Lekcja 3

Temat: Operatory

W JavaScripcie, podobnie jak i w innych językach programowania, występuje wiele operatorów, które pozwalają na wykonywanie rozmaitych operacji. Operatory te można podzielić na następujące grupy:

- arytmetyczne,
- przypisania,
- logiczne,
- porównywania (relacyjne),
- bitowe,
- pozostałe.

Można więc wykonywać operacje dodawania, odejmowania, porównywania, przypisania i wiele innych.

Operatory arytmetyczne

Operatory z tej grupy służą do wykonywania operacji arytmetycznych, czyli dodawania, odejmowania, mnożenia itp. Operatory *, /, +, -, % są dwuargumentowe.

Operator	**Wykonywane działanie**	**Przykład**
+	dodawanie	x + y
-	odejmowanie	x - y
*	mnożenie	x * y
**	potęgowanie	X**2
/	dzielenie	x / y
%	dzielenie modulo (reszta z dzielenia)	x % y
++	inkrementacja (zwiększanie)	x++, ++x
--	dekrementacja (zmniejszanie)	x--, --x

Przykład 1. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p1.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory arytmetyczne - l3p1 </title>

        <script type="text/javascript">
            var suma=10+20;
            var roznica=10-20;
            var iloczyn=10*20;
            var iloraz=10/20;

            alert("Operator dodawania: 10+20 = "+suma);
            alert("Operator odejmowania: 10-20 = "+roznica);
            alert("Operator mnożenia: 10*20 = "+iloczyn);
            alert("Operator dzielenia: 10/20 = "+iloraz);
        </script>
    </head>
    <body>
        <h2> Operatory arytmetyczne</h2>
    </body>
</html>
```

Przykład 2. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p2.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory arytmetyczne 2 - l3p2 </title>

        <script type="text/javascript">
            var mod1=4 % 2; // w tym przypadku nie ma reszty
            var mod2=4 % 3; // w tym przypadku mamy resztę
            var mod3=8 % 3; // w tym przypadku mamy resztę
            var mod4=1 % 3; // co w tym przypadku?
            var mod5=0 % 3; // co w tym przypadku?
            var mod6=3 % 0; // a co z dzieleniem przez 0?

            alert("Reszta z dzielenia 4 przez 2 to: " + mod1);
            alert("Reszta z dzielenia 4 przez 3 to: " + mod2);
            alert("Reszta z dzielenia 8 przez 3 to: " + mod3);
            alert("Reszta z dzielenia 1 przez 3 to: " + mod4);
            alert("Reszta z dzielenia 0 przez 3 to: " + mod5);
            alert("Reszta z dzielenia 3 przez 0 to: " + mod6);
        </script>
    </head>
    <body>
        <h2> Operatory arytmetyczne 2</h2>
    </body>
</html>
```

Wartości specjalne — nieskończoność i NaN

W wyniku dzielenia modulo przez 0 pojawiło się**: NaN**. Słowo kluczowe NaN oznacza „nie-liczbę” (ang. *Not a Number*), która jest zwracana, gdy wykonujemy jakąś nieprawidłową operację liczbową

Sprawdź jaki będzie wynik przy zwykłym dzieleniu przez 0.

Można użyć wartości Infinity i -Infinity, aby zdefiniować nieskończenie duże liczby dodatnie lub ujemne:

```
var reallyBigNumber = Infinity;
var reallySmallNumber = -Infinity;
```

Są niewielkie szanse, że będziesz musiał użyć wartości Infinity. Zamiast tego prawdopodobnie zobaczysz ją zwracaną w ramach jakiejś innej operacji wykonywanej przez kod. Przykładowo: zobaczysz zwracaną wartość Infinity, jeśli będziesz dzielić przez 0.

Operatory przypisania

Podstawowym operatorem przypisania jest znany Ci już znak =.
W języku JavaScript znak równości (=) jest operatorem "przypisania", a nie operatorem "równym".
Tu różni się od algebry. W algebrze nie ma sensu:

```
x = x + 5
```

W JavaScript jednak ma sens: przypisuje wartości x + 5 do x. (Oblicza wartość x + 5 i ustawia wynik na x. Wartość x jest zwiększana o 5.)

