# JAVASCRIPT (1)

**Do umieszczenia skryptów na stronie służy znacznik <script>:**

```
<html>
<head>
    <title>Super skrypt</title>
</head>
<body>
<script>
    ...instrukcje skryptu
</script>
</body>
```

W przypadku html4 trzeba do tagu script dodać dodatkowy atrybut type="javascript".

---

**Aby przyłączyć plik ze skryptami do naszej strony powinniśmy zastosować dodatkowy atrybut src:**

```
<script src="plik_ze_skryptem.js"></script>
```

---

**Typy danych**

W Javascript dane dzielą się na 2 typy: typy **proste** oraz **referencje**. Dane typu prostego reprezentują proste typy danych - liczby, teksty, wartości boolowskie (prawda/fałsz), niezidentyfikowane (undefined) oraz null.

```
//przykład danych typu prostego
var varNr = 20; //prosta liczba
var text = 'To jest tekst'; //prosty łańcuch znaków
var varBol = true; //logiczny - prawda/fałsz
var someVar = null; //nic

undefined //zmienna nie określona, nie istniejąca
```

Powyższe typy danych jak sama nazwa wskazuje są prostymi bytami, które poza wartością nic w sobie nie mają. Wszystkie zmienne nie będące typem prostym są obiektami i są typu referencyjnego (są obiektami). Ten typ danych charakteryzuje sie tym, że zmienne nie mają przypisanej bezpośrednio wartości, a tylko wskazują na miejsce w pamięci, gdzie te dane są przetrzymywane.

**Sprawdzanie typu danych**

Do sprawdzenia typu danych korzystamy z instrukcji typeof:

```
var n = 10; //liczba
var s = 'tekst'; //tekst
var a = []; //tablica
var o = {}; //obiekt
//zmienna x nie została zadeklarowana

//sprawdzamy typy zmiennych
if (typeof n === "number") {...}
if (typeof s === "string") {...}
if (Array.isArray(a)) {...}
if (typeof o === "object") {...}
if (typeof x === "undefined") {...}
```

## Operatory

| Operator | Nazwa działania | Równanie | Wynik | |
|---|---|---|---|---|
| + | Dodawanie | x = y + 2 | y = 5 | x = 7 |
| - | Odejmowanie | x = y - 2 | y = 5 | x = 3 |
| * | Mnożenie | x = y * 2 | y = 5 | x = 10 |
| / | Dzielenie | x = y / 2 | y = 5 | x = 2.5 |
| % | Reszta z dzielenia | x = y % 2 | y = 5 | x = 1 |
| ++ | Inkrementacja | x = ++y | y = 6 | x = 6 |
| | | x = y++ | y = 6 | x = 5 |
| -- | Dekrementacja | x = --y | y = 4 | x = 4 |
| | | x = y-- | y = 4 | x = 5 |

Przykład:

```
x = 5;
y = x + 2; //y = 7
y = x - 1; //y = 4
y = x * 3; //y = 14
y = x / 2; //y = 2.5
y = x % 2; //1 bo % oznacza resztę z dzielenia
x--; //to to samo co x = x-1
x++; //to to samo co x = x+1
y = x-- //y = 4, x = 5
y = --x //y = 5, x = 4
```

## Operatory Przypisania

| Operator | Przykład | Równoznaczne z | Wynik |
|---|---|---|---|
| = | x = y | x = y | x = 5 |
| += | x += y | x = x + y | x = 15 |
| -= | x -= y | x = x - y | x = 5 |
| *= | x *= y | x = x * y | x = 50 |
| /= | x /= y | x = x / y | x = 2 |
| %= | x %= y | x = x % y | x = 0 |

**Operatory Porównania**

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

| Operator | Opis | Przykład | Wynik |
|---|---|---|---|
| && | and (i) | (x < 10 && y > 1) | Prawda, bo x jest mniejsze od 10 **i** y jest większe od 1 |
| \|\| | or (albo) | (x > 8 \|\| y > 1) | Prawda, bo x nie jest większe od 8, ale y jest większe od 1 |
| ! | not (negacja) | !(x == y) | Prawda, bo negujemy to, że x == y |

Przykład:

```
var myVar = 8;
var myVar2 = 15;

if (myVar == 8 && myVar2 == 10) {
    //ten kawałek kodu się nie wykona bo mamy "i"
}

if (myVar == 8 || myVar2 == 8) {
    //ten kawałek się wykona bo mamy "lub"
}

if (!(myVar == 8)) {
    //ten kawałek się nie wykona, bo mamy negację!
    //powyższy warunek jest jednoznaczny z myVar != 8
}
```

