# FL8. Lesson 9. Data types.

# Homework

1. The result of your work should be packed to **folder** with html and js files (separate js files for each task and one html file). Inner folder structure should be exactly as follows:

*|\_\_ homework\_9 /  
 |\_\_ index.html*

*|\_\_ js /*

*|\_\_forEach.js*

*|\_\_…*

*|\_\_decypherPhrase.js*

1. Code has to be readable, tested and well-formatted.
2. The folder must be loaded into the Github repository '**front-end-lab-8**' in the **master** branch

**Task #1**

Write a function - *forEach*

It should accept **array** and **function** as arguments and runs that function per each element in array. Also that function should be invoked with that array element.  
For example:

forEach([3, 5, 2], function(el) {

console.log(el);

}) // -> 3 5 2

**Task #2**

Write a function - *getTransformedArray*

It should accept **array** and **function** as arguments and returns new **array** with every element being transformed by that function  
For example:

function increment(num) {

return num + 1;

} // just returns incremented number  
getTransformedArray([1, 7, 20], increment); // -> [2, 8, 21]

**Note**: you should use your *forEach* function. Treat it as just an easy way to write your ordinary **‘for’** iterator, and the function you pass to it as just a body of that iterator.

**Task #3**

Write a function - *getFilteredArray*

It should accept **array** and **function** (called *predicate* function) as arguments and returns a new filtered **array**. A predicate function accepts an item in the array, and returns a boolean indicating whether the item should be retained in the new array.  
For example:

function predicateFunction(num) {

return num > 3;

} // returns boolean value  
getFilteredArray([1, 7, 20], predicateFunction); // -> [7, 20]

**Note**: reuse your *forEach* function

**Task #4**

You have an array of movies:

var movies = [

{

"id": 70111470,

"title": "Die Hard",

"boxart": " http://bb-1/654356453",

"uri": " http://bb-1/654356453",

"rating": 2.7,

"bookmark": []

}, {

"id": 654356453,

"title": "Good Mood",

"boxart": "http://bb-1",

"uri": " http://bb-1/654356453",

"rating": 3.5,

"bookmark": [{time: 65876586}]

}, {

"id": 65432445,

"title": "The Chamber",

"boxart": "http://ch-1",

"uri": " http://ch-1/65432445",

"rating": 3.0,

"bookmark": []

}, {

"id": 675465,

"title": "Fracture",

"boxart": "http://fr-1",

"uri": " http://fr-1/675465",

"rating": 4.0,

"bookmark": [{time: 65876586}]

}];

Write a function – *collectIds*

It should accept an array of movies and return the ids of videos that have a rating higher than 3.0

For example: collectIds(movies) // -> [654356453, 675465]

**Note**: reuse your *getFilteredArray* and *getTransformedArray* functions

**Task #5**

Write a function – *cypherPhrase*

It takes **object** and **string** and returns **string** transformed by object rules.  
Tip: (falsy || ‘truthy’) returns ‘truthy’. e.g. var obj = { }; var letter = obj[‘a’] || ‘a’; // -> ‘a’, because obj[‘a’] would return ‘undefined’ which is falsy.  
Tip: you can treat strings as just array of characters.  
For example:  
var charactersMap = {a: ‘o’, c: ‘d’, t: ‘g’}  
cypherPhrase(charactersMap, ‘kitty cat’ ); // -> “kiggy dog”

**Note**: reuse your *getTransformedArray* function, don’t forget about [Array.prototype.join](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Array/join) to transfrom array back into string.

**Task #6**

Write a function – *decypherPhrase*

It takes **object** and **string** cyphered with *cypherPhrase* function and returns **string** back to normal.  
Tip: easiest way to solve this task is to reverse key-value in charactersMap  
Invocation example:  
var charactersMap = {a: ‘o’, c: ‘d’, t: ‘g’}  
decypherPhrase(charactersMap, ‘kiggy dog’); // -> “kitty cat”

**Note**: reuse your cypherPhrase function