Operator	Przykład	Równoznaczne z
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Przykład 3. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p3.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory przypisania - l3p3 </title>
    </head>
    <body>
        <h2> Operatory przypisania</h2>
        <script type="text/javascript">
            var x = 10;
            x -= 5;
            document.write("Dla x=10, operator x -= 5, efekt: " + x +"<br>");
            x+=8;
            document.write("operator x += 8, efekt: " + x +"<br>");
            x*=2;
            document.write("operator x *= 2, efekt: " + x +"<br>");
            x/=13;
            document.write("operator x /= 13, efekt: " + x +"<br>");
            x%=3;
            document.write("operator x %= 3, efekt: " + x +"<br>");
        </script>
    </body>
</html>
```

Operator + jak wiesz z poprzedniej lekcji jest jednocześnie operatorem konkatenacji, służącym do łączenia ciągu znaków. Można zatem wykorzystać go także w następujący sposób, pokazany w poniższym przykładzie. Dodatkowo należy pamiętać, że w zależności od użytych typów (znaki lub liczby) otrzymamy różne efekty. Dodając dwie liczby, operator zwróci sumę, ale dodając liczbę i ciąg zwróci łańcuch znaków.

Przykład 4. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p4.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory przypisania 2- l3p4 </title>
    </head>
    <body>
        <h2> Operatory przypisania 2</h2>
        <script type="text/javascript">
            //konkatenacja
            var myVar = 'Przykładowy tekst';
            myVar += ' który nie';
            myVar += ' zmieścił by się w jednej linijce';

            document.write(myVar+"<br>");
            //uzycie różnych typów danych
            x = 5 + 5;
            y = "5" + 5;
            z = "Witaj " + 5;
            document.write(x+"<br>");
            document.write(y+"<br>");
            document.write(z+"<br>");
        </script>
    </body>
</html>
```

Inkrementacja i dekrementacja

Do typowych rzeczy, jakie wykonuje się na liczbach należy inkrementacja (zwiększanie) lub dekrementacja (zmniejszanie) zmiennej o określoną wartość. Poniżej znajduje się przykład inkrementacji zmiennej i o wartość 1:

```
var i = 4;
i = i + 1;
```

Nie trzeba zwiększać lub zmniejszać koniecznie o 1. Można użyć dowolnej wartości:

```
var i = 100;
i = i - 2;
```

Powinieneś już zauważyć pewien wzorzec. Z uwagi na częste korzystanie z tego wzorca mamy kilka operatorów, które nieco to upraszczają.

Wyrażenie	Co robi
i++	Inkrementuje i o 1 (i = i + 1)
i--	Dekrementuje i o 1 (i = i - 1)
i += n	Inkrementuje i o n (i = i + n)
i -= n	Dekrementuje i o n (i = i - n)

Jeśli użyje się tych operatorów dla wcześniejszych przykładów, kod będzie wyglądać następująco:

```
i++;
i -= 2;
```

Operator inkrementacji(dekrementacji), zapisywany jako ++(--), może występować w dwóch formach: **przyrostkowej bądź przedrostkowej**. Oznacza to, że jeśli mamy zmienną, która nazywa się np. x, forma przedrostkowa będzie wyglądać tak: ++x, natomiast przyrostkowa tak: x++. Oba te wyrażenia zwiększą wartość zmiennej x o jeden, jednak wcale nie są sobie równoważne. Otóż operacja x++ zwiększa wartość zmiennej po jej wykorzystaniu, natomiast ++x przed jej wykorzystaniem.

Przyjrzyjmy się poniższemu przykładowi:

```
var i = 4;
var j = i++;
```

Po wykonaniu tych dwóch linii kodu wartość i będzie wynosiła 5, tak jak można było oczekiwać. Wartość j będzie wynosić 4. Należy zwrócić uwagę, że w tym przykładzie operator znajduje się po zmiennej.

Jeżeli umieścimy operator przed zmienną, wyniki będą nieco inne:

```
var i = 4;
var j = ++i;
```

Wartość i nadal będzie wynosić 5. A oto niespodzianka: wartość j również będzie wynosić 5.

