PODSTAWY PROGRAMOWANIA

Na przykładzie JavaScript

Paweł Kaczmarek

Zmienne

Kontenery do przetrzymywania danych w programie

- Przetrzymują wartości – m.in. tekst, liczby
- Definiujemy je kiedy ich potrzebujemy

Zmienne – deklaracje, definicje

- **var** mojaWygrana ;
- **var** mojaWygrana = 100 ;
- **var** mojKomentarz = "Komentarz" ;
- **var** mojKomentarz = **'**Komentarz**'** ;
- **var** mojaWygrana = 100 **,** mojKomentarz = 'Komentarz' **;**

Liczba

- **var** liczbaGosci = 100 ;
- **var** PI_liczba = 3.141592 ;

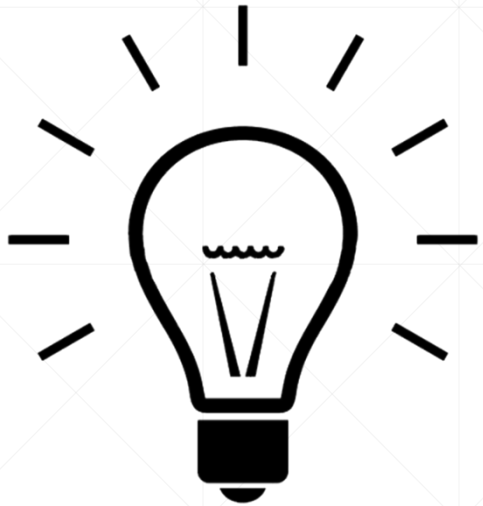
Tekst

- **var** opis = ' jakiś text ' ;
- **var** PI_text = ' 3.141592 ' ;

