**Exercises**

*Lecture 3:*

ES6 - part 2

1. **Default + rest + spread**
   1. Write a function which returns an array of characters of a string.

const test = () => {

let result = [“a”,”b”,”c”,”d”];

return rusult;

}

1.2. Given 2 arrays:

var array1 = [1, 2, 3];

var randomValue = 'something';

var array2 = [4, 5, 6];

Write a function which returns [1, 2, 3, 'something', 4, 5, 6].

*function merge(){*

*var array1 = [1, 2, 3];*

*var randomValue = 'something';*

*var array2 = [4, 5, 6];*

*return [...array1,"something",...array2];*

*}*

*console.log(merge());*

* 1. Write a function which returns sum of its arguments (number only).

*const sum = (function() {*

*"use strict";*

*return function sum(...args) {*

*return args.filter(a => Number.isInteger(a)).reduce((a, b) => a + b, 0);*

*};*

*})();*

*console.log(sum(1, 2,"Phat", 3,4,5,6,2,"Nguyen"));*

1.4. Write a function which returns sum of its arguments except for the first one (number only).

function sum\_1(...args){

args = args.splice(1,args.length);

return args.filter(a => Number.isInteger(a)).reduce((a, b) => a + b, 0);

}

console.log("sum1:"+sum\_1(5,1,2,3,4));

1.5. Write a function which returns sum of its arguments except for the last one (number only).

*function sum\_2(...args){*

*args = args.splice(0,args.length-1);*

*return args.filter(a => Number.isInteger(a)).reduce((a, b) => a + b, 0);*

*}*

*console.log(“sum2:”+sum\_2(1,2,3,4,5));*

1.6. Write a function which returns sum of its first 3 arguments (number only).

*function sum\_3(...args){*

*args = args.splice(0,3);*

*return args.filter(a => Number.isInteger(a)).reduce((a, b) => a + b, 0);*

*}*

*console.log("sum3:"+sum\_3(1,4,5,10));*

1. **Let + const**

Read some articles about let, const and var to have a better understanding about them.

Recommended articles:

* <https://medium.com/javascript-scene/javascript-es6-var-let-or-const-ba58b8dcde75>
* <https://ponyfoo.com/articles/var-let-const>
* <http://wesbos.com/let-vs-const/>

1. **Iterators + for .. of**

3.1. Create a countdown iterator that counts from 9 to 1. **Use iterator, do not use generator.**

let getCountdownIterator =

console.log( [ ...getCountdownIterator ] );

> [9, 8, 7, 6, 5, 4, 3, 2, 1]

3.2. Create an infinite sequence that generates the next value of the Fibonacci sequence.

The Fibonacci sequence is defined as follows:

fib( 0 ) = 0

fib( 1 ) = 1

for n > 1, fib( n ) = fib( n - 1 ) + fib( n - 2 )

*let fibonaci = {*

*[Symbol.iterator](){*

*let pre = 0;*

*let cur = 1;*

*return {*

*next(){*

*[pre,cur] = [cur,cur+pre];*

*return {done:false,value:cur}*

*}*

*}*

*}*

*}*

*for(let n of fibonaci){*

*console.log(n);*

*}*

1. **Generators**

4.1. Create a countdown iterator that counts from 9 to 1. Use **generator** functions!

let getCountdownIterator = function \*() {

let i = 10;

while( i > 1 ) {

yield --i;

}

}

console.log( [ ...getCountdownIterator() ] );

> [9, 8, 7, 6, 5, 4, 3, 2, 1]

4.2. Use **generator**, create an infinite sequence that generates the next value of the Fibonacci sequence.

The Fibonacci sequence is defined as follows:

fib( 0 ) = 0

fib( 1 ) = 1

for n > 1, fib( n ) = fib( n - 1 ) + fib( n - 2 )

*function \*fibonacci() {*

*let a = 0, b = 1;*

*yield a;*

*yield b;*

*while( true ) {*

*[a, b] = [b, a+b];*

*yield b;*

*}*

*}*

4.3. Create a lazy filter generator function. Filter the elements of the Fibonacci sequence by keeping the **even values only**.

function \*filter( iterable, filterFunction ) {

for( let element of iterable ) {

if ( filterFunction( element ) ) yield element;

}

}