Między pierwszym i drugim przykładem zmieniła się pozycja operatora. Pozycja operatora określa, *czy zwracana będzie wartość po inkrementacji (dekrementacji), czy przed inkrementacją (dekrementacją)*.

Przykład 5. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p5.html.*

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory inkrementacji i dekrementacji- l3p5 </title>
    </head>
    <body>
        <h2> Operatory inkrementacji i dekrementacji</h2>
        <script type="text/javascript">
            var i = 4;
            var j = i++;
            document.write("i= "+i+" j= " +j+ "<br>");

            var k = 4;
            var m = ++k;
            document.write("k= "+k+" m= "+m);

        </script>
    </body>
</html>
```

`j = i++` możesz interpretować następująco: przypisz **j** wartość **i**, a przy następnym wywołaniu **i** zwiększ jej wartość o 1. Wyświetlenie **i** jest jej wywołaniem, zatem wartość zostaje zwiększona, **j** pozostaje takie jak pierwotna wartość **i**.

`m = ++k` interpretacja: zwiększ wartość k o 1 i dopiero wtedy przypisz ją do m.

Przykład 6. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p6.html.*

Przeanalizuj poniższy kod. Nie wczytuj skryptu do przeglądarki, ale zastanów się, jaki będzie wyświetlony ciąg liczb. Następnie po uruchomieniu skryptu sprawdź swoje przypuszczenia.

```
<! DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Operatory inkrementacji i dekrementacji- l3p6 </title>
    </head>
    <body>
        <h2> Operatory inkrementacji i dekrementacji</h2>
        <script>
            var x = 12;
            var y;
/*1*/ document.write(++x);
/*2*/ document.write(" ");
/*3*/ document.write(x++);
/*4*/ document.write(" ");
/*5*/ document.write(x);
/*6*/ document.write(" ");
/*7*/ y = x++;
/*8*/ document.write(y);
/*9*/ document.write(" ");
/*10*/ y = ++x;
/*11*/ document.write(y);

        </script>
    </body>
</html>
```

Wynikiem działania skryptu (dla ułatwienia opisu wiersze zostały ponumerowane) będzie ciąg znaków 13 13 14 14 16. Dlaczego? Otóż w wierszu 1. najpierw jest zwiększana wartość zmiennej x o 1 (czyli x = 13), a następnie ten wynik jest wyświetlany. W linii 3. najpierw jest wyświetlana aktualna wartość zmiennej x (czyli 13), a następnie jest ona zwiększana o 1 (czyli x = 14). W wierszu 5. jest wyświetlana aktualna wartość zmiennej x, czyli 14. W wierszu 7. zmiennej y jest przypisywana wartość zmiennej x, a następnie zmienna x jest zwiększana o 1 (czyli y = 14, x = 15). W wierszu 10. najpierw jest zwiększana wartość zmiennej x o 1 (czyli x = 16), a następnie wartość ta jest przypisywana zmiennej y (czyli y = 16 i x = 16).

Przykład 7. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako *l3p7.html.*

Zmień kod z przykładu 6 tak, aby operator ++ został zastąpiony operatorem --. **Przeanalizuj kod. Nie wczytuj skryptu do przeglądarki, ale zastanów się, jaki będzie wyświetlony ciąg liczb. Następnie po uruchomieniu skryptu sprawdź swoje przypuszczenia.**