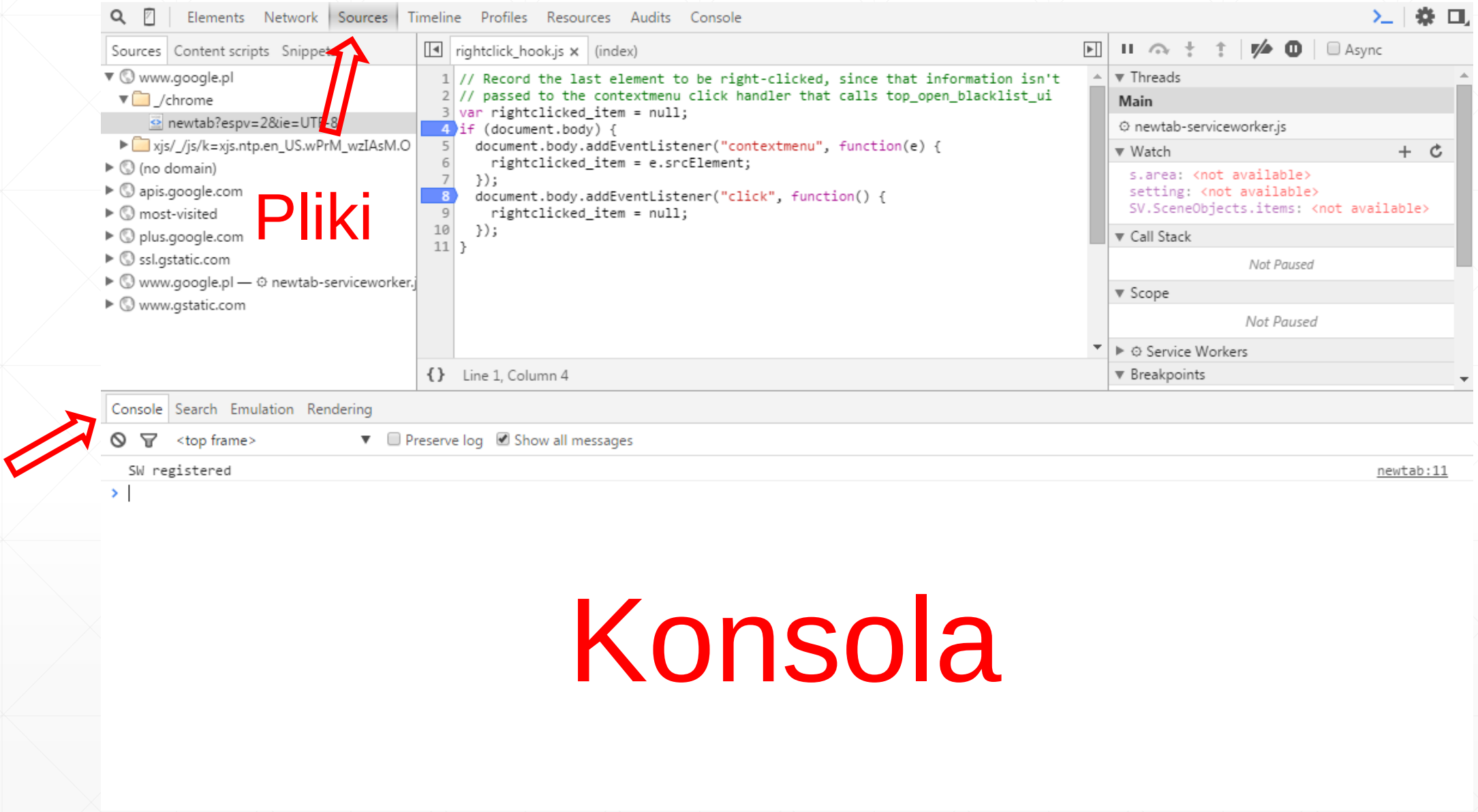
Kojeność wykonywania operacji

- **var** liczbaGosci = 100 + 30;
- **var** liczbaOsob = liczbaGosci + 15;

Osadzanie JavaScript - w pliku HTML



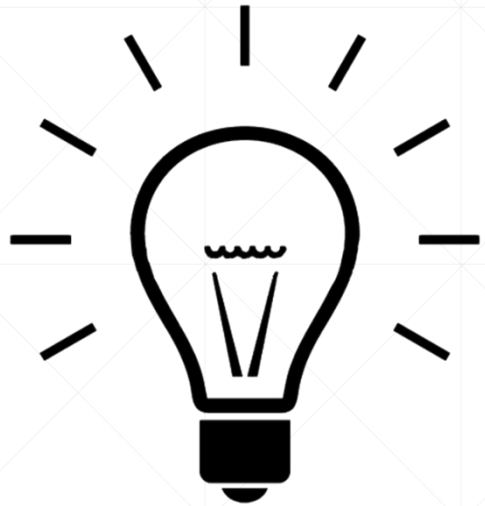
```
<script>
        console.log('test');
</script>
```


Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
Sources
Content scripts
Snippets
www.google.pl
_/chrome
newtab?espv=2&ie=UTF-8
xjs/_/js/k=xjs.ntp.en_US.wPrM_wzIAsM.O
(no domain)
apis.google.com
most-visited
plus.google.com
ssl.gstatic.com
www.google.pl — ⚙ newtab-serviceworker.j
www.gstatic.com
Pliki
rightclick_hook.js ×
(index)
1 // Record the last element to be right-clicked, since that information isn't
2 // passed to the contextmenu click handler that calls top_open_blacklist_ui
3 var rightclicked_item = null;
4 if (document.body) {
5 document.body.addEventListener("contextmenu", function(e) {
6 rightclicked_item = e.srcElement;
7 });
8 document.body.addEventListener("click", function() {
9 rightclicked_item = null;
10 });
11 }
{} Line 1, Column 4
Async
Threads
Main
⚙ newtab-serviceworker.js
Watch
s.area: <not available>
setting: <not available>
SV.SceneObjects.items: <not available>
Call Stack
Not Paused
Scope
Not Paused
⚙ Service Workers
Breakpoints
Console
Search
Emulation
Rendering
<top frame>
Preserve log
Show all messages
SW registered
newtab:11
Konsola

Jak pisać na konsolę?

```
console.log ( 'tekst, który pokaże się na konsoli' )

console.log ( 422 )
```

