



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA
Laboratorio de Lenguajes de programación

CARRERA	CURSO	AMBIENTE
Ingeniería de Sistemas	Lenguajes de programación	305

PRACTICA No	NOMBRE DE PRÁCTICA	CODIGO DE LAB.	DURACION (HORAS)
09	Programación funcional Java script	305	2

REVISION	FECHA	DESCRIPCIÓN
1	21/05/2019	Revisión de Guías de Laboratorio

1. OBJETIVOS

- Dar a conocer al estudiante de conceptos básicos programación funcional

2. TEMAS A TRATAR

- Programación funcional con Java script

3. MATERIALES, EQUIPOS, SOFTWARE



- Ordenador
- Navegador google chrome o mozilla
- Editor de texto Sublime o Notepad ++.

4. CONCEPTOS

MAP, REDUCE, FILTER and ARROW FUNCTIONS

Extraído de la documentación de javascript (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>)

MAP

map calls a provided callback function once for each element in an array, in order, and constructs a new array from the results. callback is invoked only for indexes of the array which have assigned values, including undefined. It is not called for missing elements of the array (that is, indexes that have never been set, which have been deleted or which have never been assigned a value). Example:

```
1 var array1 = [1, 4, 9, 16];
2
3 // pass a function to map
4 const map1 = array1.map(x => x * 2);
5
6 console.log(map1);
7 // expected output: Array [2, 8, 18, 32]
```

ARROW FUNCTIONS

An arrow function expression is a syntactically compact alternative to a regular function expression, although without its own bindings to the this, arguments, super, or new.target keywords. Arrow function expressions are ill suited as methods, and they cannot be used as constructors. Examples:



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Laboratorio de Lenguajes de programación

```
1 var materials = [  
2   'Hydrogen',  
3   'Helium',  
4   'Lithium',  
5   'Beryllium'  
6 ];  
7  
8 console.log(materials.map(material => material.length));  
9 // expected output: Array [8, 6, 7, 9]  
10
```

```
(param1, param2, ..., paramN) => { statements }  
(param1, param2, ..., paramN) => expression  
// equivalent to: => { return expression; }
```

```
// Parentheses are optional when there's only one parameter name:  
(singleParam) => { statements }  
singleParam => { statements }
```

```
// The parameter list for a function with no parameters should be written with a pair of parentheses.  
() => { statements }
```

```
// Parenthesize the body of function to return an object literal expression:  
params => ({foo: bar})
```

```
// Rest parameters and default parameters are supported  
(param1, param2, ...rest) => { statements }  
(param1 = defaultValue1, param2, ..., paramN = defaultValueN) => {  
statements }
```

```
// Destructuring within the parameter list is also supported  
var f = ([a, b] = [1, 2], {x: c} = {x: a + b}) => a + b + c;  
f(); // 6
```



FILTER

The filter() method creates a new array with all elements that pass the test implemented by the provided function. Examples:

```
1 var words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];
2
3 const result = words.filter(word => word.length > 6);
4
5 console.log(result);
6 // expected output: Array ["exuberant", "destruction", "present"]
7
```

REDUCE

The reduce() method executes the callback once for each assigned value present in the array, taking four arguments:

- accumulator
- currentValue
- currentIndex
- array

accumulator

The accumulator accumulates the callback's return values. It is the accumulated value previously returned in the last invocation of the callback, or initialValue, if supplied (see below).

currentValue

The current element being processed in the array.

currentIndex Optional



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Laboratorio de Lenguajes de programación

The index of the current element being processed in the array. Starts from index 0 if an initialValue is provided. Otherwise, starts from index 1.

array Optional

The array reduce() was called upon.

The first time the callback is called, accumulator and currentValue can be one of two values. If initialValue is provided in the call to reduce(), then accumulator will be equal to initialValue, and currentValue will be equal to the first value in the array. If no initialValue is provided, then accumulator will be equal to the first value in the array, and currentValue will be equal to the second.

Examples:

```
1 const array1 = [1, 2, 3, 4];
2 const reducer = (accumulator, currentValue) => accumulator + currentValue;
3
4 // 1 + 2 + 3 + 4
5 console.log(array1.reduce(reducer));
6 // expected output: 10
7
8 // 5 + 1 + 2 + 3 + 4
9 console.log(array1.reduce(reducer, 5));
10 // expected output: 15
11
```

5. PROBLEMAS PROPUESTOS

1. Convert this array, from Celcius to Fahrenheit. Use map or filter or reduce.



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA
Laboratorio de Languages de programación

```
const celsius = [-15, -5, 0, 10, 16, 20, 24, 32];
```

2. Display only the users with 'l' in their name. Use map or filter or reduce.

```
const users = [
  { id: 11, name: 'Adam', age: 23, group: 'editor' },
  { id: 47, name: 'John', age: 28, group: 'admin' },
  { id: 85, name: 'William', age: 34, group: 'editor' },
  { id: 97, name: 'Oliver', age: 28, group: 'admin' }
];
```

3. For the last exercise, checks if any user have admin privileges, the program should show true or false. Use map or filter or reduce or some.
4. Show the frequency vector for group property, for example the result is: {editor: 2, admin: 2}. Use map or filter or reduce or some.
5. Show the average age of dogs. Use map or filter or reduce or some.

```
arr = [
  {name:'luna', sex:'f', age:7, species:'dog'},
  {name:'jimmy', sex:'m', age:122, species:'human'},
  {name:'snoop', sex:'m', age:60, species:'human'},
  {name:'jennifer', sex:'f', age:250, species:'human'},
  {name:'yeller', sex:'20', age:20, species:'dog'},
];
```