Vie 150618

**InnerHTML**

Se usa para agregar texto.

H6.innerHTML

**CREATE TEXT NODE**

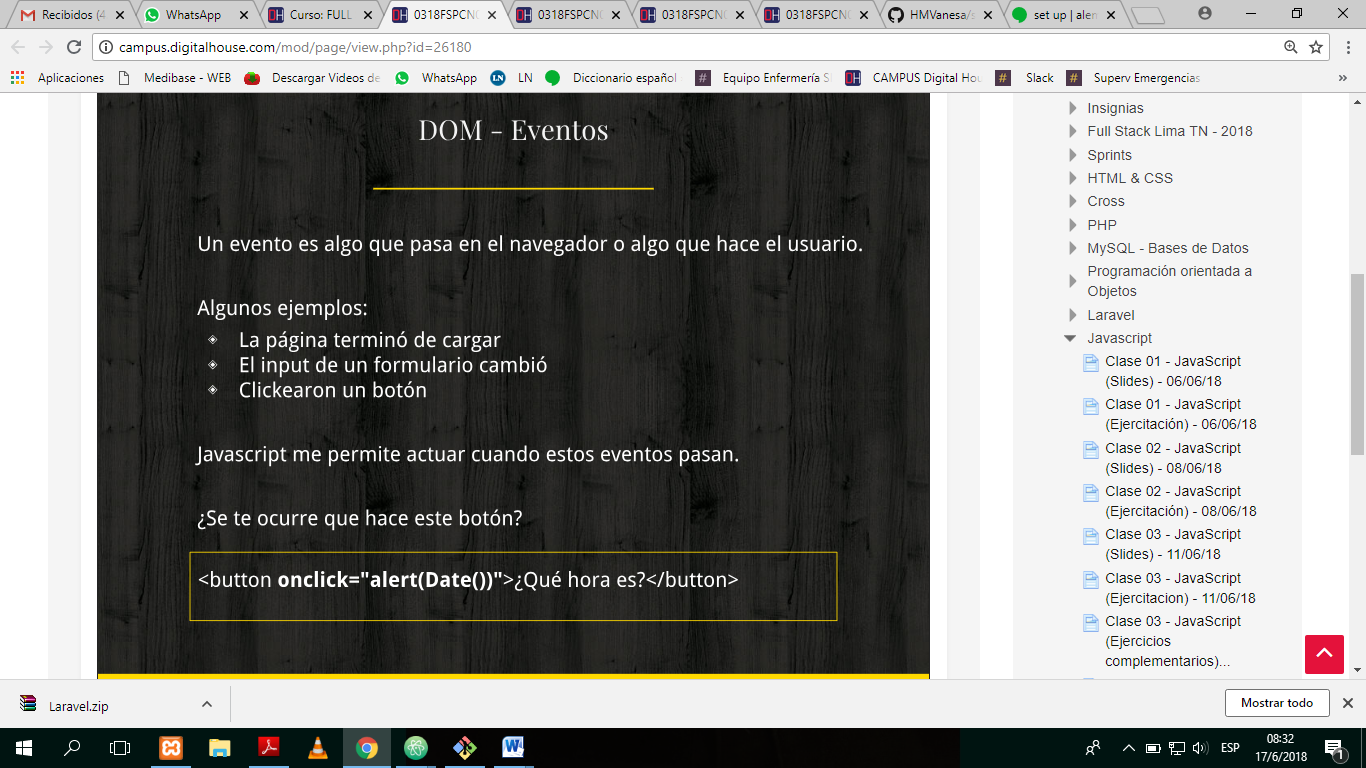
El NODE es texto plano.

Hay **dos formas** de crear elementos: esto es lo mejor:

createElement luego innerText= “crea texto plano” un nodo de JAVA

luego innerHTML= y ya crea la etiqueta y pongo dentro el texto.

