Wykład 6

# Informatyka

# MPDI 3 semestr

**JavaScript**

- operacje logiczne
- funkcje arytmetyczne
- instrukcja warunkowa

# Operatory porównania w relacjach logicznych

wyrażenie operator wyrażenie

| | |
|---|---|
| = = | **równe** |
| != | **nierówne** |
| < | |
| <= | |
| > | |
| >= | |

jest to pytanie: czy jest spełnione?
odpowiedź: tak lub nie (true/false)

Przykładowo:

x==5 x>=3 a+b> 3*y

Porównania mogą być przypisane do zmiennej:

z = **5<=6** (zmienna **z** będzie miała wartość **true**)

Relacje używane dla przypisania wartości logicznej do zmiennej lub też w innych instrukcjach (np. warunkowej) – o tym dowiemy się później

# Operatory logiczne

- koniunkcja (i) &&
- alternatywa (lub) ||
- negacja !

true && false daje false

!false daje true

## Można przypisywać wartości logiczne do zmiennej

```
x = 5
p1 =  x<4                  //false
p2 =  x == 4               //false
p3 =  2*x >= 9             //true
p4 =  x != 4               //true
p5 =  x>0 && x<=12         //true
p6 =  x>4 || x<0           //true
p7 =  (x>0) && (x<10)      //true (x w przedziale (0,10)
```

Przykład

```
<SCRIPT>
x = 7              //początkowo x ma wartość 7
x++                //zwiększamy x o 1
x /= 2             //dzielimy x przez 2
y = 3*x-1          //nowa zmienna typu liczbowego
z = x>=y           //zmienna z będzie typu logicznego (true/false)
napis = "Politechnika"   //zmienna typu tekstowego
</SCRIPT>
```

Typ zmiennej wynika z typu nadanej wartości (liczbowy, logiczny, tekstowy)

# Oczywiście ważna jest kolejność instrukcji przypisania!

```
a=5
a++ //to ma sens – teraz a ma wartość 6
```

```
b++  // jeśli nieznana wartość zmiennej b to błąd!
```

```
a = 3
b = 2*a          //to ma sens
c= 3*x           //błąd – nieznane x
```

# Instrukcja wypisania na ekranie

document.write **(lista elementów)**

**document** to wbudowany obiekt w JavaScript, a **write** (pisz) to jedna z jego funkcji (tzw. metoda)

**identyfikator obiektu i identyfikator jego metody oddzielamy kropką**

Elementy listy w metodzie **write** obiektu **document** oddzielamy przecinkami lub znakiem +

Elementami mogą być:

- teksty – otoczone " lub ' (także znaczniki HTML)
- wyrażenia obliczeniowe – wyświetlana jest ich obliczona wartość – wszystkie elementy wyrażenia muszą być znane i poprawne (wartości zmiennych, operatory, funkcje matematyczne)

# Przykład

```
<SCRIPT>
document.write ("To jest zwykły tekst<BR />")
x=5  //przypisujemy wartość zmiennej
// wyświetlamy jej wartość
document.write("Wartość zmiennej <I>x</I> : ",x,"<BR />")
napis = "Mateusz "  +  'Kowalski'
document.write(napis)
</SCRIPT>
```

Jak widzimy można wysyłać do przeglądarki znaczniki HTML(np. dla formatowania itp.)

# Wykorzystanie skryptu Javascript w dokumencie HTML

```
<HTML><HEAD></HEAD>
<BODY>
<SCRIPT>
// obiekt document i jego metoda write - wypisanie tekstu
document.write ("To jest zwykły tekst")
// wysyłamy też znacznik HTML
document.write ("<BR />")
//nadajemy wartość zmiennej
x=5
// ... i wyświetlamy jej wartość
document.write("Wartość zmiennej <I>x</I> : ",x)
document.write ("<BR>To jest liczba PI:", Math.PI)
</SCRIPT>
<P> a to już akapit poza skryptem</P>
</BODY></HTML>
```

# Obliczenia arytmetyczne w języku JavaScript

## Wykorzystujemy tu obiekt Math (uwaga! –duża litera M)

W języku istnieje wbudowany obiekt Math, zawierający:

- stałe matematyczne (właściwości - ang. *property*)
- funkcje standardowe (metody - ang. *method*).