Jak pisać na konsolę?



```
var mojaZmienna = ' moj Text ' ;

console.log(mojaZmienna);

console.log('Zawartosc zmiennej', mojaZmienna);
```

Komentarze (ctrl + /)

```
// pierwsza linia z komentarzem

console.log("linia 1");

// druga linia z komentarzem

console.log("linia 2");
```

```
/*
Wszystkie linie, aż do znaku
zamykającego komentarze
będą pominięte
*/

console.log("linia 1");
```

Funkcje

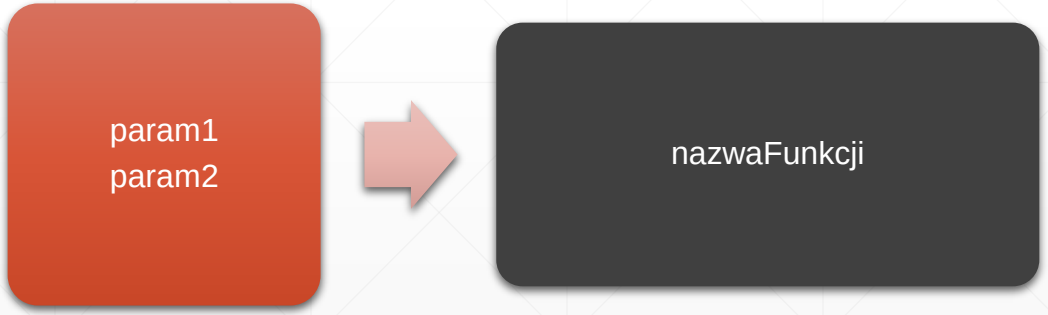
Funkcje

```
function nazwaFunkcji (wejscie1, wejscie2, wejscie3) {

        ...

        var wynik = wejscie1 + wejscie2+ wejscie3 ;

        return wynik;
}
```


param1
nazwaFunkcji
wynik


nazwaFunkcji
wynik


param1
param2
param3
nazwaFunkcji
wynik


param1
param2
nazwaFunkcji

```
function nazwaFunkcji (param1) {
        return wynik;
}
```

```
function nazwaFunkcji () {
        return wynik;
}
```

```
function nazwaFunkcji (param1, param2, param3) {
        return wynik;
}
```

```
function nazwaFunkcji (param1, param2) {
}
```

Wołanie funkcji bezparametrowej

```
function nazwaFunkcji () { ... }
```

```
nazwaFunkcji ( ) ;
```

Wołanie funkcji z parametrami

```
function nazwaFunkcji (param1, param2) {

    return wynik;

}
```

```
nazwaFunkcji ( 100 , 'jakisText' );
```

Wołania funkcji

- alert ('Komunikat')
- console.log('jakisText')
- fh2Celsius (120)

```
function fh2Celsius (fahrenheit) {
        return ( 5/9 ) * ( fahrenheit - 32 );
}
var celcius = fh2Celsius(100);
console.log( celcius );
```

Definiowanie vs Wołanie

```
function nazwaFunkcji () {
        ...
}
```

```
nazwaFunkcji ( );
```

Definiowanie vs Wołanie

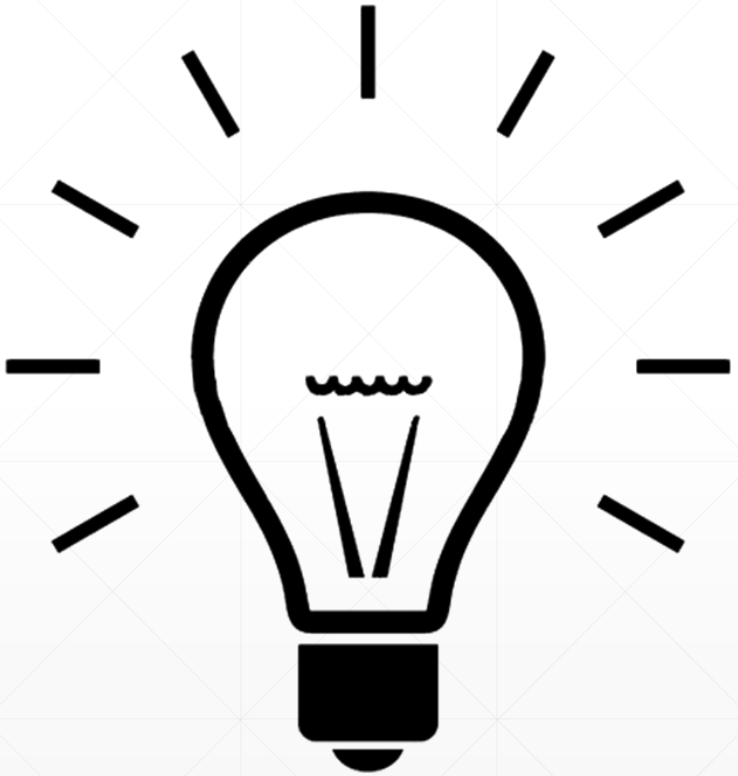
```
function nazwaFunkcji (p1, p2) {
        ...
}
```

```
nazwaFunkcji ( 10, 20 ) ;
```

