**Lekcja 8**

# Temat: Funkcje2

## Zasięg zmiennych

Przy omawianiu funkcji istotny jest **zasięg zmiennych** (ang. *variables scope*). Zasięg możemy określić jako miejsca, w których zmienna jest ważna i można się do niej bezpośrednio odwoływać. W takim kontekście zmienna może być:

- globalna,
- lokalna.

**Zmienne globalne** (o zasięgu globalnym, ang. *global variables*) to takie, które są widoczne w każdym miejscu skryptu. Zmienna staje się globalną, jeśli jest **zdefiniowana poza wnętrzami funkcji lub też, uwaga!, jeżeli zostanie utworzona bez użycia słowa var**. Można się do niej odwoływać w dowolnym miejscu kodu, również we wnętrzach funkcji. Jest ona dostępna dla dowolnego innego kodu w bieżącym dokumencie

**Przykład 1**. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
var glob = 124; // zmienna globalna utworzona poza funkcją
    function func()
    {
      document.write("Wartość zmiennej w funkcji: " + glob);
    }
    func();
    document.write("<br>Wartość zmiennej poza funkcją: " + glob);
    /* wartość zmiennej zostanie wyświetlona dwukrotnie:
    pierwszy raz dzięki wywołaniu funkcji i drugi raz
    – dzięki występującej poza funkcją instrukcji document.write.
    */
</script>
```

Zmienne o zasięgu lokalnym, w skrócie — **zmienne lokalne** (ang. *local variables*), są definiowane wewnątrz funkcji, a ich zasięg jest ograniczony tylko do wnętrza funkcji, w której zostały zdefiniowane.

Próba odwołania w innym miejscu skryptu spowoduje powstanie błędu.

**Przykład 2**. Przygotuj stronę html wykorzystującą poniższy kod. Sprawdź działanie skryptów w konsoli.

```
<script>
    function func()
    {
        var local = 124; // zmienna lokalna utworzona wewnątrz funkcji
      document.write("Wartość zmiennej w funkcji: " + local);
    }
    func();
    document.write("<br>Wartość zmiennej poza funkcją: " + local);
</script>
```

```
<p id="output"></p>
<p id="output2"></p>
<script>
    const output = document.getElementById("output");
    function func()
    {
        let local = 124; // zmienna lokalna utworzona wewnątrz funkcji
        document.getElementById("output2").innerHTML=`Wartość zmiennej w funkcji: ${local}`;
    }
    func();
    console.log(output.innerHTML);
    output.innerHTML =`Wartość zmiennej poza funkcją: ${local}`;
</script>
```

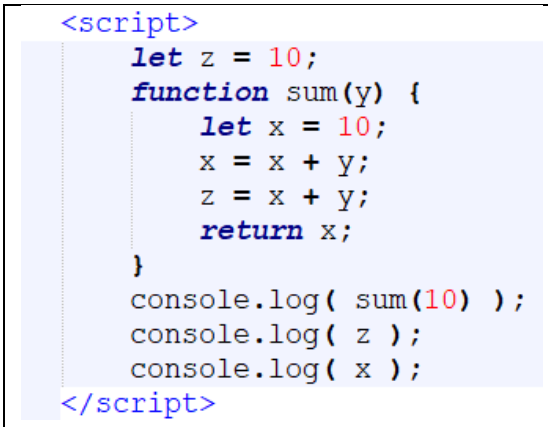
**Przykład 3**. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    var x = 10;

    function sum(y) {
        x = x + y;
        return x;
    }
    console.log( sum(10) ); //wypisze 20
    console.log( x ); //wypisze 20
</script>
```

Funkcja sum() nie tylko dokonuje wyliczenia sumy liczby, ale także **modyfikuje zmienną globalną x**. Nie zawsze chcemy, aby tak było.

Pisząc kod funkcji powinniśmy się starać, by była ona zamkniętym bytem. Podajemy do niej jakieś parametry, funkcja to przetwarza za pomocą swojego kodu i zmiennych,

| | |
|---|---|
|  | po czym kończy swoje działanie zwracając jakiś wynik. Stosując zmienne globalne nie jesteśmy w stanie tego uzyskać, bo każdorazowo zmieniamy zmienne spoza funkcji.<br>Problem ten rozwiązuje najnowsza wersja JS - ECMAScript wprowadzając deklarację zmiennych przy pomocy ***let*** |

Dzięki stosowaniu zmiennych lokalnych nie tylko unikamy zmian w zmiennych globalnych, ale także unikamy problemu z nazewnictwem zmiennych. Wyobraź sobie, że twój skrypt używał by wielu zmiennych i tyle samo funkcji. Za każdym razem musiał byś się zastanawiać jaką nową nazwę wymyślać, by nie nadpisać poprzednich zmiennych. Przy zmiennych lokalnych ten problem nie istnieje.

