 **Exercise – Closures and Callbacks  
  
Exercise (Closures - 1)  
  
Given the following code:**

function buildFunctions(){

var arr=[];

for (var i=0; i<3; i++){

arr.push(function() {console.log(i);})

}

return arr;

}

var fs = buildFunctions();

fs[0]();

fs[1]();

fs[2]();

a) What is the expected output? Make a bid and test the code in VSCode!.  
  
b) Probably you were a bit surprised (many did, at least!) – Explain the result (Hint: it’s the closures phenomenon).  
  
c) Change the code so the result becomes as expected (Hint: use *let* and a local variable).  
Explain why the code now works as intended ☺

**Exercise (Closures – 2, function factories)**

**Given the following code:**

function makeGreeting(language) {

return function(firstname, lastname){

if (language==='en'){console.log('Hello ' + firstname + ' ' + lastname)};

if (language==='dk'){console.log('Hej ' + firstname + ' ' + lastname)};

if (language==='es'){console.log('Hola ' + firstname + ' ' + lastname)}

}

}

var greetEnglish = makeGreeting('en');

var greetDanish = makeGreeting('dk');

var greetSpanish = makeGreeting('es');

greetEnglish('John', 'Doe');

greetDanish('Henrik', 'Høltzer');

greetSpanish('Pablo', 'Fuentes');

a) What is the expected output? Make a bid and test the code in VSCode.  
b) Explain, with the Closures phenomenon in mind, what happens in the code.  
c) Rewrite the anonymous function (our function expressions) so it uses the ES6 (Arrow-function).

**Exercise (Closures – 3, function factories)**

a) Make a new function *makeAdder(x)*, with a ”function factories” that returns a new function object (use function expressions). The returning anonymous function, must take a new argument and add with x.

b) Test your code with:

var add5 = makeAdder(5);

var add10 = makeAdder(10);

var addHello = makeAdder("Hello ");

console.log(add5(2)); // 7

console.log(add10(2)); // 12

console.log(addHello("Henrik")); // Hello Henrik

**Exercise (Callback function)**a) Try the following code in VSCode:

function tellMeWhenDone(callback){

// .... some work

callback();

}

tellMeWhenDone(function(){console.log("Im' done!")});

tellMeWhenDone(function(){console.log("All done!")});

b) Find yourself an example where you test the ”template” of a callback function ☺