**Stałe matematyczne:**

**Math.*property***

**Funkcje**

**Math.*method( )***

gdzie *property* lub *method* jest jednym z podanych dalej elementów.

*property* (właściwości) – stałe matematyczne

| | |
|---|---|
| **E** | **e - stała Eulera, która wynosi ok. 2.718** |
| **PI** | **wartość liczby π, czyli ok. 3.14159** |

Uwaga: DUŻE LITERY

```
document.write(Math.E)
document.write(Math.PI)
```

Są też inne, np. **Math.SQRT2** $\sqrt{2}$

# *method* (metody)

| | |
|---|---|
| **abs(wyrażenie)** | **wartość bezwzględna *liczby*** |
| **cos(wyrażenie)**<br>**sin(wyrażenie)**<br>**tan(liczba)** | **funkcje trygonometryczne (argument w radianach!!!)** |
| **sqrt(wyrażenie)** | **pierwiastek kwadratowy *liczby*** |
| **exp(wyrażenie)** | **$e^x$ UWAGA!!!** |
| **log(wyrażenie)** | **logarytm naturalny *liczby*** ! |
| **log10(wyrażenie)** | **logarytm dziesiętny (*Chrome*)** |
| **pow(liczba1,liczba2)** | **wartość *liczby1* podniesionej do potęgi *liczby2*** |
| **ceil(liczba)** | **zaokrąglenie do całkowitej w górę** |
| **floor(liczba)** | **zaokrąglenie do całkowitej w dół** |
| **round(liczba)** | **zaokrąglenie do najbliższej całkowitej** |
| **random()** | **wartość pseudolosowa z przedziału (0,1)** – bez argumentu |

## Przykłady:

```
<SCRIPT  >
document.write(Math.sin(4*Math.PI/180),"<BR />")
</SCRIPT>
```

lub wykorzystując zmienną:

```
<SCRIPT  >
wynik=Math.sin(3*Math.PI/180)
document.write(wynik)
//można też
alert ("Wynik=:"+wynik)  //w dodatkowym oknie
</SCRIPT>
```

Sekwencja obliczeń:

```
<SCRIPT  >
//Pierwiastki równania kwadratowego
a=5
b=5
c=1
delta=b*b-4*a*c
pdelta=Math.sqrt(delta)
x1=(-b-pdelta)/2/a   //albo …/(2*a)
x2=(-b+pdelta)/2/a
document.write("x1:",x1,"<BR />")
document.write("x2:",x2,"<BR />")
</SCRIPT>
```

x1:-0.7236067977499789

x2:-0.27639320225002103

Oczywiście gdy delta będzie ujemne, to błąd!

**NaN** – nieokreślone

Jak przeciwdziałać? Instrukcja badania warunku **if** (test, sprawdzenie!)

Przykład pisania wyrażeń

$$y = \frac{sin^2 x - \sqrt[3]{(x-3)x}}{\left|x^{-3}\right| + 4}$$

*zapis w skrypcie JavaScript*

```
x=Math.PI  //musimy określić wartość x

y= (Math.pow(Math.sin(x),2) - Math.pow((x-3)*x,1/3))
      /(Math.abs(Math.pow(x,-3))+4)

document.write(y)
```

łatwo o błędy (dużo nawiasów!)