# EVENTOS



Por ejemplo Windows.**onload**, es un evento, que significa a la ventana cuando se termine de cargar la página ejecutá tal cosa.(la function que se escribe después de onload)

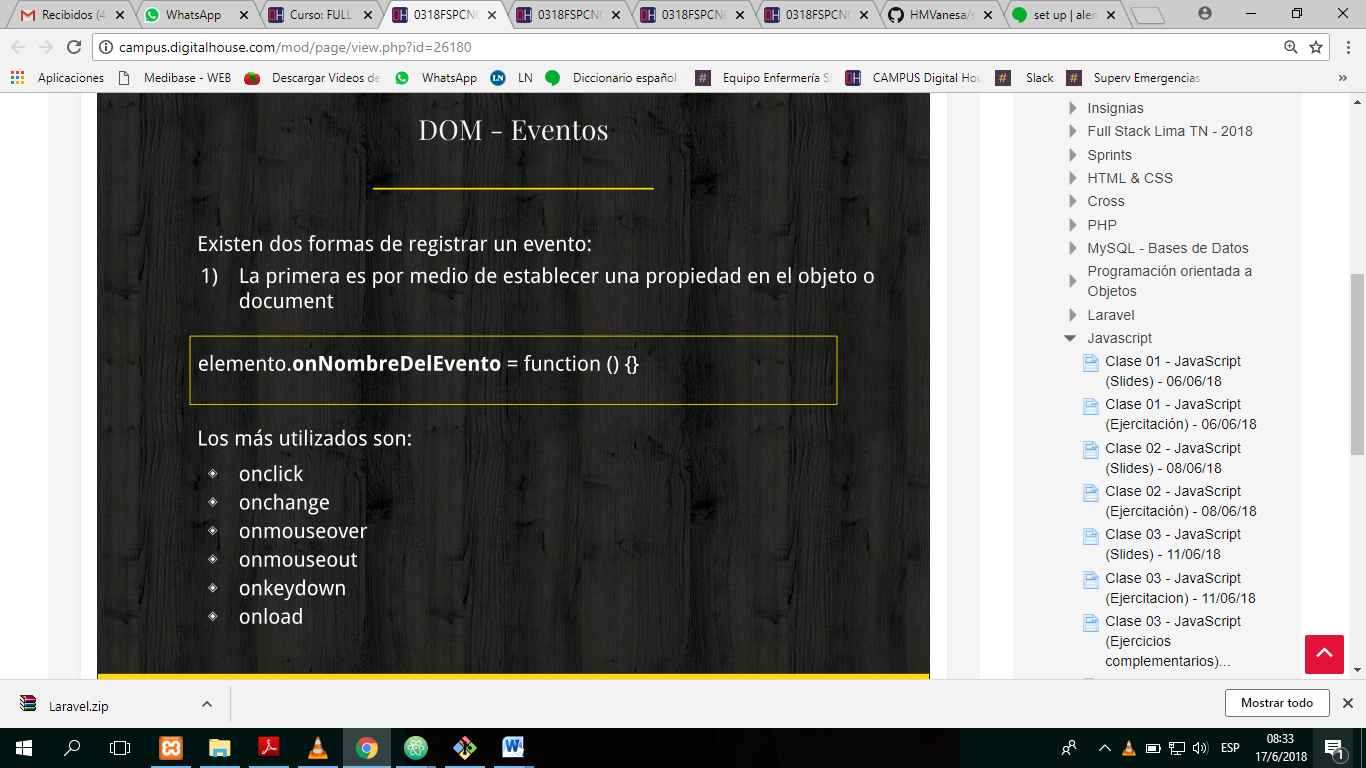
Usar EVENTOS es lo más **recomendable** para **VALIDAR** un formulario desde el FRONTEND**.** Sólo desde el BACKEND es justo y necesario validar información sensible como son el usuario y la contraseña.

Una de las formas más comunes de usar los EVENTOS es pasarlos como ATRIBUTOS de la etiqueta. Todas las etiquetas tienen estos atributos. Hay algunos atributos que funcionan en algunas etiquetas y otros atributos que funcionan en otra etiqueta. Por ej: “onSubmit” funciona sólo en formularios y no en la etiqueta <a>.

