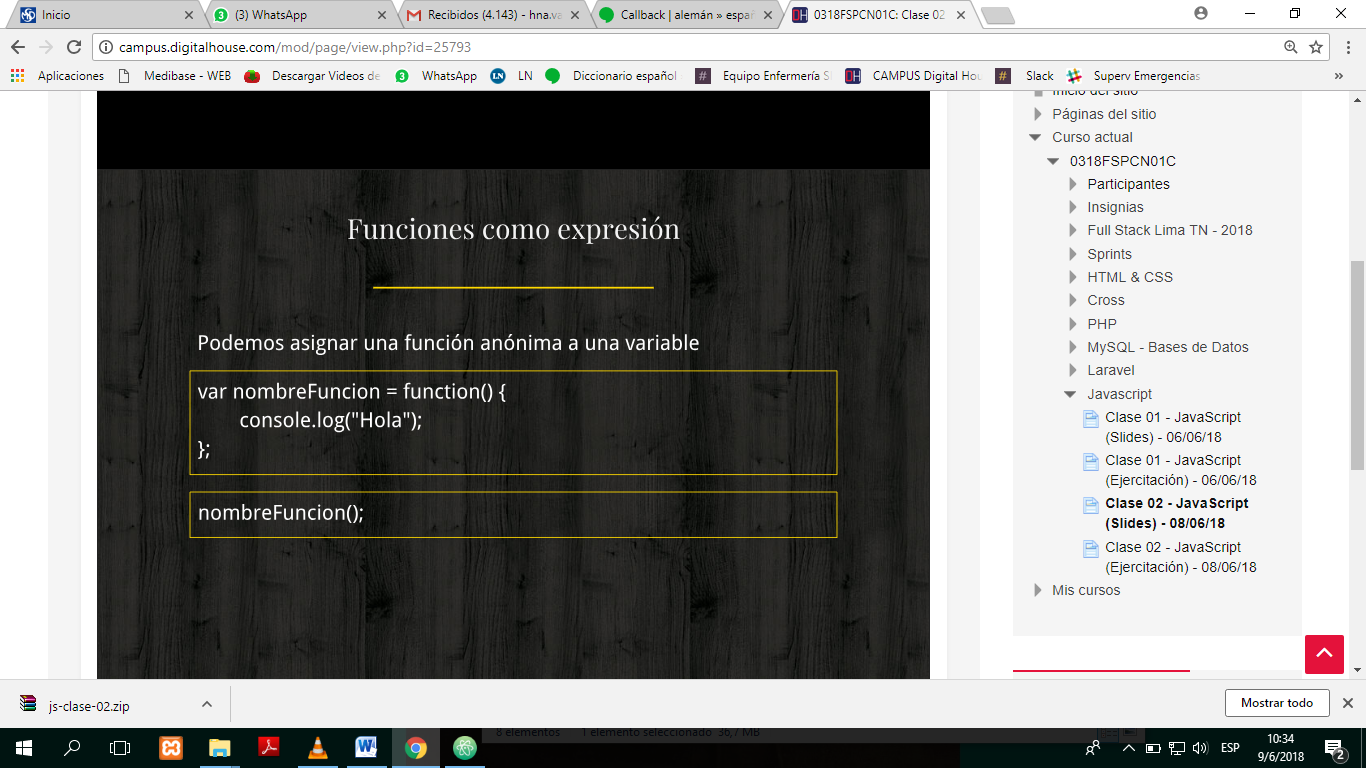
Clase 2

**FUNCION ANONIMA o func por expresión: una func que vive dentro de una variable.** La función persé **NO** tiene nombre



**SCOPE (alcance) que tiene nuestro código. Una variable puede tener un SCOPE local**

**HOISTING: Es lo que hace JAVA para poder setear en su memoria las variables en función de no arrojar errores. Trabaja con precedencia**, o sea ejecuta aquellas operaciones que son más importantes q otras**. La precedencia es la operación que se resuelve primero.**

**CLOSURE o función ANIDADA o funcionalidad:** Es una función que vive dentro de otra función. El clousure **NO EXISTE POR FUERA DE LA FUNCIÓN**. Nunca espera que le pase un parámetro porque toma el parámetro del elemento Padre.

