Więcej o funkcjach

Przenoszenie instrukcji - hoisting

- Silnik JavaScriptu wykonuje instrukcje krok po kroku, zaczyna od pierwszej linii, kończy na ostatniej.
- Zdarza się jednak, że czasem przenosi pewne instrukcje na samą górę.
- Zjawisko to ma swoją nazwę - jest to hoisting.

```
/* hoisting */
function zrobHerbate() {
    /* ciało funkcji */ }
/* hoisting */
zrobHerbate(); /* ok */
function zrobHerbate() {
    console.log("Nalej wodę do czajnika");
    console.log("Wsyp do szklanki herbatę");
    console.log("Zagotuj wodę");
    console.log("Zalej herbatę");
}
```

Przenoszenie instrukcji - hoisting

Wyrażenia funkcyjne nie są przenoszone - przenoszone są tylko deklaracje zmiennych i deklaracje funkcji.

```
/* hoisting */
var herbata;
/* hoisting */
herbata(); /* błąd */
var herbata = function zrobHerbate() {
    console.log("Nalej wodę do czajnika");
    console.log("Wsyp do szklanki herbatę");
    console.log("Zagotuj wodę");
    console.log("Zalej herbatę");
}
```

W przypadku wyrażenia funkcyjnego musimy pamiętać, gdzie wywołujemy funkcję.

Funkcje wyższego rzędu

- Bardzo często zdarza się, że do jakiejś funkcji przekazujemy drugą funkcję jako argument.
- Może też wystąpić sytuacja, gdy jedna funkcja zwraca inną funkcję.
- Takie funkcje nazywamy funkcjami wyższego rzędu (higher-order functions).

Zmienne lokalne i globalne

W JavaScripcie występują dwa typy zmiennych:

globalne – są to zmienne, które nie są zadeklarowane w żadnym zakresie (na razie możecie uznać, że zakres to funkcja) albo są zadeklarowane bez słowa kluczowego **var/let**,

lokalne – są to zmienne zadeklarowane w środku jakiejś funkcji przy pomocy słowa kluczowego **var/let**.

Zmienne globalne są bardzo niebezpieczne i nie powinno się ich używać w funkcjach!

Zmienne globalne są widoczne w całym naszym programie i są niszczone dopiero podczas zamknięcia okna przeglądarki.

Zmienne lokalne są widoczne tylko i wyłącznie w zakresie funkcji, w której zostały stworzone. Są niszczone w chwili, w której ta funkcja się kończy.

Zmienne lokalne i globalne

```
function sayGlobalName() {
    console.log(name);
}
f
unction sayLocalName() {
    var name = "Janek"; /* Zmienna lokalna */
    console.log(name);
}
v
ar name = "Adam"; /* Zmienna globalna name */
sayGlobalName(); /* Adam */
sayLocalName(); /* Janek */
console.log(name); /* Adam */
```

W funkcji sayGlobalName() nie ma zmiennej name. Brana jest pod uwagę zmienna globalna.

Call stack

- Call stack to tak zwany stos wywołań. Trzyma on dokładne informacje na temat tego, w którym miejscu znajduje się nasz program w danej chwili.
- Funkcja, która jest właśnie uruchomiona, jest na samej górze tego stosu. Kiedy komputer natrafia na słowo kluczowe return, usuwa najwyższą funkcje.
- Kiedy wywoływana jest jakaś funkcja, cały stan (tak zwany kontekst) zawierający wartości zmiennych musi zostać zapamiętany.
- Kiedy funkcja się kończy, wszystkie jej zmienne są zapominane, a program „wczytuje” ze stosu informacje potrzebne do przywrócenia programu w miejscu, w którym funkcja została uruchomiona.
- Proces ten zajmuje zarówno pamięć, jak i czas procesora, powinno się zatem unikać wielokrotnych zagnieżdżeń funkcji.

Call stack

```
function foo() {
    var i = 0;
    var j = bar(i);
    return j;
}

function bar(i) {
    return i + 5;
}

function basic() {
    var j = foo();
    return j;
}
basic();

Call stack:
JavaScript context
basic
foo
bar
bar
foo
basic
JavaScript context
```

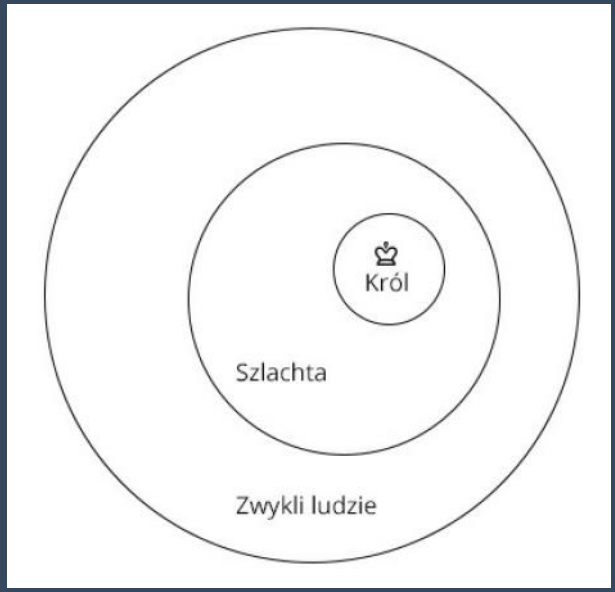
Zakresy zagnieżdżone

- Zmienne lokalne mają zakres, w którym są widoczne (variable scope).
- Do tej pory omówiliśmy zakres globalny (zmienne widoczne wszędzie) i lokalny (zmienne widoczne tylko w danej funkcji).
- W języku JavaScript istnieje wiele poziomów zakresu lokalnego. Zjawisko to określa się poprzez termin zakresy zagnieżdżone (nested scopes).

- Zagnieżdżanie zakresów polega na tym, że jeżeli w jakiejś funkcji zdefiniujemy drugą funkcje, to wtedy tworzy ona własny zakres lokalny.
- Funkcja zagnieżdżona ma jednak dostęp do wszystkich zmiennych lokalnych zakresów, w których jest osadzona.
- Wartości tych zmiennych są brane pod uwagę w chwili użycia funkcji zagnieżdżonej.

Zakresy zagnieżdżone

- Na obrazku poniżej mamy drobny przykład jak można rozumieć zakresy w JavaScript.
- Król mieszka w zamku. Król może wychodzić poza teren zamku i spotykać się ze szlachtą. Król może również wychodzić poza dziedziniec – do miasta i spotykać zwykłych ludzi.
- Szlachta nie ma prawa wejścia na zamek, może natomiast wychodzić na miasto i spotykać zwykłych ludzi.
- Zwykli ludzie nie mogą wchodzić na teren dziedzińca, gdzie mieszka szlachta, ani tym bardziej na teren zamku, gdzie mieszka król. Mogą tylko spacerować po mieście.



Czas na zadania