Kojeność wykonywania operacji, vol 2

- **var** liczbaGosci = 100 + policzDzieci();
- **var** liczbaOsob = liczbaGosci + policzPersonel();

1.Wagoniki



Zasięg zmiennych

GLOBALNE

```
var zmienna = 1;
function policz() {
        console.log(   zmienna   );
}
```

LOKALNE

```
function policz() {
        var zmienna = 1;
}
console.log(   zmienna   );
```

Operatory arytmetyczne

+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
++	Inkrementacja
--	Dekrementacja
%	Dzielenie modulo (reszta z dzielenia)

Operatory przypisania

=	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

Instrukcje warunkowe

- if
- If-else
- switch

if

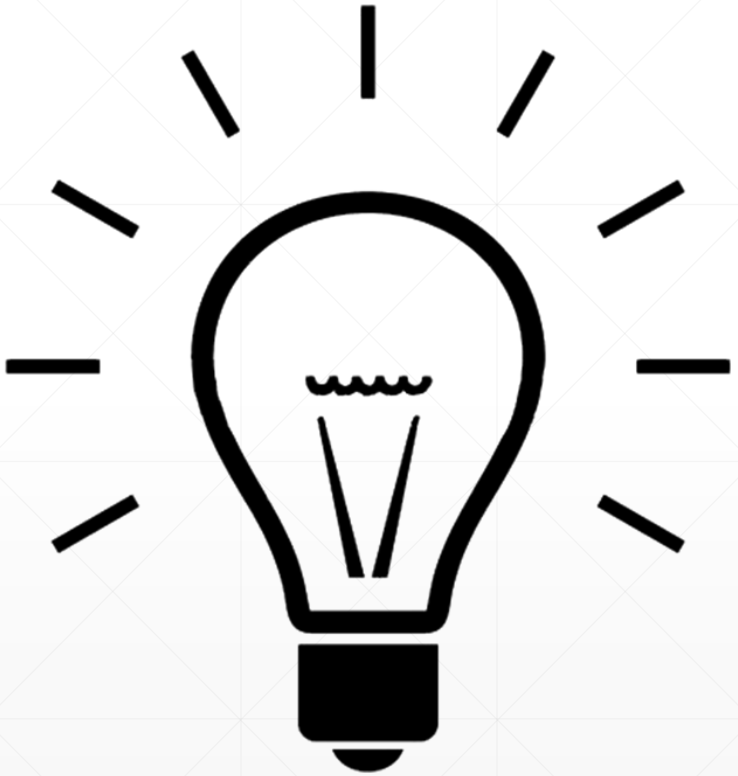
```
if ( warunek ) {
        // akcje zachodzące kiedy warunek jest prawdziwy
} else {
        // akcje zachodzące w pozostałych przypadkach
}
```