Ojo! Si trabajamos con **BOTONES** **y no ponemos el TIPO (type)[[1]](#footnote-1)** , el tipo por defecto es un SUBMIT y si está dentro de un formulario va a viajar por GET o X POST, es decir va a refrescar la ventana. ¡por ello especificar siempre el TIPO!!! Si está dentro de un formulario y tiene Type=button, NO HACE NADA.

<button name="button" **type**="button"> Fecha y hora </button>

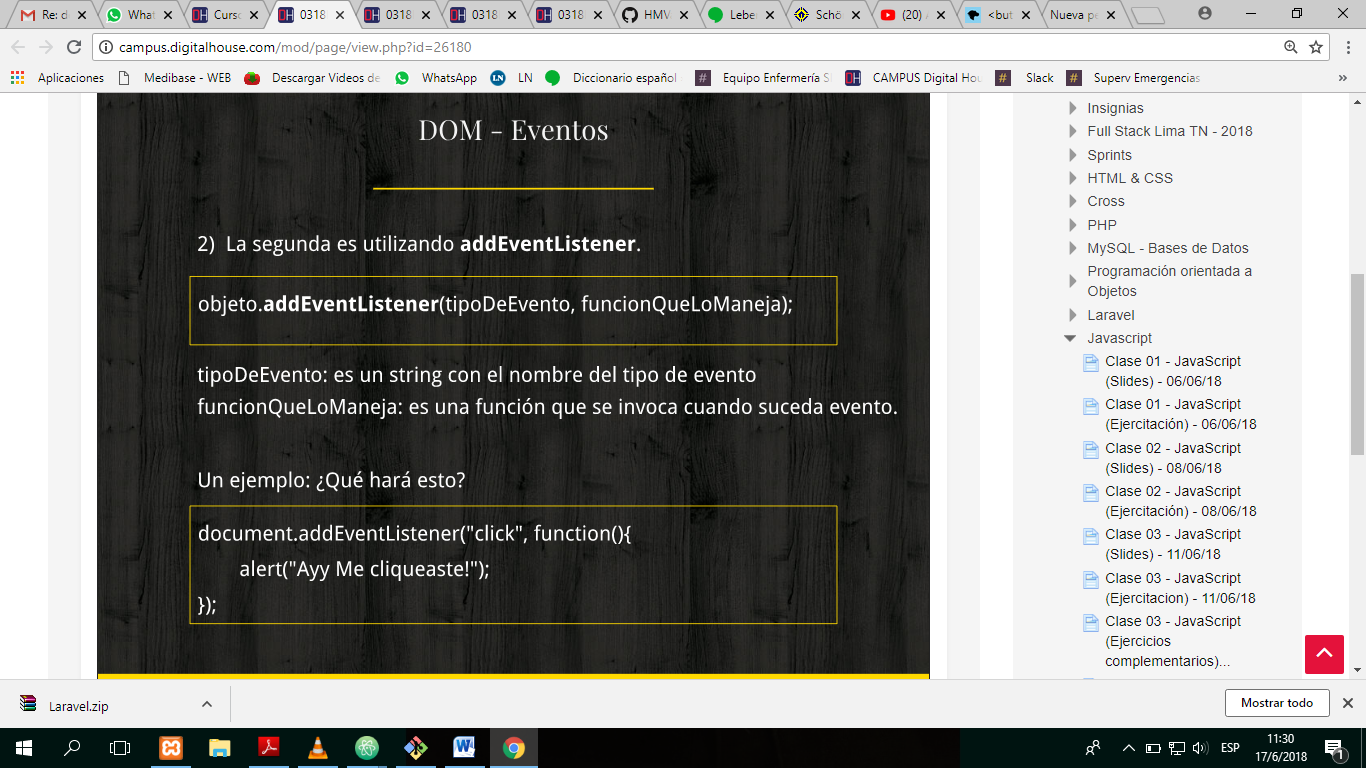
<button onclick= “alert (Date ())”> Fecha y hora </button>



Una función anónima

**Esto es más recomendado: addEventListener**, (agregáme un escuchador de eventos!!) Es un método que es una función.

Recibe DOS parámetros



Pero **sin** las letras “on”

es un callback (que se ejecuta dentro de otra func) que se ejecuta cuando se dispara ese evento.

