var vs let vs const

Sposoby deklarowania zmiennych w JavaScript

PK, 2021/08/13

Brak przedrostka

- Niezależnie od tego, gdzie zadeklarujecie zmienną, będzie ona dostępna w całym kodzie (tzw. zmienna globalna)
- Nie robimy tak ;-)

Brak przedrostka

```
variableWithoutDeclaration = 'some value';

function defineAnotherVariable() {
    antotherVariableWithoutDeclaration = 'another value';
}

defineAnotherVariable();

// W tym miejscu dostępna jest zarówno zmienna variableWithoutDeclaration jak i antotherVariableWithoutDeclaration.
// antotherVariableWithoutDeclaration "wycieka" z funkcji
```

Przedrostki "var" i "let"

- Przed ES6 jedynym dostępnym sposobem deklarowania zmiennych lokalnych był "var"
- "var" ma kilka cech, które czynią go nieintuicyjnym
- "let" został wprowadzony jako lepsza alternatywa
- "var" nadal działa po staremu aby zapewnić wsteczną kompatybilność języka
- Nie używamy "var" w ogóle

"var" vs "let"

Zmienne zadeklarowane "var" można używać przed miejscem deklaracji i mają wtedy wartość undefined.

Zmienne zadeklarowane "let" są niedostępne przez miejscem deklaracji, a próba ich użycia skutkuje takim samym błędem, jak gdyby zmienna nie została zadeklarowana w ogóle.

```
// wyświetli undefined!
console.log(someVariable);

var someVariable = 'some value';

console.log(someVariable);
```

```
// fatal error, zmienna nie jest jeszcze dostępna
console.log(someVariable);

let someVariable = 'some value';

console.log(someVariable);
```

"var" vs "let"

Scope w którym dostępne są zmienne zadeklarowane przez "var" ograniczony jest najbliższą funkcją.

Scope w którym dostępne są zmienne zadeklarowane przez "let" ograniczony jest najbliższym blokiem (wąsem).

```
function someFunction() {
    if (someCondition) {
        var someVariable = 'some value';
    }

    // someVriable nieoczekiwanie dostępne jest równiez tutaj!
}

someFunction();

// someVariable jest ograniczone funkcją, więc tutaj nie jest
// juz dostepne
```

```
function someFunction() {
    if (someCondition) {
        let someVariable = 'some value';
    }

    // someVriable nie jest dostępne nigdzie poza blokiem if. Tu go nie ma.
}

someFunction();

// Tutaj tez go nie ma
```

"var" vs "let"

Uproszczony przypadek (ale z życia wzięty), w którym użycie "var" skutkuje nieprawidłowym działaniem programu.

Zamiarem programisty było wyświetlenie cyfr od 0 do 9.

```
// Wyświetli dziesięć razy liczbę 10
for (var i = 0; i < 10; i++) {
    setTimeout(
        function() { console.log(i); },
    );
}
```

```
// Wyświetli cyfry od 0 do 9
for (let i = 0; i < 10; i++) {
    setTimeout(
        function() { console.log(i); },
    );
}
```

Przedrostek "const"

- Zachowuje się tak samo jak "let"
- Przedrostek "const" nie definiuje stałej, to nadal zmienna!
- Zmienna zadeklarowana jako "const"może być niemal swobodnie modyfikowana.
- W stosunku do "let" różni się wyłącznie tym, iż referencja takiej zmiennej jest stała.
- Co oznacza powyższe zdanie, nauczymy się w dalszej części kursu.
- W praktyce zmiennych zadeklarowanych jako "const" nie można używać z operatorem przypisania ("=") oraz z operatorami unarnymi ("++", "--").

```
let someVariable = 10;

// operator przypisania zadziała
someVariable = 10;

// operatory unarne zadziałają
someVariable++;
someVariable--;
```

```
const someVariable = 10;

// operator przypisania zwróci błąd
someVariable = 10;

// operatory unarne zwrócą błąd
someVariable++;
someVariable--;
```

Do zobaczenia na kolejnym spotkaniu