Warunki - operatory porównujące

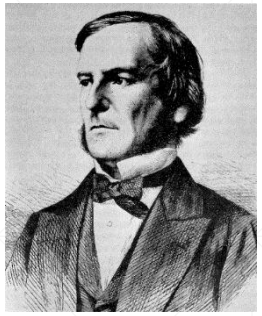
- A == B
- A != B
- A === B
- A !== B

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

2.Languages



Działania logiczne



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

$X = \bar{A}$

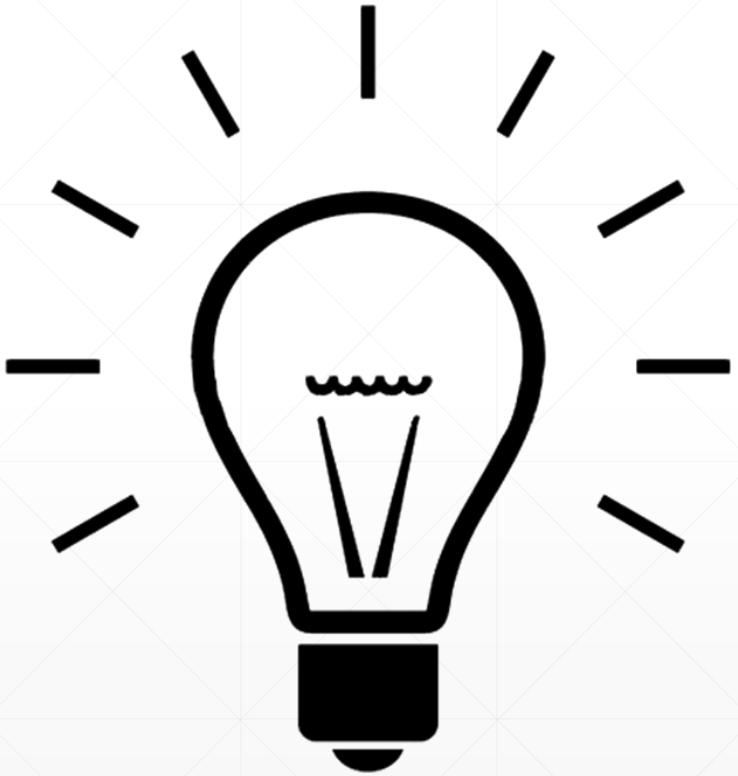
A	X
0	1
1	0

Operatory logiczne

- || lub *suma logiczna*
- && i *Iloczyn logiczny*
- ! negacja

```
var x, y, z;


if ( x ) { ... }

if ( !x ) { ... }

if ( x || y ) { ... }

if ( x || y || z ) { ... }

if ( x || y && z ) { ... }
```

2b.Promos



switch

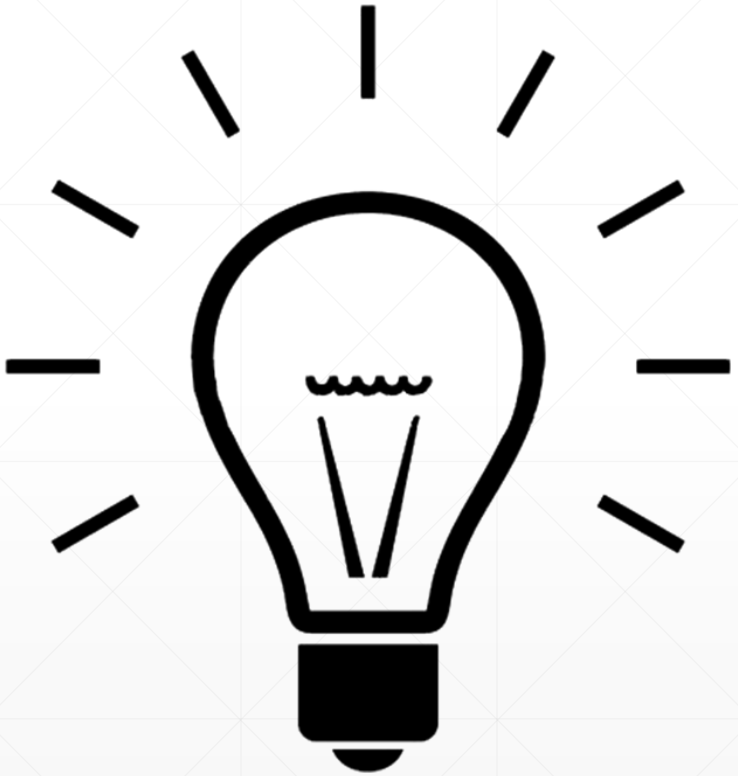
```
var mojaZmienna;

switch ( mojaZmienna ) {
        case wartosc1 :
                ....
                break;
        case wartosc2 :
                ...
                break;
        case wartosc3 :
                ...
                break;
        default:
                ...
}
```

