

Języki programowania i GUI

Lista 2 - 2021

0. Ze strony <https://javascript.info/> przeczytaj rozdziały:

- <https://javascript.info/symbol>
- <https://javascript.info/object-toprimitive>
- <https://javascript.info/iterable>
- <https://javascript.info/destructuring-assignment>

a potem zrozum i wypróbuj poniższy kod:

```
function zakres(a,b){this.a=a;this.b=b}
zakres.prototype[Symbol.iterator]=
    function*(){
        for(let i=this.a;i<=this.b;i++) yield i;
    }
zakres.prototype[Symbol.toPrimitive]=
    function(hint){return hint=="number"?
        (this.a+this.b)*(this.b-this.a+1)/2:
        this.a+"..." +this.b;}

z=new zakres(10,15); console.log(z);
for(let x of z)
    console.log(x);
console.log("suma("+z+")="+ +z)
console.log(Array.from(z))
```

Następnie zapisz ten kod w postaci definicji klasy `zakres` i wypróbuj ponownie.

1. Zapisz funkcję gwiazdkową `Fibonacci()` zwracającą iterator na wszystkie liczby Fibonacciego zwracane jako typ `BigInt`, czyli z dowolnie dużą ilością cyfr. Dla przetestowania wywołaj 200 razy metodę `next()` wynikowego operatora, i wypisuj pole `value`. Napisz drugą wersję programu, która za pomocą `setInterval` lub `setTimeout` będzie uruchamiała `next()` co pół sekundy.
2. Napisz funkcję `Fibo()` – konstruktor oraz `Fibo.prototype.next=function(){...}` tak, aby obiekty `let z1=Fibonacci()` z zadania 1, oraz `let z2=new Fibo()` z tego zadania zachowywały się identycznie w trakcie testów z zadania 1.
3. Zapisz `function* fragment(iter,skip,limit=1)`, która zwróci iterator, który z argumentu `iter`, który też jest iteratorem, pobiera kolejne wartości za pomocą `for(of)`, ale pomija ilość = `skip` początkowych wartości i zwraca (przez `yield`) ilość = `limit` następnych, a potem kończy działanie. Zastosuj ją tak:
`for(let x of fragment(Fibonacci(),100,3)) console.log(x)`
4. Napisz `Array.prototype.wspak=function*(){...}`, która zwraca iterator idący po elementach tablicy od końca. Użyj pętli po indeksach tablicy oraz `yield`. Nie możesz użyć `Array.reverse()`. Zastosowanie: `for(let x of [2,3,4,5].wspak())`
5. Napisz funkcję `function BST(key,left,right)`, która zwraca węzeł drzewa BST o poddrzewach `left` i `right` i kluczu `key`. Zainicjuj `BST.prototype[Symbol.iterator]` taką funkcją gwiazdkową, by pętla `for(of)` dla drzew BST pokazywała ich klucze w porządku `in order`, czyli rosnącym. Aby zademonstrować działanie utwórz jednym poleceniem drzewo o co najmniej 7 kluczach na nie więcej niż 4 poziomach.
6. <https://javascript.info/destructuring-assignment#smart-function-parameters> pokazuje, jak pisać funkcje z dużą ilością argumentów.