Podstawy Programowania

Na przykładzie JavaScript - cześć 2

Tomasz Nastały

1.
Pętle

Po co nam pętle?

```
var names = ['Tomek', 'Ania', 'Janina',
'Jonasz', 'Krzysztof', 'Bartek'];

names[0] = names[0].toUpperCase();
names[1] = names[1].toUpperCase();
names[2] = names[2].toUpperCase();
names[3] = names[3].toUpperCase();
names[4] = names[4].toUpperCase();
names[5] = names[5].toUpperCase();
```

…w alternatywnym świecie bez pętli…

Po co nam pętle?

```
var names = ['Tomek', 'Ania', 'Janina',
'Jonasz', 'Krzysztof', 'Bartek'];

for (var i = 0 ; i < names.length ; i++) {
  names[i] = names[i].toUpperCase();
}
```

Po co nam pętle?

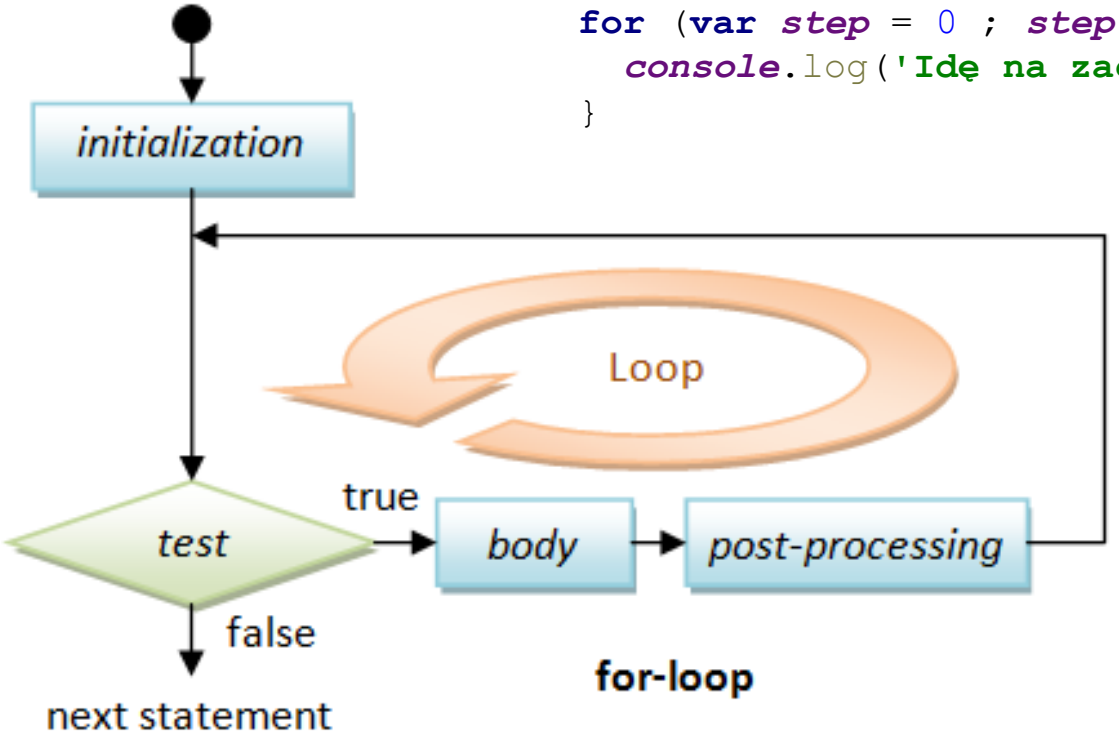
- Pętla to instrukcja, która pozwala na cykliczne wykonywanie ciągu Instrukcji określoną liczbę razy

```
for (initialization ; test ; post-processing) {
  body
}
```

```
for (var step = 0 ; step < 5 ; step++) {
  console.log('Idę na zachód jeden krok')
}
```

Pętla FOR

```
for (var step = 0 ; step < 5 ; step++) {
  console.log('Idę na zachód jeden krok')
}
```



Pętle po tablicy

```
var numbers = [1,2,3,4,5];

for (var i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
```

