JAVAScript w dokumentach HTML (2)

Interakcyjne wprowadzanie danych

Jednym ze sposobów jest stosowanie metody **prompt** dla wbudowanego obiektu **window**:

```
zmienna= prompt("Tekst zachęty, np. Wprowadź x:");
```

Instrukcja powoduje otwarcie okienka z polem edycyjnym do wpisania tekstu. Po zatwierdzeniu tekst ten przechowywany jest w zmiennej.

Uwaga: Jeśli chcemy wprowadzać w okienku **prompt** tekst w postaci liczby i potem wykonywać na niej działania arytmetyczne (szczególnie dodawanie), należy dokonać konwersji tekstu na liczbę funkcją ***Number***:

Zadanie

1. Sprawdzić powyższy ciąg instrukcji:

```
x= prompt("Wprowadź x: ");
y= prompt("Wprowadź y: ");
sumaT=x+y;
alert (sumaT)
sumaL=Number(x)+Number(y);
alert (sumaL)
```

   Zrozumieć efekty. Przetestować, czy konwersja potrzebna jest dla mnożenia dwóch liczb podanych jako teksty w okienku **prompt** lub obliczania dowolnej funkcji matematycznej, np. Math.sqrt(x).

2. Utworzyć kod obliczający deltę i pierwiastki równania kwadratowego dla wprowadzanych przez użytkownika współczynników a, b, c (wyjaśnić działanie w przypadku ujemnej wartości delta).

Funkcje definiowane przez użytkownika

Funkcje własne użytkownika są definiowane między znacznikami <SCRIPT> </SCRIPT> w sekcji HEAD. Pozwala to na załadowanie ich na samym początku, aby dowolny skrypt na stronie mógł je wykorzystywać. Funkcje są definiowane przez określenie ich nazwy, argumentów i sposobu działania.

Przykład

Definiujemy funkcję o nazwie **poleTrojkata**, jej argumentami są długość podstawy i wysokość trójkąta, funkcja ma zwrócić pole trójkąta jako wynik do miejsca wywołania:

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function poleTrojkata(a, h){
pole=a*h/2;
return pole
}
</SCRIPT>
</HEAD>
```

W nawiasie po nazwie funkcji umieszczamy listę argumentów (danych) funkcji. Działania funkcji (operacje na argumentach) są zdefiniowane między nawiasami { }. Po słowie kluczowym *return* określamy którą wartość ma zwrócić funkcja.

Aby wywołać funkcję, trzeba umieścić **wykonanie** funkcji w tekście skryptu, w ramach znaczników <SCRIPT> i </SCRIPT>.

Skrypt z wykorzystaną funkcją:

```
<BODY>
<SCRIPT language="Javascript">
document.write ("Pole trójkąta 1=", poleTrojkata( 4,  6.7), "<BR />");
```

```
x= prompt ("Podaj długość podstawy trójkąta:")
y= prompt ("Podaj wysokość trójkąta:")
document.write ("Pole trójkąta 2=", poleTrojkata( x, y), "<BR />");
</SCRIPT>
</BODY>
```

Funkcja ma zwrócić wartość do miejsca wywołania (*return*) – wartość funkcji wyświetlamy z wykorzystaniem *document*.*write*.

Można wykonać funkcję na rozkaz użytkownika, umieszczając ją w atrybucie *onclick* dowolnego elementu (znacznika):

Przykład

Sprawdzić działanie skryptu wykorzystującego własną funkcję *nacisnij,* wykonywaną bez argumentów .

alert jest metodą dla obiektu *window*, tworzącą okienko dialogowe z napisem informacyjnym. Znacznik INPUT typu *button* reprezentuje przycisk.

```
<HTML> <HEAD>
<SCRIPT language="JavaScript"> //definicja funkcji
  function nacisnij( ) //funkcja bezargumentowa
    { alert("Witaj!"); }
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<P>Wykorzystanie funkcji</P>
<INPUT type="button"  value="OK"   onclick="nacisnij( );">
</FORM>
</BODY></HTML>
```

Zwróćmy uwagę na przypisanie wykonania funkcji do atrybutu *onclick* obiektu INPUT. Nasza funkcja nie posiada argumentów.

Zadanie

1. Wypróbować powyższy przykład oraz wzbogacić definicję funkcji dowolny argument (tekstowy, numeryczny) przekazywany do funkcji i stworzyć dwa przyciski wykorzystujące funkcję z różnymi wartościami argumentów. Metoda **alert** powinna wyświetlać wartość przekazanego argumentu.
2. Utworzyć i wykorzystać własne funkcje do:
   a. Losowania liczby całkowitej z przedziału (-50, 50),
   b. Obliczania objętości kuli dla podanego promienia.
3. Utworzyć funkcję obliczającą pierwiastki równania kwadratowego (argumentami są 3 współczynniki) i wykorzystać tę funkcję do obliczeń.

Pobieranie danych z pól edycyjnych i operacje na ich wartościach

Przykład pokazuje możliwość wykonania funkcji pobierającej daną z pola edycyjnego w powiązaniu ze zdarzeniem kliknięcia przycisku i wyświetlenie wartości w oknie *alert*:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function pobierz(s) {
  alert("Cześć  "+ s+"!");
 }
</SCRIPT>
</HEAD><BODY>
Wpisz swoje imię:
 <INPUT type="text" id="imie">
```

```
 <INPUT type="button"  value="Naciśnij" onclick="pobierz(imie.value)">
</BODY>
</HTML>
```

Polu tekstowemu nadaliśmy identyfikator (atrybut *id*) o nazwie **dane1**. W atrybucie *onclick* przycisku każemy wykonać funkcję z argumentem **dane1.value** (*value* oznacza wartość wpisaną w polu tekstowym). Funkcja nie posiada frazy **return**, zatem nie zwraca wartości, jedyną jej akcją jest pokazanie okienka alertu z podawanym przez argument tekstem.

