Contenido

[1-Variables 3](#_Toc95832525)

[2-Tipos de datos 3](#_Toc95832526)

[3-Bucles 3](#_Toc95832527)

[4-Ventanas 3](#_Toc95832528)

[5-Funcion isNaN 3](#_Toc95832529)

[6-Funciones 3](#_Toc95832530)

[7-Funciones con parámetros 3](#_Toc95832531)

[8-Funciones con parámetros opcionales 4](#_Toc95832532)

[9-Funciones con parámetros Rest y Spread 4](#_Toc95832533)

[10-Spread: 4](#_Toc95832534)

[11-Funciones Anónimas 4](#_Toc95832535)

[12-Función Fecha 4](#_Toc95832536)

[13-Funciones de texto 5](#_Toc95832537)

[13-1-Convertir un número a texto (PGM relacionados 13-funciones-texto.js y 13v-funcionesTexto.js y plantilla.js) 5](#_Toc95832538)

[13-2-Convertir en minúscula y mayúscula 5](#_Toc95832539)

[13-3-Calcular la longitud de un texto: 5](#_Toc95832540)

[13-4-Función para buscar una palabra en una cadena 5](#_Toc95832541)

[13-5-Función search 5](#_Toc95832542)

[13-6-Función match 5](#_Toc95832543)

[13-7-Extraer una parte de un string función substr 5](#_Toc95832544)

[13-8-Para extraer una letra concreta de un String función charAt 5](#_Toc95832545)

[13-9-Busqueda de una palabra al comienzo del string 6](#_Toc95832546)

[13-10-Métodos para reemplazar texto por otro 6](#_Toc95832547)

[13-11-Método para sacar una porción de string a partir de una posición. (slice) 6](#_Toc95832548)

[13-12-Método Split 6](#_Toc95832549)

[13-13-Eliminar espacios en un string utilizamos el método trim 6](#_Toc95832550)

[14-Definición de un arreglo (PGM arreglos.js mezclado con etiquetas HTML) 6](#_Toc95832551)

[15-Foreach 6](#_Toc95832552)

[16-Array Multidimensionales (pgm arreglos-multi.js y arregloFunciones.js – funciones find y findIndex- pgm ejerciciosArreglo.js usa funciones de ordenado como son SORT y REVERSE) 6](#_Toc95832553)

[17-Para acceder a un elemento del array multidimensional 7](#_Toc95832554)

[17-1-Funciones para array (push,pop,indexOf,splice,join y split): 7](#_Toc95832555)

[17-1-1-Recorrer un array usando for in 7](#_Toc95832556)

[17-2-Funciones de búsqueda en un array (find, findIndex y some) 7](#_Toc95832557)

[18-DOM 7](#_Toc95832558)

[Para seleccionar un id concreto 7](#_Toc95832559)

[Para capturar el código que tiene la etiqueta html 7](#_Toc95832560)

[Para agregar un valor nuevo a la etiqueta HTML 7](#_Toc95832561)

[Agregando un estilo 7](#_Toc95832562)

[Para agregar un clase al elemento html uso el atributo className 8](#_Toc95832563)

[Otra forma de seleccionar un elemento HTML tipo CSS 8](#_Toc95832564)

[//Seleccionar elementos HTML por etiqueta 8](#_Toc95832565)

[//Para seleccionar el contenido de un elemento html 8](#_Toc95832566)

[Realizar operaciones en el DOM -> createElement, createTextNode y appendChild 8](#_Toc95832567)

[Crear un nodo 8](#_Toc95832568)

[Creando un nodo de tipo Text 8](#_Toc95832569)

[Seleccionar un elemento por su elemento TAG 8](#_Toc95832570)

[Seleccionar un elemento por su clase HTML 8](#_Toc95832571)

[Eventos 8](#_Toc95832572)

[Función escuchadora de eventos--> addEventListener 8](#_Toc95832573)

[Evento click 9](#_Toc95832574)

[Evento mouseOver 9](#_Toc95832575)

[Evento mouseOut 9](#_Toc95832576)

[Eventos del Teclado 9](#_Toc95832577)

[Focus 9](#_Toc95832578)

[Blur: 9](#_Toc95832579)

[Keydown: 9](#_Toc95832580)

[Keypress: 9](#_Toc95832581)

[Keyup: 9](#_Toc95832582)

[Evento Load 9](#_Toc95832583)

[Como seleccionar el contenido de un elemento HTML usando document.querySelector y su propiedad value 9](#_Toc95832584)

[Timers 9](#_Toc95832585)