Pętla z dekrementacją

```
var letters = ['a', 'b', 'c', 'd', 'e'];

for (var i = letters.length - 1 ; i >= 0 ; i--){
  console.log(letters[i]);
}
```

Pętla po tablicy obiektów

```
var cars = [
  {name: 'Ford'},
  {name: 'BMW'},
  {name: 'Audi'}
];

for (var i = 0 ; i < cars.length ; i++) {
  console.log(cars[i].name);
}
```

Pętla WHILE

Gram w ruletkę do 12:00. Po czym wstanę i odejdę.

```
function isBeforeNoon() {
  // return false jeżeli po południu
  // return true jeżeli przed południem
}

while(isBeforeNoon()) {
  playAgain();
}
```

Pętla DO-WHILE

Do-while gwarantuje przynajmniej jedno wykonanie pętli.

```
function haveILost() {
  // return true jeżeli przynajmniej raz przegrałem
  // return false jeżeli ani razu nie przegrałem
}

do {
  playAgain();
} while(!haveILost());
```

break – wyjście z pętli

```
while ( .... ) {
    ...
    ...
    if ( ... ) {
        break;
    }
}
```

```
do {
    ...
    ...
    if ( ... ) {
        break;
    }
} while ( ... )
```

```
for ( ... ; ... ; ... ) {
    ...
    ...
    if ( ... ) {
        break;
    }
}
```

continue – wyjście z bieżącego cyklu pętli

```
while ( .... ) {
    ...
    ...
    if ( ... ) {
        continue;
    }
}
```

```
do {
    ...
    ...
    if ( ... ) {
        continue;
    }
} while ( ... )
```

```
for ( ... ; ... ; ... ) {
    ...
    ...
    if ( ... ) {
        continue;
    }
}
```

ZADANIA

2.
Operatory porównujące i instrukcje warunkowe

Operatory porównujące

- A == B
- A != B
- A === B
- A !== B
- A < B
- A <= B
- A > B
- A >= B

== vs ===

```
3 == '3';
// Prawda, te same wartości (to nic, że różne typy)

3 === '3';
// Fałsz, te same wartości ale różne typy!

'Adam' === 'Adam';
// Prawda
```

Porównywanie obiektów

```
[] == [], [] === []
// fałsz

[1,2,3] === [1,2,3]
// fałsz

{ name: 'Adam' } === { name: 'Adam' }
// fałsz
```