```
1   <! DOCTYPE html>
2   <html>
3       <head>
4           <meta charset="UTF-8">
5           <title>Operatory inkrementacji i dekrementacji- l3p7 </title>
6       </head>
7       <body>
8           <h2> Operatory inkrementacji i dekrementacji</h2>
9           <script>
10              var x = 12;
11              var y;
12  /*1*/ document.write(--x);
13  /*2*/ document.write(" ");
14  /*3*/ document.write(x--);
15  /*4*/ document.write(" ");
16  /*5*/ document.write(x);
17  /*6*/ document.write(" ");
18  /*7*/ y = x--;
19  /*8*/ document.write(y);
20  /*9*/ document.write(" ");
21  /*10*/ y = --x;
22  /*11*/ document.write(y);
23
24          </script>
25      </body>
26  </html>
```

Operatory logiczne

Operacje logiczne mogą być wykonywane na argumentach, które posiadają wartość logiczną:

prawda (true) lub fałsz (false).

Operatory logiczne zostały przedstawione w tabeli:

Operator	Wykonywane działanie	Przykład
&&	iloczyn logiczny (AND)	a && b
||	suma logiczna (OR)	a || b
!	negacja logiczna (NOT)	!a

Iloczyn logiczny

Wynikiem iloczynu logicznego jest wartość **true wtedy i tylko wtedy, kiedy oba argumenty mają wartość true**. W każdym innym przypadku wynikiem jest false

Działanie iloczynu logicznego:

Argument 1	Argument 2	Wynik &&
true	true	true
true	false	false
false	true	false
false	false	false

Suma logiczna

Wynikiem sumy logicznej jest **wartość false wtedy i tylko wtedy, kiedy oba argumenty mają wartość false**. W każdym innym przypadku wynikiem jest true.

Działanie sumy logicznej:

Argument 1	Argument 2	Wynik ||
true	true	true
true	false	true
false	true	true
false	false	false

Negacja logiczna

Operacja logicznej negacji zamienia wartość argumentu na przeciwną. Czyli jeśli argument miał wartość true, będzie miał wartość false; i odwrotnie, jeśli miał wartość false, będzie miał wartość true.

Operatory && i || są dwuargumentowe, natomiast operator ! jest jednoargumentowy.

Z operatorów logicznych będziemy korzystać już wkrótce podczas pracy z instrukcjami warunkowymi.

Operatory porównywania (relacyjne)

Operatory porównania, czyli relacyjne, służą oczywiście do porównywania argumentów. Wynikiem takiego porównania jest wartość logiczna true (jeśli jest ono prawdziwe) lub false (jeśli jest fałszywe). Zatem wynikiem operacji argument1 == argument2 będzie true, jeżeli argumenty są sobie równe, oraz false, jeżeli argumenty są różne. Czyli 4 ==5 ma wartość false, a 2 ==2 ma wartość true. Do dyspozycji mamy operatory porównania

Operator	Opis	Przykład
==	Wynikiem jest true, jeśli argumenty są sobie równe. W przeciwnym przypadku wynikiem jest false.	x == y
!=	Wynikiem jest true, jeśli argumenty są różne. W przeciwnym przypadku wynikiem jest false.	x != y
===	Wynikiem jest true, jeśli oba argumenty są tego samego typu i są sobie równe. W przeciwnym przypadku wynikiem jest false.	x === y
!==	Wynikiem jest true, jeśli argumenty są różne bądź są różnych typów. W przeciwnym przypadku wynikiem jest false.	x !== y
>	Wynikiem jest true, jeśli argument lewostronny jest większy od prawostronnego. W przeciwnym przypadku wynikiem jest false.	x > y
<	Wynikiem jest true, jeśli argument lewostronny jest mniejszy od prawostronnego. W przeciwnym przypadku wynikiem jest false.	x < y
>=	Wynikiem jest true, jeśli argument lewostronny jest większy od prawostronnego lub mu równy. W przeciwnym przypadku wynikiem jest false.	x >= y
<=	Wynikiem jest true, jeśli argument lewostronny jest mniejszy od prawostronnego lub mu równy. W przeciwnym przypadku wynikiem jest false.	x <= y

przykłady dla x = 5

Operator	Opis	Równanie	Zwróci
==	równe	x == 8	false
!=	różne	x != 8	true
===	równa wartość i taki sam typ danych W naszym przykładzie x to numer, a 5 po prawej stronie jest tekstem	x === 5	true
		x === "5"	false
!==	różne i inny typ danych W naszym przykładzie x to numer, a 5 po prawej stronie jest tekstem	x !== "5"	true
		x !== 5	false
>	większe od	x > 8	false
<	mniejsze od	x < 8	true
>=	większe bądź równe od	x >= 8	false
<=	mniejsze bądź równe od	x <= 8	true

Operatory te, podobnie jak logiczne wykorzystamy już wkrótce w instrukcjach warunkowych.