**PROCEDURAL: se ejecuta primero una func y dsps otra.**

Un OBJETO LITERAL es un TIPO DE DATO.

FUNCIONES:

**F.ANONIMA O FUNCION POR EXPRESIÓN**: es aquella función que está guardada en una function DEFINIDA por VAR, **una func que vive dentro de una variable.** La función persé **NO** tiene nombre

**HOISTING:** es lo que hace JAVA y tiene que ver con el SCOPE. (audio 8.30) **Es lo que hace JAVA para poder setear en su memoria las variables en función de no arrojar errores.** Eso quiere decir que es procedural. **Trabaja con precedencia**, de aquellas operaciones que son más importantes q otras**. La precedencia es la operación que se resuelve primero**. Por ejemplo si llamo a una función sin definirla, JAVA la hace funcionar sin problema.. Lleva todas las funciones en memoria para arriba, lo organiza por nosotros. Por ej.llamo a una variable global antes de definirla y no va a pinchar ☺

**PRECEDENCIA**: operac matemáticas más importantes que otras, es la operación que se resuelve primero!!!!!

**SCOPE o alcance**: (audio 6.41)

Scope /Alcance local: variable que vive dentro de una función. No puedo hacer console.log de esta variable xq no existe.

Alcance global: cuando la variable está afuera de la función

Null: significa que no tiene valor alguno.

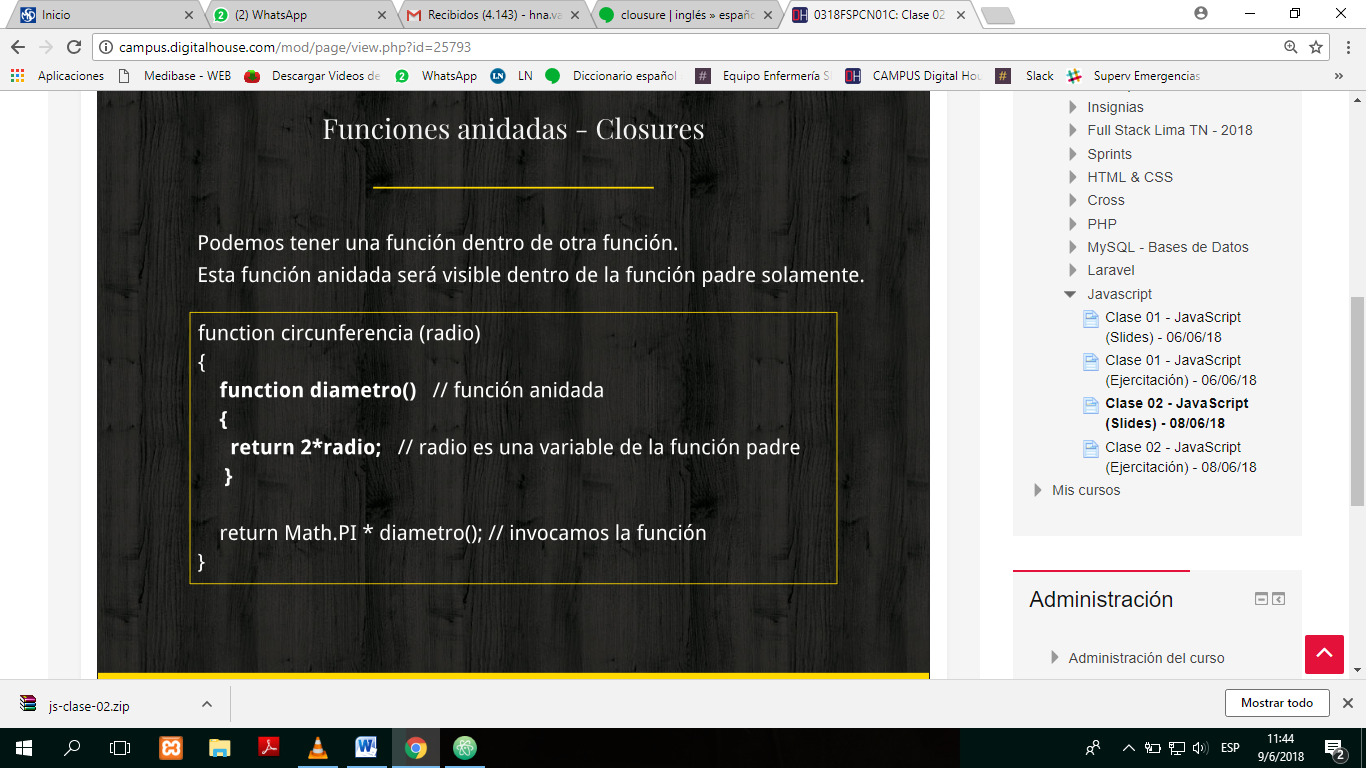
Undefined: significa que no está definida.