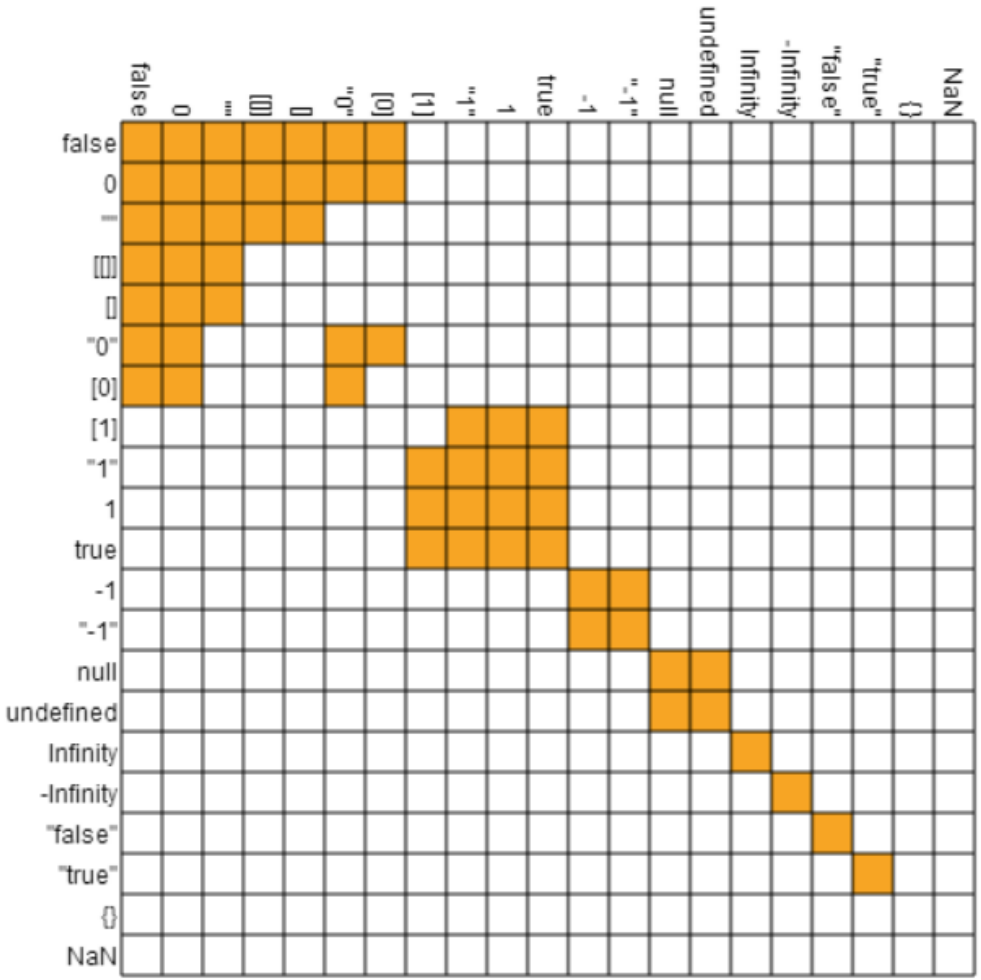
Porównywanie obiektów

```
var person = { name: 'Adam' };
var personCopied = person;

person === personCopied;
// true
```

```
var numbers = [1, 2, 3];
var numbersCopied = numbers;

numbers === numbersCopied;
// true
```



Tablica prawdy dla słabych porównań (==)

http://algassert.com/visualization/2014/03/27/Better-JS-Equality-Table.html

Operatory logiczne – Algebra Boole'a

1 === true
0 === false

AND

$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

OR

$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

NOT

$X = \bar{A}$

A	X
0	1
1	0

Operatory logiczne

- || lub
- && iloczyn
- ! negacja

```
var x = 3, y = 2;

if (x === 3 || y === 5) {
  // true
}
if (x < 10 && y > 1) {
  // true
}
if (x === 5 && y === 5) {
  // false
}
if (!( x === y)) {
  // true
}
```

if

OK!

```
var goodWeather = true;

if (goodWeather) {
    takeAWalk();
}
```

BAD!

```
var goodWeather = true;

if (goodWeather === true) {
    takeAWalk();
}
```

if-else

```
var goodWeather = true;

if (goodWeather) {
    takeAWalk();
} else {
    playPS3():
}
```

if-else-if

```
var goodWeather = true;
var padCount = 2;

if (goodWeather) {
    takeAWalk();
} else if (padCount === 2) {
    playPS3();
} else {
    watchAMovie();
}
```

UWAGA! ELSE / IF nie sprawdza pozostałych warunków po spełnieniu jednego!

Falsy values – wartości z natury fałszywe

```
if (false) { } else { }
if (null) { } else { }
if (undefined) { } else { }
if ('') { } else { }
if (0) { } else { }
if (NaN) { } else { }
```

W każdym z tych przypadków, instrukcja przejdzie od razu do ELSE

switch

Jeżeli trafię 6tkę to pojadę dookoła świata, jeżeli 5tkę to kupię komputer, jeżeli 4kę to znowu zagram. W każdym innym przypadku dam sobie spokój.

```
switch (numbersHit) {
  case 6:
    roundWorldTrip(); break;
  case 5:
    buyPC(); break;
  case 4:
    playAgain(); break;
  default:
    giveUp();
}
```

Ternary operator

```
CONDITION ? BLOCK 1 : BLOCK 2;
```

```
if condition is TRUE
{
    execute block one;
} else {
    execute block two;
}
```

```
var elvisLives = Math.PI > 4 ? 'Yep' : 'Nope';
```

ZADANIA

THE END

Pytania?

nastalytomasz@gmail.com
Tomasz Nastały @slack