Uwaga: wolno spacje, ale nie wewnątrz nazw

wolno przenieść do następnego wiersza

jak sobie ułatwić?

wprowadzać zmienne pomocnicze

liczymy etapami….

$$y = \frac{sin^2 x - \sqrt[3]{(x-3)x}}{\left|x^{-3}\right| + 4}$$

```
<SCRIPT >
x=Math.PI  //jak poprzednio
L1= Math.pow(Math.sin(x),2)
L2=Math.pow((x-3)*x,1/3)
L= L1- L2 //licznik
M= Math.abs(Math.pow(x,-3))+4  //mianownik
y= L/M  //wynik
document.write(y)
</SCRIPT>
```

# Uwaga na zagnieżdżanie funkcji

| Matematyka | JavaScript |
|---|---|
| $\sin^2 x$ | Math.pow(Math.sin(x),2)<br>lub<br>Math.sin(x)*Math.sin(x) |
| $\sqrt[3]{\ln(x)}$ | Math.pow(Math.log(x),1/3) |
| $e^{-3\sin x}$ | Math.exp(-3*Math.sin(x)) |

itp.

Poprawne pisanie wymaga "treningu"

Standard notacji dla wielu platform obliczeniowych

# Najczęstsze błędy

- pominięcie operatora mnożenia *
- niedomknięte lub źle postawione nawiasy sterujące hierarchią działań
- nieprawidłowe nazwy (np. duże litery zamiast małych w nazwach zmiennych i funkcji)
- brak konwersji typu tekstowego na liczbowy dla sumowania
- użycie ^ dla potęgowania zamiast funkcji *pow*

# Generator liczb losowych

z=Math.random() $z \in (0, 1)$

jeśli chcemy zmienić przedział losowania:

z=k*Math.random() $z \in (0, k)$

jeśli chcemy dodatkowo przesunąć przedział losowania:

z=k*Math.random()+D $z \in (D, k+D)$

# Algorytm zaokrąglania do określonej liczby miejsc dziesiętnych

-mnożymy liczbę przez $10^k$ , gdzie k to liczba miejsc dziesiętnych,

- zaokrąglamy do całkowitej,

- wynik dzielimy przez $10^k$

```
z=100*Math.random()-50
```
z∈ (-50, 50) dziesiętne

```
z=Math.round(100*Math.random()-50)
```
z∈ (-50, 50) całkowite

```
z=Math.round((100*Math.random()-50)*1e3)/1e3
```
z∈ (-50, 50) liczba dziesiętna zaokrąglone do 3-ch miejsc dziesiętnych

*można też wykorzystać metodę toFixed dla zmiennej (typu liczbowego):*

```
liczba = 5.56789
n = liczba.toFixed(2)    //zaokrąglenie do 2 miejsc dziesiętnych
```

# Logarytmy o innych podstawach niż *e*

$$log_k x = \frac{\ln x}{\ln k}$$

Logarytm dziesiętny liczymy:

```
<SCRIPT>
x= 1000
y= Math.round(Math.log(x)/Math.log(10)*1E3)/1E3
document.write("log(", x, ")=",y)
</SCRIPT>
```

otrzymamy:

**log(1000)=3**

W przeglądarce Chrome można korzystać z funkcji Math.log10(x)

# Debugging – detekcja błędów kodu JavaScript

Przykładowo:

```
…
document.write("log(5)<BR />")
document.write(Math.log(5)/Math.ln(10)+"<BR />")
…
```

W **Edge**– Menu Więcej narzędzi../Narzędzia programistyczne:

- zakładka Console

*Po odświeżeniu strony w konsoli pojawi się komunikat:*

Brak definicji „ln”

W **Chrome** – Menu Więcej narzędzi../Narzędzia dla deweloperów - Console

Udefined function ln

# Błędy arytmetyczne

NaN – not a number

np. Math.sqrt(-3)  Math.log(-5)