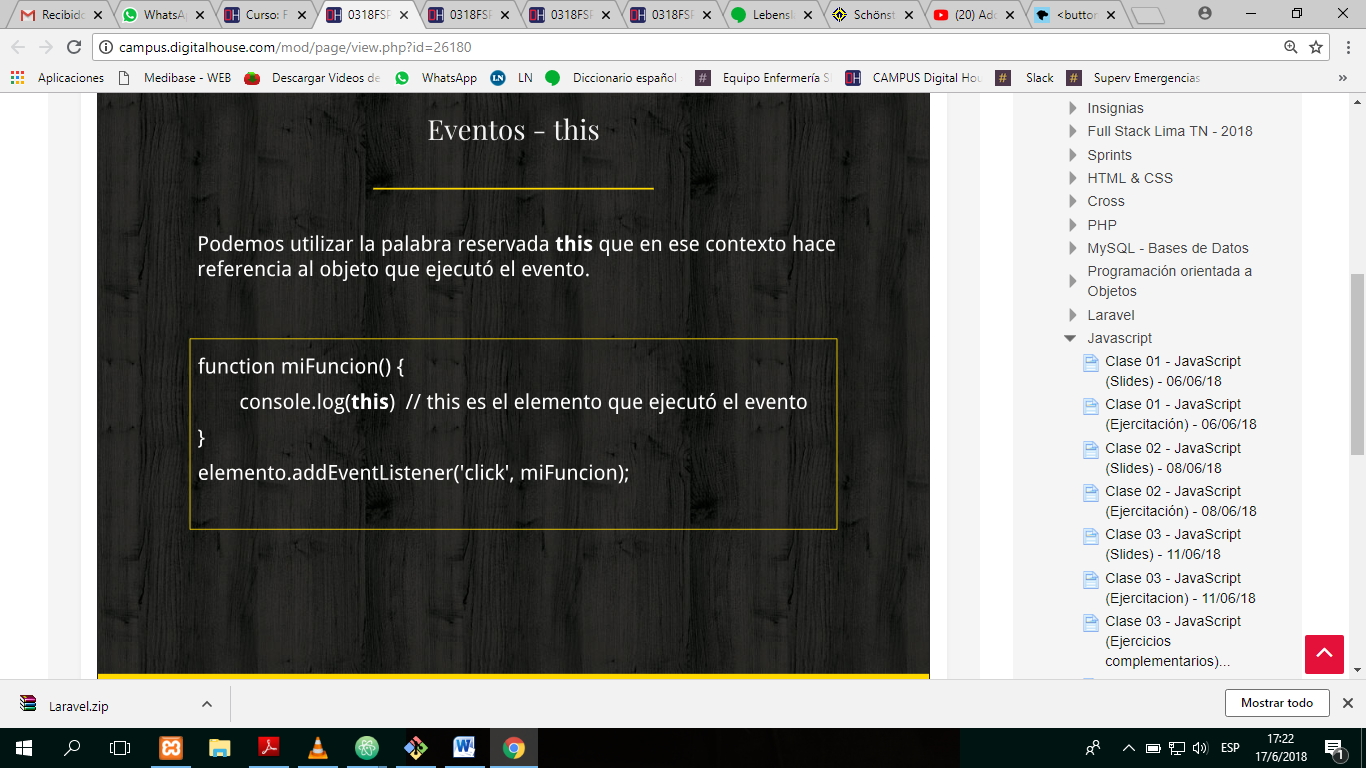
Se ejecuta primero el evento de JAVA S y luego se ejecuta el evento de la etiqueta.