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 ( liczbaTrafionychLiczb ) {
        case 6:
                jedziemyNaRocznyUrlop();
                break;
        case 5:
                kupujemyKomputer();
                break;
        case 4:
                gramyDalej();
                break;
        default:
                odpuszczamySobie();
}
```

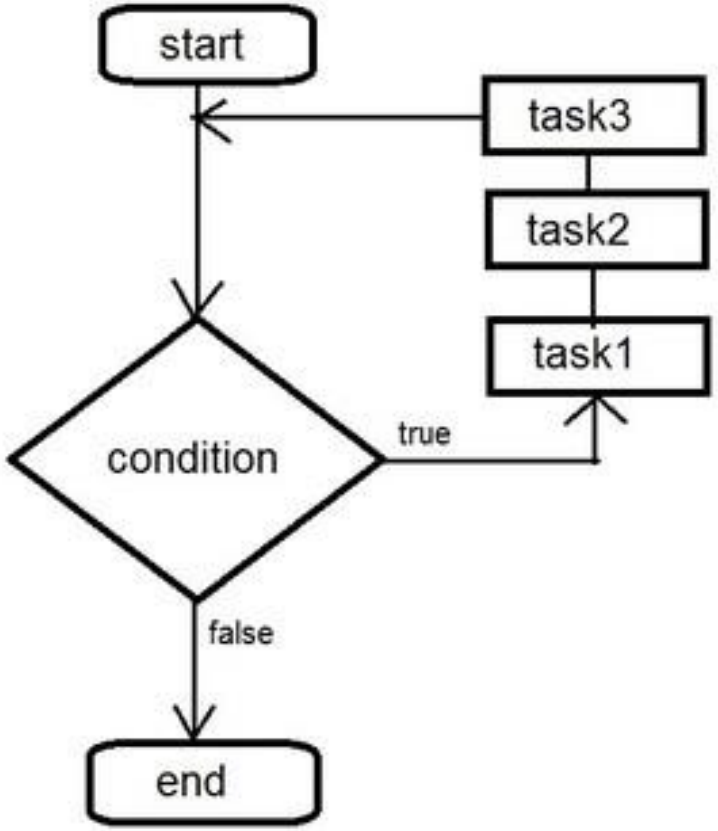
2a.Languages - with switch



Pętle

- while
- do-while
- for
- break
- continue

Pętle



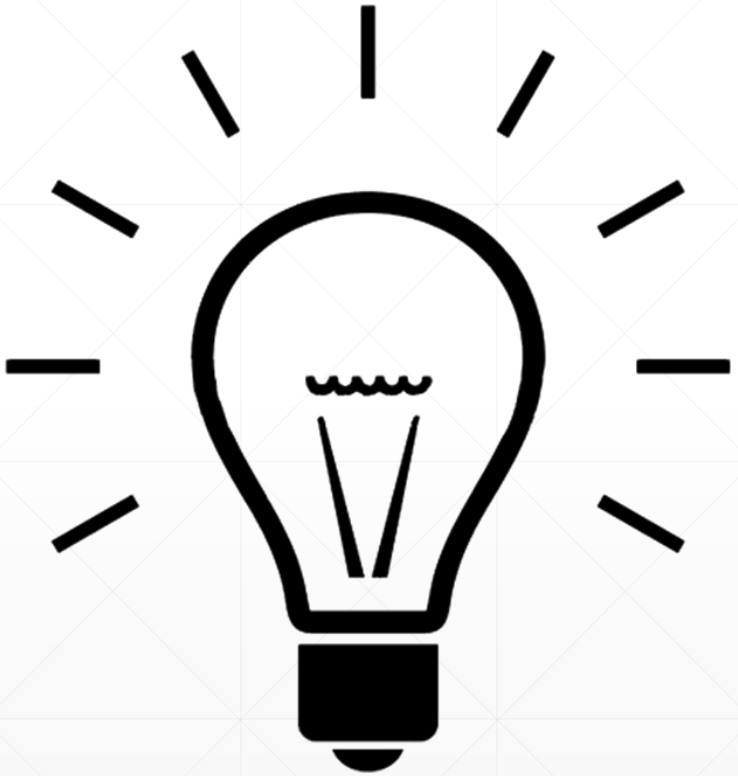
while

Zagram w ruletkę 10 razy. Po czym wstanę i odejdę.