JS niestety ma kilka „dziwactw” związanych z przetwarzaniem zmiennych. Szukanie przyczyn niespodziewanego działania kodu może być mało ciekawe.

## Przesłanianie zmiennych

Jeżeli zmienna lokalna zdefiniowana wewnątrz funkcji lub parametr funkcji **ma taką samą nazwę** jak zmienna globalna, to zmienna

globalna jest przesłonięta wewnątrz funkcji przez zmienną lokalną lub parametr i zmiany dokonane na zmiennej lokalnej wewnątrz funkcji nie powodują zmiany zmiennej globalnej. Takie zjawisko nazywane jest **przesłanianiem zmiennej**.

**Przykład 4**. Przygotuj strony html wykorzystujące poniższe kody.

```
<script>
    var x = 10;

    function sum() {
        document.write( "Zmienna x wewnątrz funkcji przed zmianą: "+x+"<br>");
        var x = 15; // przysłonięcie zmiennej globalnej, zmienna lokalna o tej samej nazwie
        x = x + 1;
        document.write( "Zmienna x wewnątrz funkcji po zmianie: "+x+"<br>");
        return x;
    }
    document.write("Wywołanie funkcji: "+sum()+"<br>");
    document.write( "Zmienna x na zewnątrz funkcji "+x+"<br>");

</script>
```

```
<script>
    var x = 10;
    function sum(x) {
        // przesłonięcie zmiennej globalnej przez parametr
        document.write( "Zmienna x wewnątrz funkcji przed zmianą: "+x+"<br>");
        x = x + 1;
        document.write( "Zmienna x wewnątrz funkcji po zmianie: "+x+"<br>");
        return x;
    }
    document.write("Wywołanie funkcji: "+sum(5)+"<br>");
    document.write( "Zmienna x na zewnątrz funkcji "+x+"<br>");
    document.write( "<h3>Drugie wywołanie funkcji z argumentem będącym zmienna globalną</h3>");
    document.write("Wywołanie funkcji ze zmienną x: "+sum(x)+"<br>");

</script>
```

## Funkcje zagnieżdżone (wewnętrzne)

Wewnątrz jednej funkcji można umieścić definicję kolejnej. Powstaje w ten sposób funkcja zagnieżdżona . Schemat definicji funkcji wewnątrz innej funkcji jest następujący:

```
function funkcja_zewnętrzna(argumenty)
{
  function funkcja_wewnętrzna(argumenty)
  {
    // instrukcje funkcji wewnętrznej
```

```
    }
    // instrukcje funkcji zewnętrznej
}
```

Funkcja wewnętrzna ma dostęp do parametrów i zmiennych funkcji zewnętrznej.

**Przykład 5**. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    function obliczenia(argument1, argument2)
    {
      function kwadrat(argument)
      {
        return argument * argument;
      }
      return kwadrat(argument1) + kwadrat(argument2);
    }
    var number1=3;
    var number2=5;
    var wynik = obliczenia(number1, number2);
    document.write("Suma kwadratów " + number1 +" i "+number2+" to " + wynik);

</script>
```

```
<p id="output"></p>
<script>
    function calc(arg1, arg2)
    {
      function square(arg)
      {
        return arg * arg;
      }
      return square(arg1) + square(arg2);
    }
    const out=document.getElementById("output");
    let number1=3;
    let number2=5;
    let result = calc(number1, number2);
    out.innerHTML=`Suma kwadratów ${number1} i ${number2} to  ${result}`;

</script>
```

**Przykład 6**. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    // wszystkie obliczenia  wykonywane w funkcji wewnętrznej
    function obliczenia(argument1, argument2)
    {
      function suma_kwadratow()
      {
        return argument1 * argument1 + argument2 * argument2;
        /*
        funkcja wewnętrzna ma pełny dostęp do argumentów
        i zmiennych funkcji zewnętrznej.
        */
      }
      return suma_kwadratow();
    }
    var wynik = obliczenia(2, 4);
    document.write("Suma kwadratów 2 i 4 to " + wynik);
</script>
```

**Funkcja może być rekurencyjna  tj. wywoływać samą siebie**.
Przykładem może być funkcja licząca silnię:

**Przykład 7**. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    function silnia(n)
    {
        if ((n == 0) || (n == 1))
            return 1
        else {
            var result = (n * silnia(n-1) );
            return result
        }
    }
    document.write("1!= "+silnia(1)+"<br>");
    document.write("2!= "+silnia(2)+"<br>");
    document.write("3!= "+silnia(3)+"<br>");
    document.write("4!= "+silnia(4)+"<br>");
    document.write("5!= "+silnia(5)+"<br>");
</script>
```