**CLOSURES (cierre-clausura) o función ANIDADA** O funcionalidad**:** audio 11.50

Es una función que vive dentro de otra función.

El clousure **NO EXISTE POR FUERA DE LA FUNCIÓN**. En el ejemplo la función diámetro es el closure.

En el ejemplo: no puedo usar la función diámetro solita xq pincharía, dará error. La func diámetro existe sólo dentro de circunferencia.



Nunca espera que le pase un parámetro porque toma el parámetro del elemento Padre; en este caso el elemento padre es la función circunferencia.

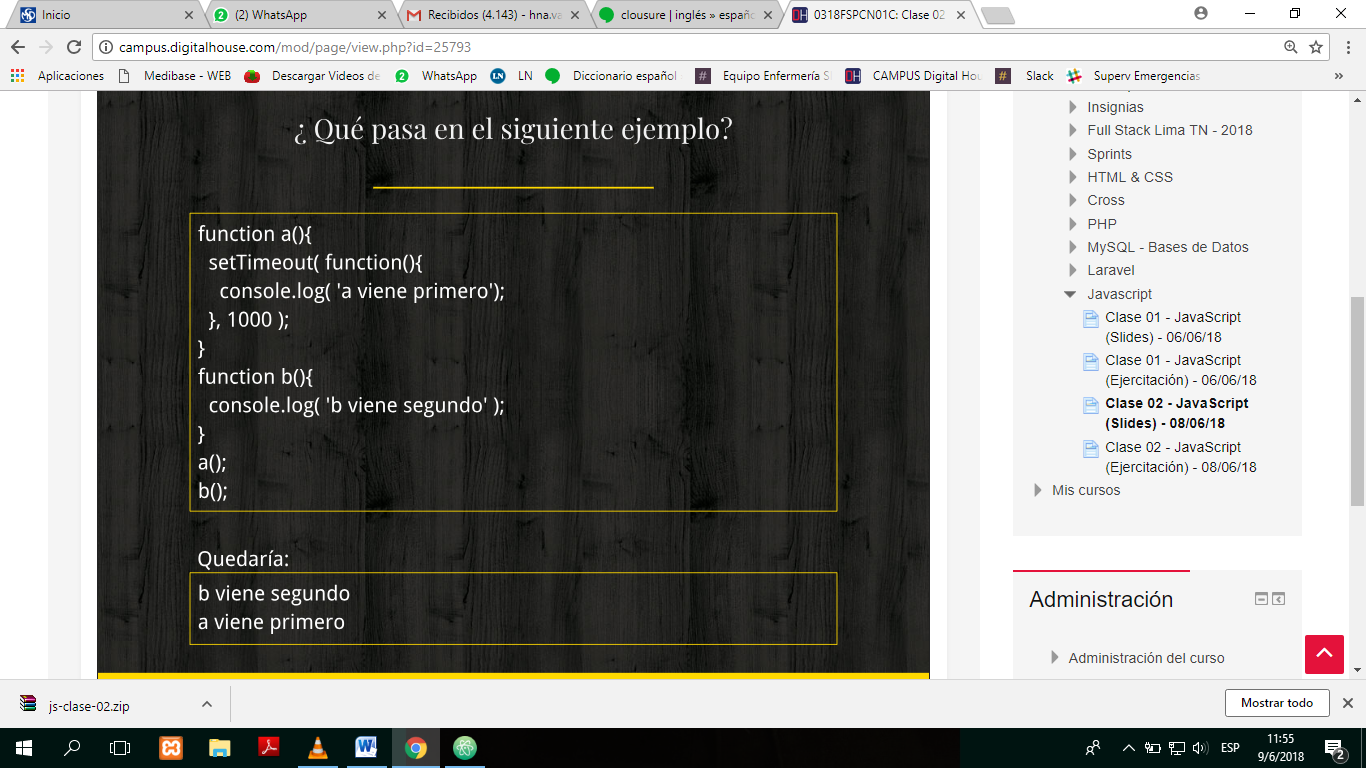
Las funciones siempre DEVUELVEN algo, un **return.** No olvidarse de escribirlo, más allá del console.log. (audio 15.30)

**SET TIME OUT** (función nativa de un método nativo de JAVA S)

Ejecuta un bloque de código dentro o desps de una determinada cantidad de tiempo, se le indica los milisegundos que debe esperar.