```
while ( warunek ) {
        ...
}
```

```
var games = 0;

while ( games < 10 ) {
        playAgain();
        games = games + 1;
}
```

3a.while



do-while

```
doWork();
while ( condition ) {
        doWork();
}
```

do-while

```
doWork();

while ( condition ) {

        doWork();

}
```

===

```
do {

        doWork();

}
while ( condition )
```

do-while

Gwarantuje przynajmniej jedno wykonanie pętli.

Zagram w ruletkę. Będę grał tak długo dopóki nie będę miał mniej niż 100 zł.

```
do {
        ...
}
while ( condition )
```

```
var winnings;
do {
        playAgain();
}
while ( winnings < 100 )
```

for

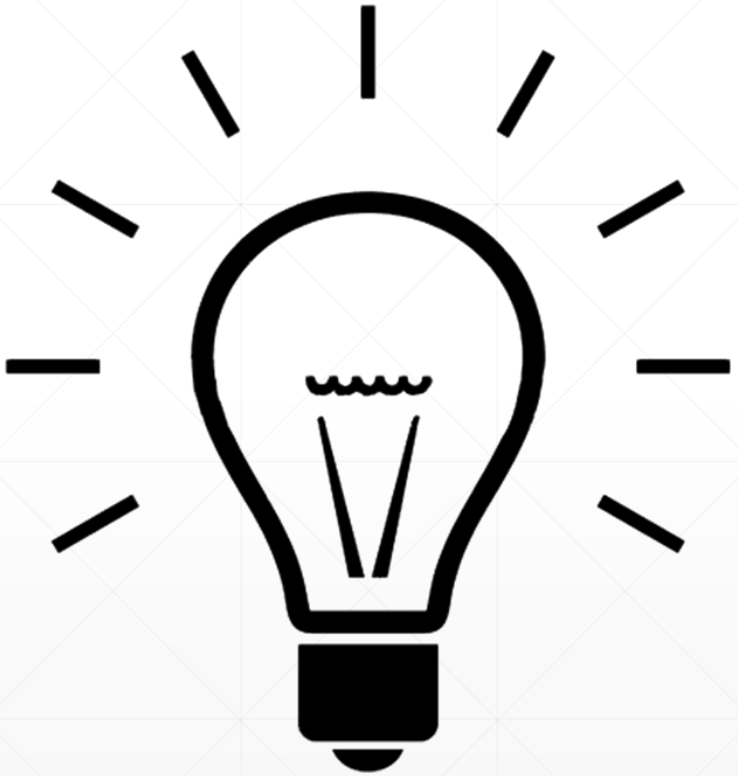
Wykonuje określoną liczbę powtórzeń

```
for ( inicjalizacja ; warunek ; zmiana ) {
        ...
}
```

```
for ( var i = 0 ; i < 6 ; i = i + 1 ) {
        console.log( i );
}
```

```
for ( inicjalizacja ; warunek ; zmiana ) {
        ...
}

for ( ; warunek ; zmiana ) {
        ...
}

for ( ; warunek ; ) {
        ...
}
```

3b.for



break

Wyjście z pętli

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

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

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

continue

Zakończenie bieżącego cyklu pętli

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

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

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

PODSTAWY PROGRAMOWANIA

Dziękuję za uwagę!

Bonus

- condition **?** doIfTrue() **:** doIfFalse()