkę. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.5 oraz 10.6.

Przykład z list. 10.4 przedstawia trzy informacje dotyczące skryptu: tytuł dokumentu, datę ostatniej modyfikacji oraz aktualny adres URL. Do wyświetlenia informacji na stronie zastosowano metodę **write**, jedną z najczęściej stosowanych metod obiektu **document**.

Tabela 10.5. Właściwości obiektu **document**

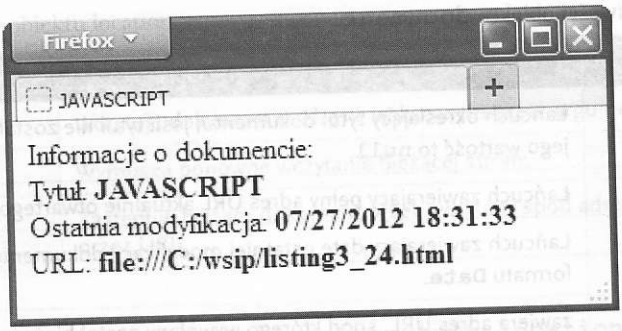
Właściwości	Opis
`title`	Łańcuch określający tytuł dokumentu; jeśli tytuł nie został zdefiniowany, jego wartość to `null`.
`location`	Łańcuch zawierający pełny adres URL aktualnie otwartego dokumentu.
`lastModified`	Łańcuch zawierający datę ostatniej modyfikacji dokumentu; jest on formatu `Date`.
`referrer`	zawiera adres URL, spod którego wywołany został bieżący dokument.
`bgColor`	Łańcuch określający kolor tła dokumentu.
`fgColor`	Łańcuch określający kolor tekstu w dokumencie.
`linkColor`	Łańcuch określający kolor odsyłaczy hipertekstowych w dokumencie.
`vlinkColor`	Łańcuch określający kolor odwiedzonych odsyłaczy hipertekstowych.
`alinkColor`	Łańcuch określający kolor aktywnego odsyłacza hipertekstowego.
`forms[]`	Tablica zawierająca pozycję każdego formularza.
`forms.length`	Przechowuje dane o liczbie formularzy w dokumencie.
`links[]`	Tablica zawierająca pozycję każdego obiektu area i link.
`links.length`	Przechowuje dane o liczbie odsyłaczy (obiektów link i area) w dokumencie.
`anchors[]`	Tablica zawierająca pozycję każdego zakotwiczenia.
`anchors.length`	Przechowuje wartość liczby zakotwiczeń w dokumencie.

Tabela 10.6. Metody obiektu document

Metody	Opis
`write („łańcuch")`	Wypisuje wyrażenie HTML w dokumencie w bieżącym oknie.
`clear()`	Czyści zawartość bieżącego okna.
`close()`	Powoduje zamknięcie bieżącego okna.

Listing 10.4

```
<script type="text/javascript">
document.write("Informacje o dokumencie: <br>");
document.write("Tytuł: <b>"+document.title+"</b><br>");
document.write("Ostatnia modyfikacja: <b>"+document.lastModified+"
</b><br>");
document.write("URL: <b>"+document.location+"</b><br>");
</script>
```



Rys. 10.4. Wynik działania skryptu z list. 10.4 – informacje o dokumencie

Obiekt string stanowi każdy ciąg znaków ujęty w znakach cudzysłowu lub apostrofu. Najpopularniejsze właściwości i metody przedstawiono w tabelach 10.7 i 10.8.

Tabela 10.7. Właściwości obiektu string

Właściwości	Opis
`length`	Zwraca wartość liczbową charakteryzującą liczbę znaków w łańcuchu.

Tabela 10.8. Metody obiektu string

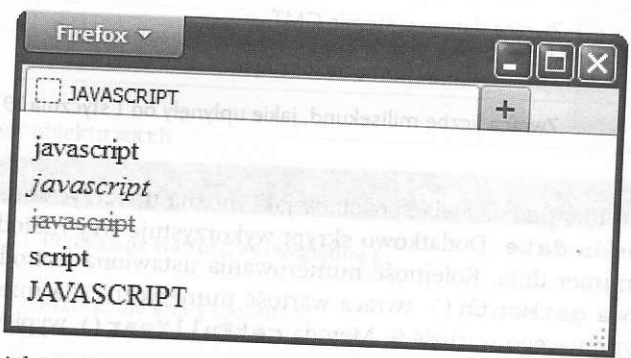
Metody	Opis
`big()`	Zwiększa rozmiar czcionki; odpowiednik znacznika `<big>`.
`blink()`	Tekst migający; odpowiednik znacznika `<blink>`.
`bold()`	Tekst pogrubiony; odpowiednik znacznika `<b>`.
`fixed()`	Odpowiednik znacznika `<tt>`.
`italics()`	Tekst pochylony; odpowiednik znacznika `<i>`.
`small()`	Zmniejsza rozmiar czcionki; odpowiednik znacznika `<small>`.
`sub()`	Odpowiednik znacznika `<sub>`.
`strike()`	Tekst przekreślony; odpowiednik znacznika `<strike>`.
`sup()`	Odpowiednik znacznika `<sup>`.
`fontColor(kolor)`	Ustawia kolor czcionki (tekstu) na wartość `kolor`.
`fontSize(rozmiar)`	Ustawia rozmiar czcionki na wartość `rozmiar`.
`charAt(indeks)`	Zwraca znak z pozycji określonej przez indeks.
`indexOf(podłańcuch [,indeks])`	Przeszukuje łańcuch w poszukiwaniu podłańcucha i zwraca pozycję pierwszego znalezionego znaku.