Como 1°parámetro recibe una función, la función recibe un bloque de ejecución.

Como 2° parámetro recibe una cantidad de milisegundos q debe esperar para ejecutar esa función.



Primer parámetro

Segundo parámetro

JAVA llama a las funciones, (a) se **ejecutó** pero como tiene un SET TIME OUT, la muestra después xq está esperando.

Esta función de SET TIME OUT se usa mucho en un homebanking. Es la función que se usa en los cajeros y te pregunta ¿estás ahí? Si no estás ahí te cierra la sesión.

**CALLBACKS (rellamada)**

Audio: 21.30

Es una función que se ejecuta adentro de otra función, pero que yo paso esa función como parámetro. Es una función que se define **por FUERA** de la función ppal, pero que se pasa como parámetro a esa función. Y en su momento se va a ejecutar. ¿Cuándo? cuando la función padre lo defina.

Pero que NO es un CLOSURE, xq toda la definición del closure está dentro de su elemento padre.

Siempre pasarle una FUNCTION.

En un archivo HTML debo escribir <script> …aquí va el código JAVA … </script>

Función saludar recibe un callback

function saludar (callback) {

setTimeout **(** function () **{**

console.log('Hola. Cómo están?');

Ése callback se ejecuta después de este console log

callback();**}**, 1000**)**;

}

function despedirse () {

console.log('Ok, adiós!');

**¿Cuál es la función padre?**

}

function queDiaEs () {

console.log('Es viernes');

}

**EN LA CONSOLA ESCRIBO la función saludar y como parámetro la función despedirse:**

saludar(despedirse);

saludar(queDiaEs);

nunca pasarle paréntesis () aquí:

**INTERPOLACIÓN de variables:** va **${…}**

Se usa para concatenar. Va la comilla francesa  **`…`** y adentro va **${…} ej: ${ing1}**

Es el **template string,** para que funcione necesita la **comilla inclinada. Ej.:**

**console.log(`**Estoy comiendo un sambuche de: **${ing1} y ${ing2}** por **${potencia(4, 6)}** vez!**`)**

inclusive dentro de las llaves puedo escribir expresiones como funciones!!!!!!

Interpolar: es generar un HTML falso, en donde lo que está dentro de las llaves se entiende que es una variable.