**Instrukcje warunkowe**

```
var x = 1;
var y = 2;

if (x > 0) {
    if (y > 0) {
        ...//jeżeli liczba > 0 i druga_liczba > 0
    }
}

//to samo co powyżej
if (x > 0 && y > 0) {
    ...//kod wykonywany jeżeli liczba > 0 i druga_liczba > 0
}
```

Lub

```
var x = 5;

if (x > 5) {
    document.write ('Liczba jest większa od 5');
} else if (x < 5) {
    document.write ('Liczba jest mniejsza od 5');
} else {
    document.write ('Liczba równa się 5');
}
```

Skrócona wersja IF

Skrócona wersja warunku IF ma postać:

(wyrażenie) ? zwróć_jeżeli_wyrażenie_true : zwróć_jeżeli_wyrażenie_false

Przykład:

```
var x = 3;
document.write((x%2==0)? 'parzysta': 'nieparzysta'); //wypisze 'nieparzysta'
```

**Instrukcja switch**

Instrukcja switch jest kolejnym sposobem testowania warunków działającym na zasadzie przyrównania wyniku do podanych przypadków.

Każdy przypadek kończy się słowem break, która kończy wykonywanie instrukcji switch. Jeżeli pominiemy to słowo, wtedy nawet przy pomyślnym przyrównaniu zostaną wykonane kolejne sprawdzenia, co często może powodować błędy. Dodatkowo instrukcja switch ma specjalny przypadek default który będzie wybierany, gdy wszystkie inne przypadki będą błędne.

```
var numer = 4;
//poniższy warunek zwróci "Numer równa się cztery"
switch (numer) {
    case 1:
        document.write ('Numer równa się jeden');
        break;
    case 2:
        document.write ('Numer równa się dwa');
        break;
    case 3:
        document.write ('Numer równa się trzy');
```

```
        break;
    case 4:
        document.write ('Numer równa się cztery');
        break;
    default:
        document.write ('Nie wiem ile równa się numer');
}

var tool = "obcęgi";
//poniższy warunek zwróci "Wybrałeś obcęgi"
switch (tool) {
    case 'wiertarka':
        document.write('Wybrałeś wiertarkę');
        break;
    case 'śrubokręt':
        document.write('Wybrałeś śrubokręt');
        break;
    case 'obcęgi':
        document.write('Wybrałeś obcęgi');
        break;
    case 'młotek':
        document.write('Wybrałeś młotek');
        break;
    default:
        document.write('Nie wiem co to za narzędzie');
}
```

**Pętle**

Pętla typu while

```
var x = 1;

while (x <= 100)
{
    document.write ("Nie będę rozmawiał na lekcji Informatyki.");
    x++;
}
```

Pętla typu do ... while

Podobnym rodzajem pętli jest pętla typu do ... while. Zasadniczą różnicą między tym typem a poprzednim jest to, że kod który ma być powtarzany zostanie wykonany przed sprawdzeniem wyrażenia.

```
var x = 1;

do {
    document.write ("Nie będę rozmawiał na lekcji Informatyki.");
    x++;
} while (x <= 100)
```

Pętla typu for

Kolejnym rodzajem pętli jest pętla typu for. Jest to najczęściej używany rodzaj pętli.

for (zainicjowanie zmiennych; wyrażenie testujące; zwiększenie/zmniejszenie zmiennej) {
  kod ktory zostanie wykonany pewną ilość razy
}

Przykładowo:

```
for (var x=0; x<100; x++) {
    document.write('Nie będę rozmawiał na lekcji Informatyki.');
}
```

Nie musisz deklarować wszystkich trzech parametrów opisujących pętlę. Dla przykładu:

```
for (var x=0; x<5; ) {
    document.write("To jest przykładowe zdanie"); //wypisze sie 3 razy
    x = x + 2;
}

var x = 3;
for (; x<10;) {
    document.write("To jest przykładowe zdanie"); //wypisze sie 7 razy
    x++;
}
```

**Funkcje**

Przykład:

```
function test1()
{
    document.write("To jest przykładowe zdanie");
}

function test2(a, b)
{
    document.write(a+b);
}

function test3(a, b)
{
    return (a+b);
}
```

**Tablice**

```
//deklaracja za pomocą literału
var ourTable = ['Marcin', 'Ania', 'Agnieszka'];

albo

var ourTable = new Array('Marcin', 'Ania', 'Agnieszka');

albo

ourTable[3] = "Piotrek";
ourTable[4] = "Grzegorz";
```

**Pętla po tablicy**

```
var ourTable = ['Marcin', 'Ania', 'Agnieszka', 'Piotrek', 'Grześ', 'Magda'];
for (var i=0; i<ourTable.length; i++) {
    document.write ('Imię numer ' +i+ ': ' +ourTable[i]);
}
```