Metody	Opis
`lastindexOf(podłańcuch [,indeks])`	Przeszukuje łańcuch w poszukiwaniu podłańcucha w kierunku przeciwnym, czyli od końca.
`substring(x, y)`	Zwraca podłańcuch wycięty z łańcucha od pozycji x do pozycji y.
`toLowerCase()`	Konwertuje znaki w łańcuchu na małe litery.
`toUpperCase()`	Konwertuje znaki w łańcuchu na wielkie litery.

Przykład z list. 10.5 przedstawia zastosowanie wybranych metod obiektu **string**. W skrypcie zadeklarowano zmienną **tekst** typu łańcuchowego. Poddano ją formatowaniu z wykorzystaniem jednej lub kilku metod jednocześnie.

Listing 10.5

```
<script type="text/javascript">
var tekst="javascript";
document.write(tekst+"<br>");
document.write(tekst.italics()+"<br>");
document.write(tekst.strike().fontcolor("red")+"<br>");
document.write(tekst.substring(4,10)+"<br>");
document.write(tekst.toUpperCase()+"<br>");
</script>
```



Rys. 10.5. Wynik działania skryptu z list. 10.5 – przykładowe metody obiektu **string**

Obiekt `date` pozwala na wykonywanie operacji z wykorzystaniem daty i czasu. Pozwala na uzyskanie aktualnej wartości daty i czasu, na korzystanie z ich składowych oraz niezależną zmianę każdej z nich. Praca z obiektem **date** uzależniona jest od użycia konstruktora. Może to być konstruktor bezparametrowy:

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

lub konstruktor mający od jednego do siedmiu parametrów (rok, miesiąc, dzień, godzina, minuty, sekundy, milisekundy).

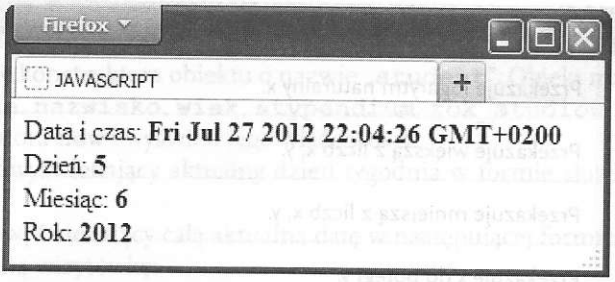
Tabela 10.9. Metody obiektu **date**

Metody	Opis
`getDay()`	Zwraca dzień tygodnia.
`getDate()`	Zwraca dzień miesiąca.
`getHours()`	Zwraca wartość reprezentującą godzinę.
`getMinutes()`	Zwraca wartość reprezentującą minuty.
`getMonth()`	Zwraca wartość reprezentującą numer miesiąca.
`getSeconds()`	Zwraca wartość reprezentującą sekundy.
`getTime()`	Zwraca wartość numeryczną określającą czas; wartość w milisekundach.
`getYear()`	Zwraca rok.
`setDate()`	Ustawia dzień miesiąca.
`setHour()`	Ustawia godzinę.
`setMinutes()`	Ustawia minutę.
`setMonth()`	Ustawia miesiąc.
`setSeconds()`	Ustawia sekundy.
`setTime()`	Ustawia wartość obiektu **Date** – wartość w milisekundach.
`setYear()`	Ustawia rok.
`toGMTString()`	Zwraca datę w systemie GMT.
`toLocaleString()`	Zwraca datę w formacie lokalnym.
`parse(date)`	Zwraca liczbę milisekund, jakie upłynęły od 1 stycznia 1970.