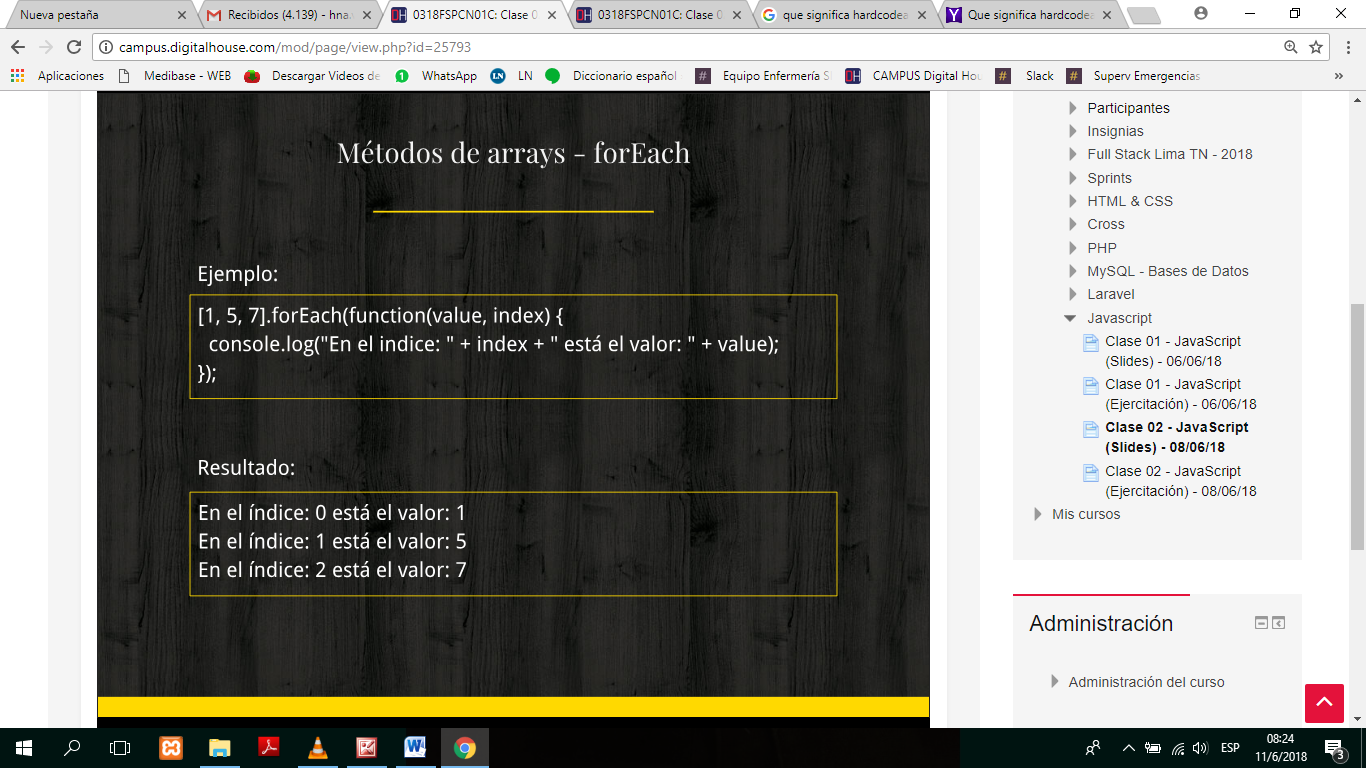
**ARRAY**

*Métodos para recorrer Arrays*

**ForEach**: es un método!!!!! De un array. Porque va con la expresión del punto. El ForEach recibe un array que puede ser harcodeado[[1]](#endnote-1) o bien guardado en una variable. ForEach recibe un CALLBACK q’ es una función, esa función recibe 2 parámetros o argumentos:

1°parámetro: el valor de c/u de los elementos del array. **Es obligatorio pasar este dato.**

2°p: índice de c/u de los elementos de ese array.



/ LET: Es una variable de bloque. Eso signfica q' ciudades sólo existe en el Scope de esta función - la que está en el forEach- y la otra ciudades sólo existe en en el Scope de donde me encuentro, en este caso en Windows. Let es una variable que se puede cambiar.

let ciudades = ['Buenos Aires', 'Bogotá', 'Rio de Janerio'];

ciudades**.forEach**(function (ciudad, i) {

console.log(`Estamos en la ciudad ${ciudad} - ${i + 1}`);

});