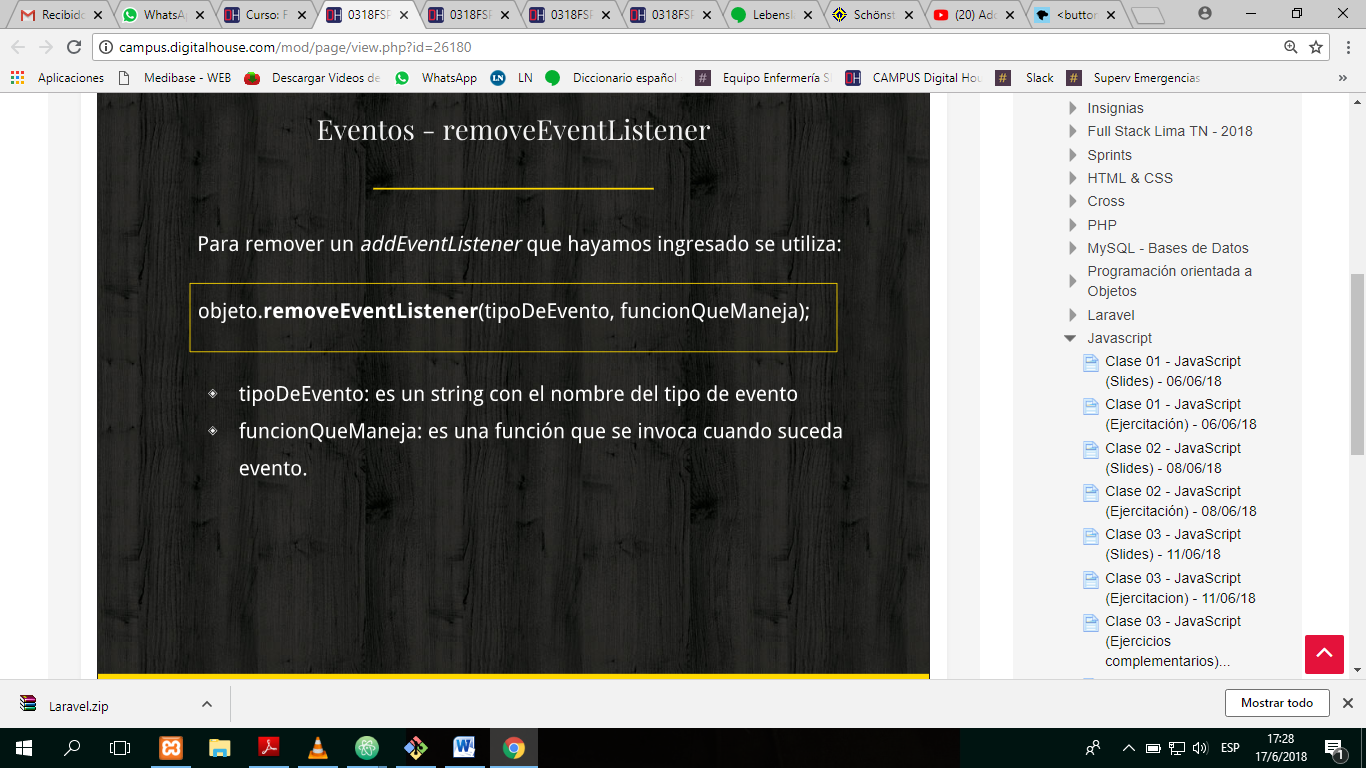
Si quiero q’ varios objetos hagan siempre lo mismo, o sea ejecuten una misma función, se puede hacer una función NO ANÓNIMA, es decir una function con Nombre q voy a definir ANTES de la EVENTLISTENER y luego se la paso como CALLBACK.

Por ej. Cuando quiero verificar que los campos de un formulario no estén vacíos, hago la function, hago un forEach de los elementos del formulario y voy validando y en pequeñas líneas tengo la validación.