Przykład z list. 10.6 przedstawia sposób, w jaki można utworzyć konstruktor bezparametrowy dla obiektu **date**. Dodatkowo skrypt wykorzystuje trzy metody. Metoda **getDay()** zwraca numer dnia. Kolejność numerowania ustawiona jest od 0 (niedziela) do 6 (sobota). Metoda **getMonth()** zwraca wartość numeryczną dla miesiąca, począwszy od stycznia przyjmującego wartość 0. Metoda **getFullYear()** wypisuje rok w zapisie czterocyfrowym.

Listing 10.6

```
<script type="text/javascript">
var data_czas = new Date();
document.write ("Data i czas: <b>"+data_czas+"</b><br>");
document.write ("Dzień: <b>"+data_czas.getDay()+"</b><br>");
document.write ("Miesiąc: <b>"+data_czas.getMonth()+"</
b><br>");
document.write ("Rok: <b>"+data_czas.getFullYear()+"</b><br>");
</script>
```



Rys. 10.6. Wynik działania skryptu z list. 10.6 – przykładowe metody obiektu **date**

Obiekt **math** wykorzystywany jest do wykonywania różnych obliczeń matematycznych. Udostępnia również szereg stałych matematycznych (tab. 10.10) oraz dodatkowe metody (tab. 10.11).

Tabela 10.10. Właściwości obiektu **math**

Właściwości	Opis
LN10	Stała wartość; logarytm naturalny z 10 = 2.302585...
LN2	Stała wartość; logarytm naturalny z 2 = 0.693147...
PI	Stała wartość; liczba pi = 3.141592...
SQRT1_2	Stała wartość; pierwiastek kwadratowy z ½ = 0.707107...
SQRT2	Stała wartość; pierwiastek kwadratowy z 2 = 1.414213...

Tabela 10.11. Metody obiektu **math**

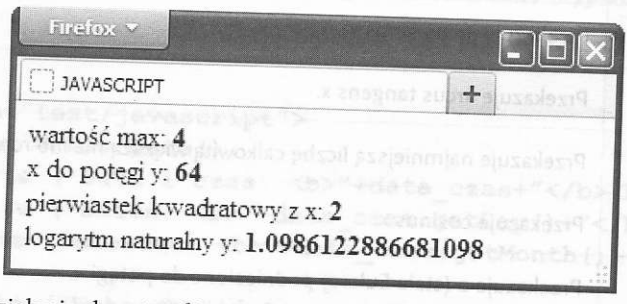
Metody	Opis
abs(x)	Przekazuje wartość bezwzględną x.
acos(x)	Przekazuje arcus cosinus x.
asin(x)	Przekazuje arcus sinus x.
atan(x)	Przekazuje arcus tangens x.
ceil(x)	Przekazuje najmniejszą liczbę całkowitą większą niż lub równą x.
cos(x)	Przekazuje cosinus x.
exp(x)	Przekazuje e (stała Eulera) podniesione do potęgi x.
floor(x)	Przekazuje największą liczbę całkowitą mniejszą niż lub równą x.

Metody	Opis
`log(x)`	Przekazuje logarytm naturalny x.
`max(x, y)`	Przekazuje większą z liczb x, y.
`min(x, y)`	Przekazuje mniejszą z liczb x, y.
`pow(x, y)`	Przekazuje x do potęgi y.
`round(x)`	Przekazuje x zaokrąglone do najbliższej liczby całkowitej.
`sin(x)`	Przekazuje sinus x.
`sqrt(x)`	Przekazuje pierwiastek kwadratowy z x.
`tan(x)`	Przekazuje tangens x.

Przykład z list. 10.6 przedstawia działanie czterech metod obiektu **math**. Skrypt pobiera za pomocą okna tekstowego wartości zmiennych **x** i **y**. Następnie za pomocą metody **max(x,y)** określa, która ze zmiennych ma większą wartość. Kolejna metoda **pow(x,y)** podnosi wartość zmiennej **x** do potęgi o wartości zmiennej **y**. Metoda **sqrt(x)** wyciąga pierwiastek kwadratowy ze zmiennej **x**. Metoda **log(y)** podaje wartość logarytmu naturalnego liczby **y**.

Listing 10.7

```
<script type="text/javascript">
var x=prompt("Podaj x:","");
var y=prompt("Podaj y:","");
document.write("wartość max: <b>"+Math.max(x,y)+"</b><BR>");
document.write ("x do potęgi y: <b>"+Math.pow(x,y)+"</b><BR>");
document.write ("pierwiastek kwadratowy z x: <b>"+Math.sqrt(x)+"</b><BR>");
document.write ("logarytm naturalny y: <b>"+Math.log(y)+"</b><BR>");
</script>
```



Rys. 10.7. Wynik działania skryptu z list. 10.7 dla x = 4 i y = 3