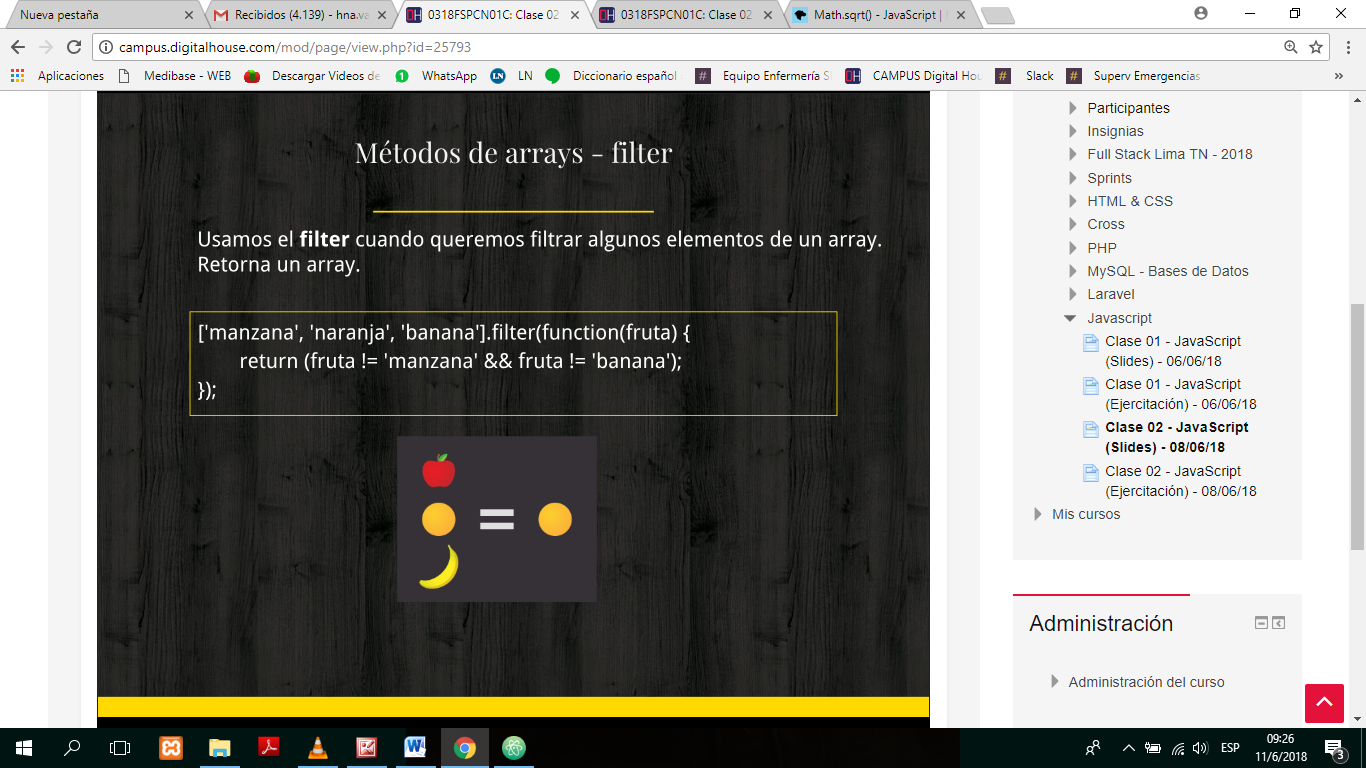
**Map**:genera un array nuevo pero cada elemento del array me lo puede devolver modificado



* Itera un array, recorre uno por uno sus elementos
* Tb recibe un callback, o sea una función, a la que le paso un parámetro
* Me devuelve un array
* Hace algo con cada uno de los elementos
* Tb se le puede pasar el índice (la manzana está en la posicipon 1: i + 1 y así sucesiva//)