Wartość atrybutu *onclick* może zawierać kilka instrukcji *JavaScript* (oddzielanych średnikami).

Zadania

1. Wykonać test powyższego kodu *JavaScript*.
2. Wstawić na stronie formularz z dwoma polami edycyjnymi do wpisania dwóch liczb. Po kliknięciu przycisku w trzecim polu edycyjnym pojawić się ma iloczyn tych liczb. W akcji kliknięcia wykorzystać schemat:

   pole3.value=pole1.value*pole2.value

   Sprawdzić sytuację błędnego wpisywania liczb. Jak prawidłowo wpisywać liczby dziesiętne w polu INPUT, z kropką czy przecinkiem?

   Dla operacji sumowania należy wykorzystać standardową funkcję *Number.*

   Można również utworzyć i wykorzystać własną funkcję *sumuj* (a , b).
3. W trzech polach INPUT podajemy trzy liczby i przekazujemy jako argumenty funkcji obliczającej pierwiastki równania kwadratowego. Wyniki po kliknięciu przycisku pojawiają się w okienku alert.

Instrukcja warunkowa

Stosowana do podejmowania decyzji – alternatywne operacje w zależności od wartości sprawdzanego warunku(-ów).

Postać instrukcji warunkowej ***if***

```
if (warunek) {
instrukcje wykonywane jeżeli warunek spełniony
}
else  {
instrukcje wykonywane jeżeli warunek niespełniony
}
```

Warunkiem jest poprawne wyrażenie logiczne o wartości *true* lub *false*.

Można zagnieżdżać instrukcję **if** w innej instrukcji **if** celem sprawdzenia kilku warunków.

```
<SCRIPT LANGUAGE="JavaScript">
liczba= Math.log(3);//można też podawać liczbę w okienku prompt
if ((liczba>0)&&(liczba<1))
        {
        document.write("liczba w przedziale (0, 1)");
        }
        else
        {
        if ((liczba>=1)&&(liczba<2))
                document.write("liczba w przedziale [1,  2)");
            else
                document.write("liczba>2");
        }
</SCRIPT>
```

Sprawdzić i zrozumieć działanie powyższego skryptu.

Zadanie

Wzbogacić funkcję obliczającą pierwiastki równania kwadratowego z poprzedniego zadania o analizę wartości *delta* i wypisać na ekranie komunikat czy jest ona ujemna, równa zero czy dodatnia (wykorzystać instrukcję **if**).

Instrukcja iteracyjna ("pętla") for

Umożliwia wielokrotne wykonanie bloku instrukcji – liczba powtórzeń uzależniona od zmian wartości zmiennej będącej licznikiem

Schemat uproszczonej postaci instrukcji:

```
for (licznik=wartosc_pocz;  warunek kontynuacji;   sposób zmiany licznika)
        {instrukcje powtarzane}
```

Nawiasy klamrowe mogą zostać pominięte jeśli istnieje jedna instrukcja wykonawcza "pętli".

Sprawdzić i zrozumieć działanie skryptu:

```
<SCRIPT LANGUAGE="JavaScript">
    for (i=0; i<20; i++)
        document.write(i+"<BR />");
</SCRIPT>
```

Zagnieżdżanie pętli **for**:

```
<SCRIPT LANGUAGE="JavaScript">
    for (i=1; i<6; i++)
    {
        for( k=1; k<4; k++)
            {
            document.write("i=",i," k=",k,"<BR />");
            document.write("Iloczyn  i*k:",(i*k)+,"<BR />");
            }
    }
</SCRIPT>
```

Zadanie

Sprawdzić i zrozumieć działanie powyższych skryptów.

Instrukcja pętli warunkowej – while (dopóki..)

Umożliwia wielokrotne wykonanie bloku instrukcji – liczba powtórzeń jest uzależniona od wartości logicznej warunku, dopóki warunek jest spełniony (*true*) instrukcje są powtarzane. Warunek sprawdzany jest na początku. Instrukcje powinny mieć wpływ na wartość logiczną warunku (kiedyś warunek musi osiągnąć wartość *false*, inaczej będzie nieskończona liczba powtórzeń i przeglądarka się "zawiesi").

Postać instrukcji:

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

Sprawdzić i zrozumieć działanie skryptu:

```
<SCRIPT LANGUAGE="JavaScript">
    x=0;
    while ( Math.sin(x*Math.PI/180)<0.3 )
    {
        document.write(x, "---------", Math.sin(x*Math.PI/180), "<BR />");
        x+=2;
    }
</SCRIPT>
```

Zadania do wykonania

1. Wykonać pętlę, w której na stronie wypisywane są liczby parzyste od 0 do 100.

2. Wykonać pętlę, która utworzy przy pomocy znaczników HTML obramowaną tabelę o 1 wierszu i 20 komórkach w wierszu. W komórkach umieścić liczby naturalne od 1 do 20.
3. Wykonać pętlę wypisującą wszystkie wartości funkcji ***sin***(x) dla kątów od 0 do 90^0 z krokiem równym 1^0 , spełniające warunek $\sin(x) \in (0.3, 0.6)$. *(Uwaga: we wnętrzu pętli umieścić odpowiednią instrukcję warunkową)*.
4. Spróbować zmienić warunek pętli **while** z przykładu tak, aby:
    a. instrukcja pętli ani razu się nie wykonała,
    b. odbyła się nieskończona liczba powtórzeń (skutkiem będzie zawieszenie przeglądarki).

Wszystkie pliki HTML wysłać do foldera sprawozdań w TEAMS. Treść wykonanych skryptów i ich rezultaty umieścić w sprawozdaniu.