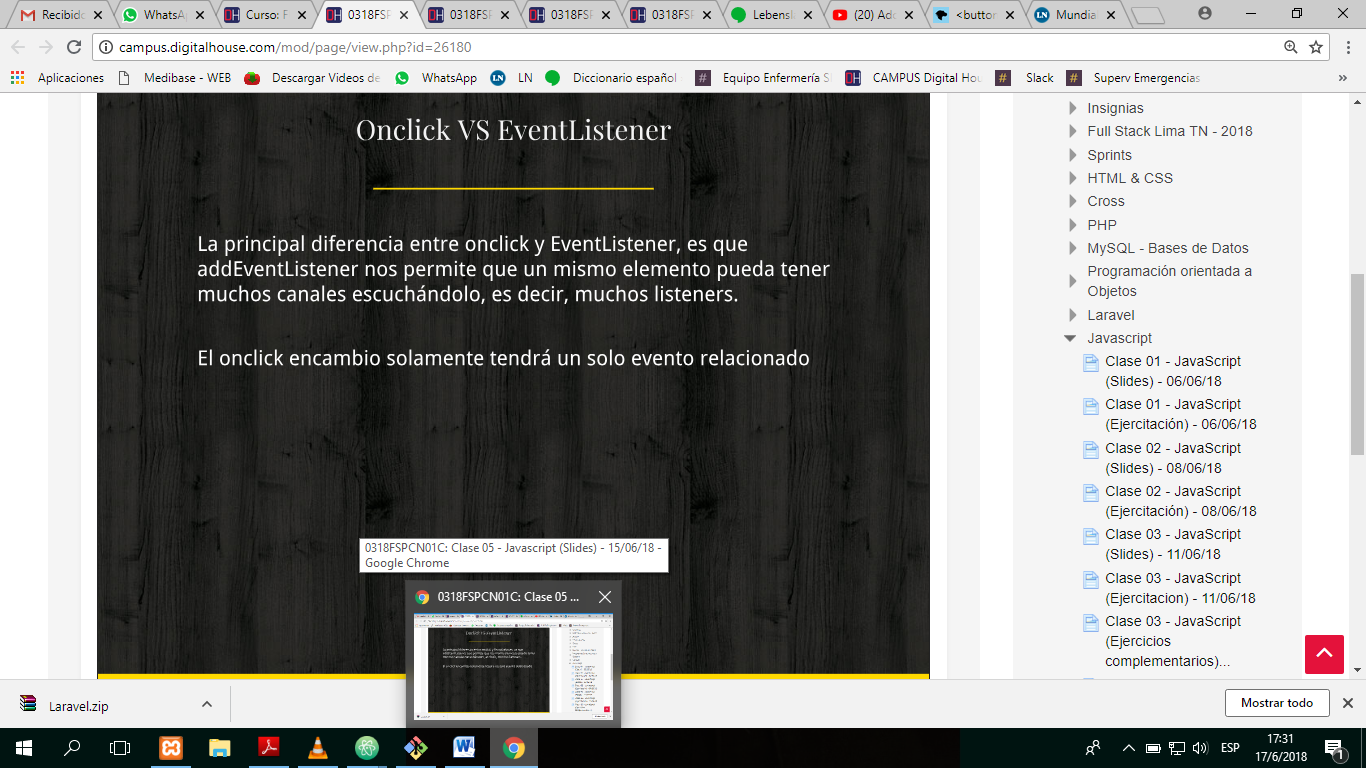
**THIS (de éste)**

Tiene un SCOPE local si se encuentra adentro de una function de un objeto. THIS hace referencia al elemento que estoy ejecutando en ese momento. Repasar lo de objeto literal (en el ej.teníamos una propiedad que tenía una function que saludaba, Hola THIS nombre, THIS apellido)



**REMOVE EVENT LIstener**

¿**Cuál es la diferencia entre el ONclick y el EventLISTENER?**



Y si escribí varios ONCLICK sirve sólo para el último

Los selectores son=

# para los id

En el siguiente ejemplo, tengo distintos canales escuchando y por ello se ejecutan los dos:

document.querySelector('h1').addEventListener ("mouseover", function (){

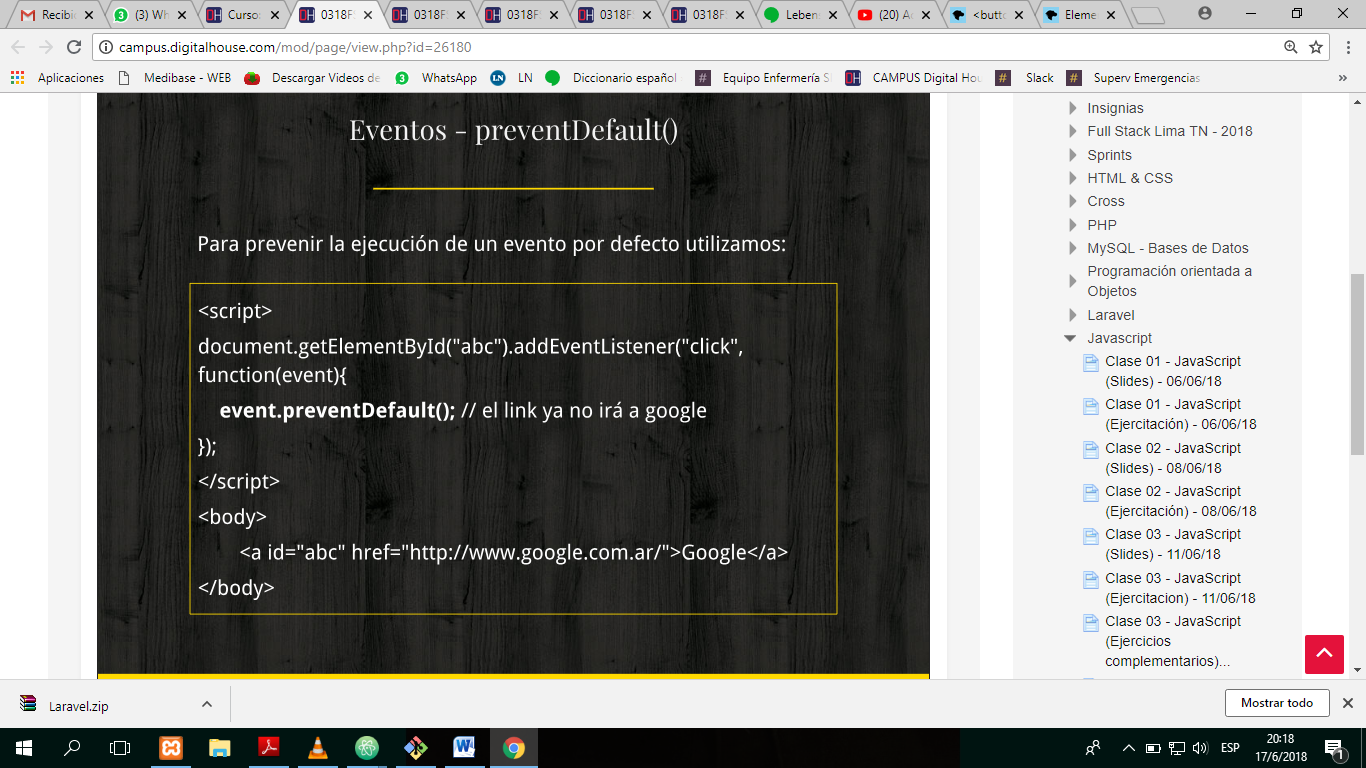
alert ('Entraste al h1');

})

document.querySelector('h1').addEventListener ("mouseout", function (){

alert ('Saliste del h1');

})

**PREVENT DEFAULT**

var elA.onclick= document.querySelector (‘a’);

elA.onclick= function (event){

event.preventDefault();

};

Otra forma prolija:

var elA.onclick= document.querySelector (‘a’);

elA.onclick= function (e){

e.preventDefault();

};

**ASINCRONISMO o AJAX**

Pedidos asincrónicos.

TIMERS

1. ***Type:*** *Los valores posibles son:*

   * *submit: El botón envía los datos del formulario al servidor. Este es el valor predeterminado si no se especifica el atributo, o si el atributo es cambiado dinámicamente a un valor inválido o vacío.*
   * *reset: El botón reinicia todos los controles a sus valores iniciales.*
   * *button: El botón no tiene comportamiento predeterminado. Puede haber scripts de la parte del cliente asociados a los eventos del elemento, que serán lanzados cuando el evento ocurra.*

   [↑](#footnote-ref-1)