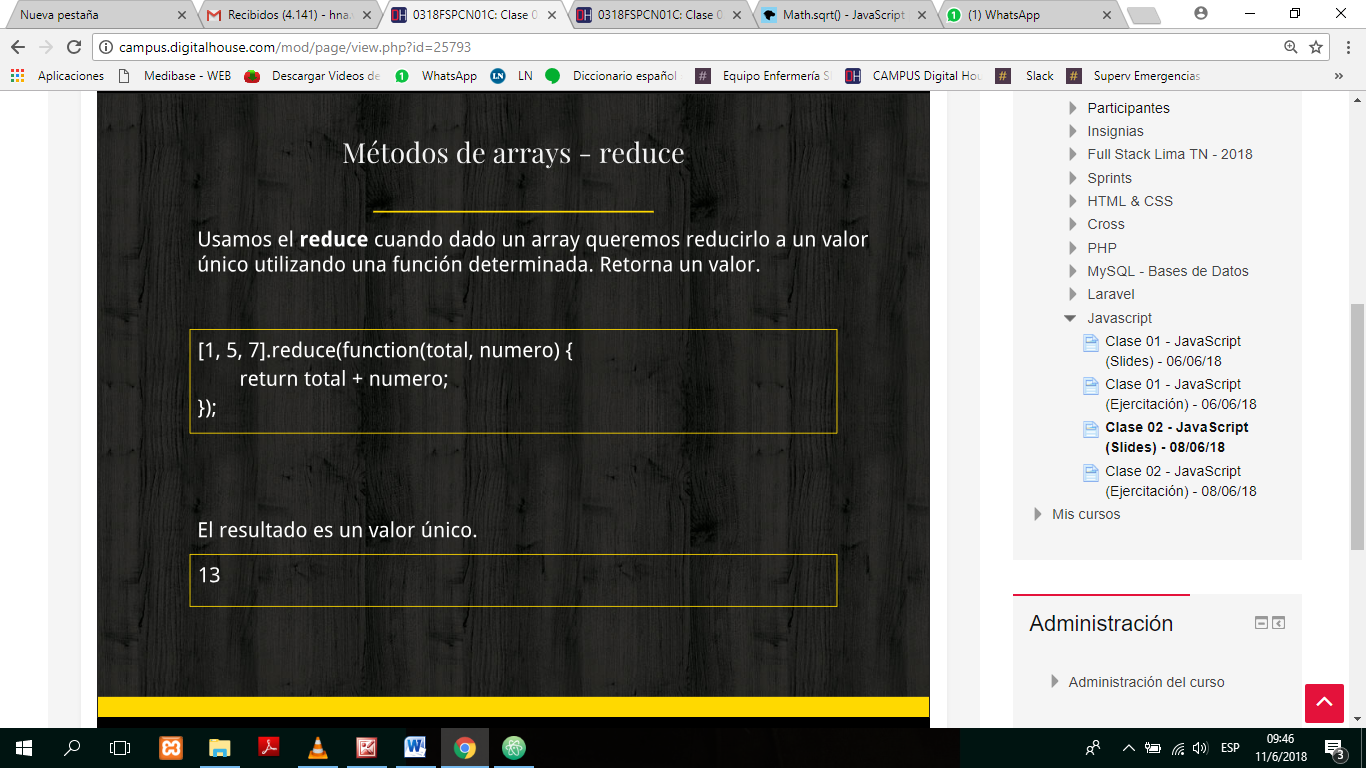
**Filter:** devuelve un array nuevo, no modifica el primer Array pero necesita devolver un valor booleno para devolverme un objeto que cumpla con esa función.

* Recibe un callback, quien sólo recibe un elemento
* Por más que me devuelva UN solo elemento, el resultado siempre SERÁ un **ARRAY.**
* O sea, **devuelve un ARRAY!!!**



**Reduce**: devuelve un valor, puede ser un n°, un string o un array.

* Itera cada elemento del array y te los devuelve todo en CONJUNTO.
* Al Callback, a esa función, Necesito pasarle un acumulador y el elemento del array.



OBJETO LITERAL: es un tipo de datos.

La propiedad no puede tener guiones, ni espacios

Una propiedad puede ser una function, un array, un string, un número

COMO LO ESCRIBO: Llamar al elemento (punto) y a continuación la propiedad.

THIS: se refiere a a la propiedad que mencione de un objeto.

En la consola si la uso, para acceder a un archivo java, escribo node y el nombre del archivo

PARA ROMPERSE LA CABEZA: codewars.com

Para descargar en la consola

https://nodejs.org/en/

1. Hardcodear: perder flexibilidad y/o prolijidad dejando valores y/o comportamientos fijos en el código del programa.  Es predeterminar ciertos datos dentro del código fuente. Eso hace un código menos flexible, pero más rápido de desarrollar.

   EJEMPLO 1:   
   Uno puede hacer un programa de contabilidad en el que el IVA sea siempre el 21%, escribiendo ese 21% dentro del código, es algo rápido y directo, pero ¿qué sucedería si el gobierno cambia la tasa del 21% al 22% por ejemplo?, pues que tu programa ya no vale, el 21% está hardcodeado dentro del código, no es algo configurable por el cliente.   
     
   EJEMPLO 2:   
   Imaginemos que estamos haciendo un programa que simplemente abre un documento cuando pulsamos un botón. Podemos hacer que el usuario pueda configurar qué documento abrir mediante una ventana de configuración, o "hardcodear" dentro del código del programa la ruta de un archivo en concreto, por ejemplo "C:/midocumento.doc". Hardcodear es algo muy socorrido a la hora de programar, programas rápido, pero hace un programa menos flexible, en nuestro caso el botón abrirá siempre ese documento, y no otro, porque la ruta del documento está "hardcodeada" [↑](#endnote-ref-1)