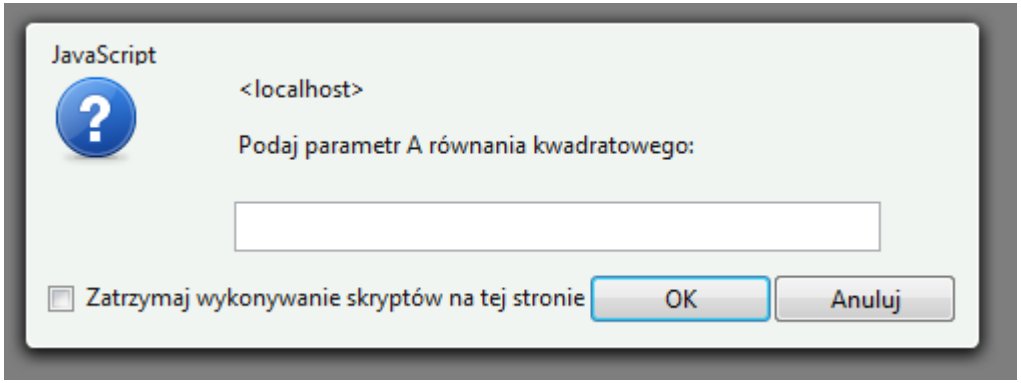
także pierwiastki stopni nieparzystych z liczb ujemnych!

Infinity - przekroczenie zakresu (za duża liczba)

np. Math.pow(100, 300)

# Wprowadzanie danych przez użytkownika

**prompt** jest metodą obiektu window – pojawia się okienko z polem edycyjnym do wpisania **tekstu** (nawet liczba jest interpretowana jako tekst)




```
x= prompt(" Podaj parametr A równania kwadratowego:")

liczba= Number(x)
```

Funkcja **prompt** jako rezultat zwraca wartość typu tekstowego, który możemy przechować w zmiennej

Jeśli podajemy dane rozumiane jako liczby – szczególnie gdy będziemy je dodawać (dwoistość operatora +)- konieczna jest konwersja na typ liczbowy

# Ilustracja

```
<SCRIPT>
x= prompt(" Podaj liczbę:")  //podajemy 56
liczba= Number(x)
liczba2=parseInt(x) //alternatywnie
document.write(x+1,"<BR>")  //wypisze 561
document.write(liczba+1,"<BR>")  //wypisze 57
document.write(liczba2+1,"<BR>")  //wypisze 57
document.write(typeof x,"<BR>") //wypisze string
document.write(typeof liczba,"<BR>") //wypisze number
document.write(typeof liczba2,"<BR>") //wypisze number
</SCRIPT>
```

## Przykład programu z dialogiem:

```
<SCRIPT  >
document.write ("Pole trójkąta <BR />")
a= prompt ("Podaj długość podstawy trójkąta:")
h= prompt ("Podaj wysokość trójkąta:")
P=a*h/2
document.write ("Pole trójkąta =", P , "<BR />")
</SCRIPT>
```

# Uwaga na sumowanie !!!

```
<SCRIPT >
document.write ("Sumowanie<BR />")
a= prompt ("Podaj liczbę 1:")
b= prompt ("Podaj liczbę 2:")
suma=a+b
document.write ("Suma=", suma , "<BR />")
</SCRIPT>
```

Jeśli podamy liczby 4 i 6 to wynik będzie 46

Dlaczego? Bo dane z okienka *prompt* są typu tekstowego i mamy sklejenie tekstów (operator +)

# Poprawne sumowanie danych z okienek prompt

```
<SCRIPT  >
document.write ("Sumowanie<BR />")
a= prompt ("Podaj liczbę 1:")
b= prompt ("Podaj liczbę 2:")
suma=Number(a)+Number(b)
document.write ("Suma=", suma , "<BR />")
</SCRIPT>
```

Wbudowana funkcja **Number** dokonuje konwersji typu tekstowego na typ liczbowy (o ile poprawnie wpisano liczbę w okienku prompt)

albo …

```
<SCRIPT  >
document.write ("Pole trójkąta <BR />")
a= Number(prompt ("Podaj liczbę 1:"))
b= Number(prompt ("Podaj liczbę 2:"))
suma=a+b
document.write ("Suma=", suma , "<BR />")
</SCRIPT>
```

# Instrukcje strukturalne w JavaScript

## Takie instrukcje, w skład których wchodzi inna instrukcja

Przykładowo:

- zbadaj warunek i jeśli jest prawdziwy wykonaj jedną instrukcję (lub blok instrukcji), a jeśli nieprawdziwy inne instrukcje
- powtarzaj jakąś instrukcję wiele razy

# Instrukcje strukturalne

## w JavaScript

!!!

**if** instrukcja warunkowa (badanie warunku)

**for**

**while**

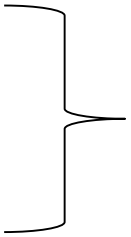
**do while**

} iteracje (pętle)

**są trudniejsze, mogą powstawać konstrukcje zagnieżdżane**

# Instrukcja warunkowa if...else

```
if (warunek) {
instrukcje 1
}
else {
instrukcje 2
}
```



*blok **else** opcjonalny (czyli można opuścić) - wówczas przy niespełnionym warunku program przechodzi do następnej instrukcji po if*

**Warunek** jest porównaniem wyrażeń arytmetycznych lub wyrażeniem logicznym – ma wartość **true (prawda**) lub **false (fałsz**).

Instrukcja **if** powoduje wykonanie kodu źródłowego *instrukcje1* tylko wtedy, gdy warunek logiczny ma wartość true. Jeżeli zostanie użyty poszerzony wariant instrukcji if , to gdy warunek jest prawdziwy (true) zostanie wykonany kod pierwszy, w przeciwnym wypadku (false) zostanie wykonany kod *instrukcje2*.

# Prosty przykład dla instrukcji warunkowej

```
<HTML><HEAD></HEAD><BODY>
<SCRIPT>
x=prompt('Podaj x:')
if (x>0)
 document.write("tak")
  else
 document.write("nie")
</SCRIPT>

</BODY></HTML>
```

*Nawiasy klamrowe można pominąć, jeśli do wykonania jest jedna instrukcja*

# Zagnieżdżanie instrukcji if (jedna wewnątrz drugiej) – jeśli mamy kilka warunków

```
if (warunek1) {
kod wykonywany jeżeli warunek1 spełniony
}
else  if (warunek2){
        instrukcje  wykonywane jeżeli warunek2 spełniony (warunek1
        niespełniony)
        }
        else
                if (warunek3){
                instrukcje  wykonywane jeżeli warunek2 spełniony
                (warunki 1 i 2 niespełnione)
                }
                else {
                instrukcje  wykonywane jeżeli żaden z warunków
                niespełniony
                }
```

wcięcia zwiększają czytelność kodu

Jeśli któryś warunek jest prawdą (true) wykonywane są odpowiednie instrukcje kodu i reszta analiz jest pomijana

# Przykład badania kilku warunków:

```
<HTML><HEAD></HEAD><BODY>
<SCRIPT  >
x=Math.round(Math.random()*10) //losowanie – l. całk. od 0 do 10
if (x<5) {
          document.write("Mniejsze od 5")
         }
else if ((x>=5)&&(x<=8) ){// koniunkcja (przedział)
          document.write("W przedziale [5, 8]")
         }
      else if ((x>8) &&(x<10)){// inny przedział
          document.write("W przedziale (8,10) więc tylko 9")
            }
            else
            {//pozostałe
            document.write("Pozostało tylko 10")
            }
document.write("<BR />Sprawdzam :",x)
</SCRIPT>
</BODY></HTML>
```

# Poprzedni kod można uprościć (dlaczego?)

```
<HTML><HEAD></HEAD><BODY>
<SCRIPT  >
x=Math.round(Math.random()*10) //losowanie – l. całk. od 0 do 10
if (x<5) {
          document.write("Mniejsze od 5")
         }
else if (x<=8) {// koniunkcja (przedział)
          document.write("W przedziale [5, 8]")
         }
      else if (x<10){// inny przedział
          document.write("W przedziale (8,10) więc tylko 9")
            }
            else
            {//pozostałe
            document.write("Pozostało tylko 10")
            }
document.write("<BR />Sprawdzam :",x)
</SCRIPT>
</BODY></HTML>
```