[INDEX.HTML -> códigos (tipos de datos, operadores, ventanas, ejercicios de programación, funciones, parámetros, funciones anónimas, funciones de texto, plantillas, arreglos, arreglos multidimensionales, operaciones con array, funciones para arreglos, ejercicios con arreglos ) 9](#_Toc95832586)

[INDEX2.HTML 10](#_Toc95832587)

[INDEX3.HTML -> EVENTOS, TEMPORIZADORES 10](#_Toc95832588)

[Definición de un objeto (Json): son array asociativos en javascript 10](#_Toc95832589)

[-Definición de un arreglo con objetos dentro 10](#_Toc95832590)

[-Imprimir un atributo del objeto 11](#_Toc95832591)

[-Cargar un elemento a un objeto 11](#_Toc95832592)

[Recorriendo un arreglo con objetos json y creando un párrafo e insertándolo en un div creado en el HTML 11](#_Toc95832593)

[LocalStorage: es como una memoria que permanece mientras navego en la página web. 11](#_Toc95832594)

# 1-Variables

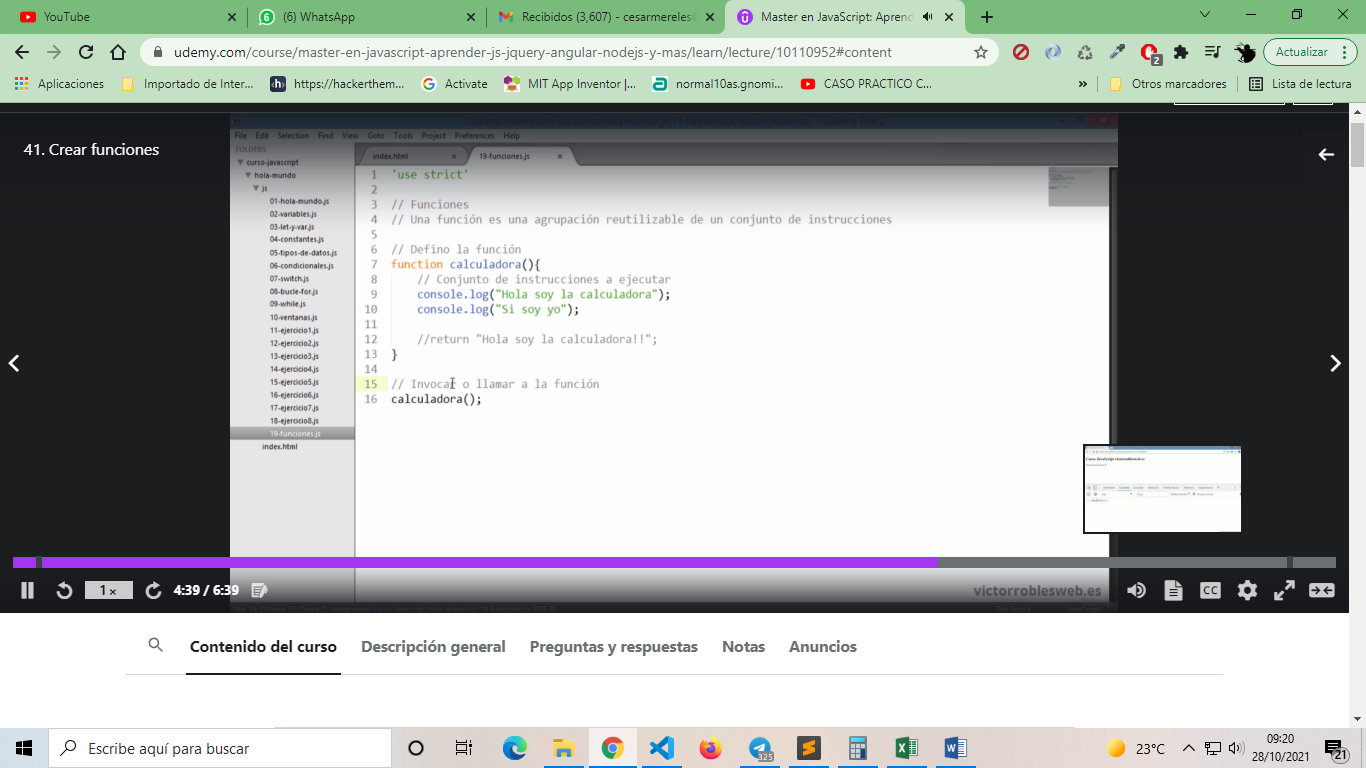
2-Tipos de datos-> funciones Number, parseInt y String

# 3-Bucles

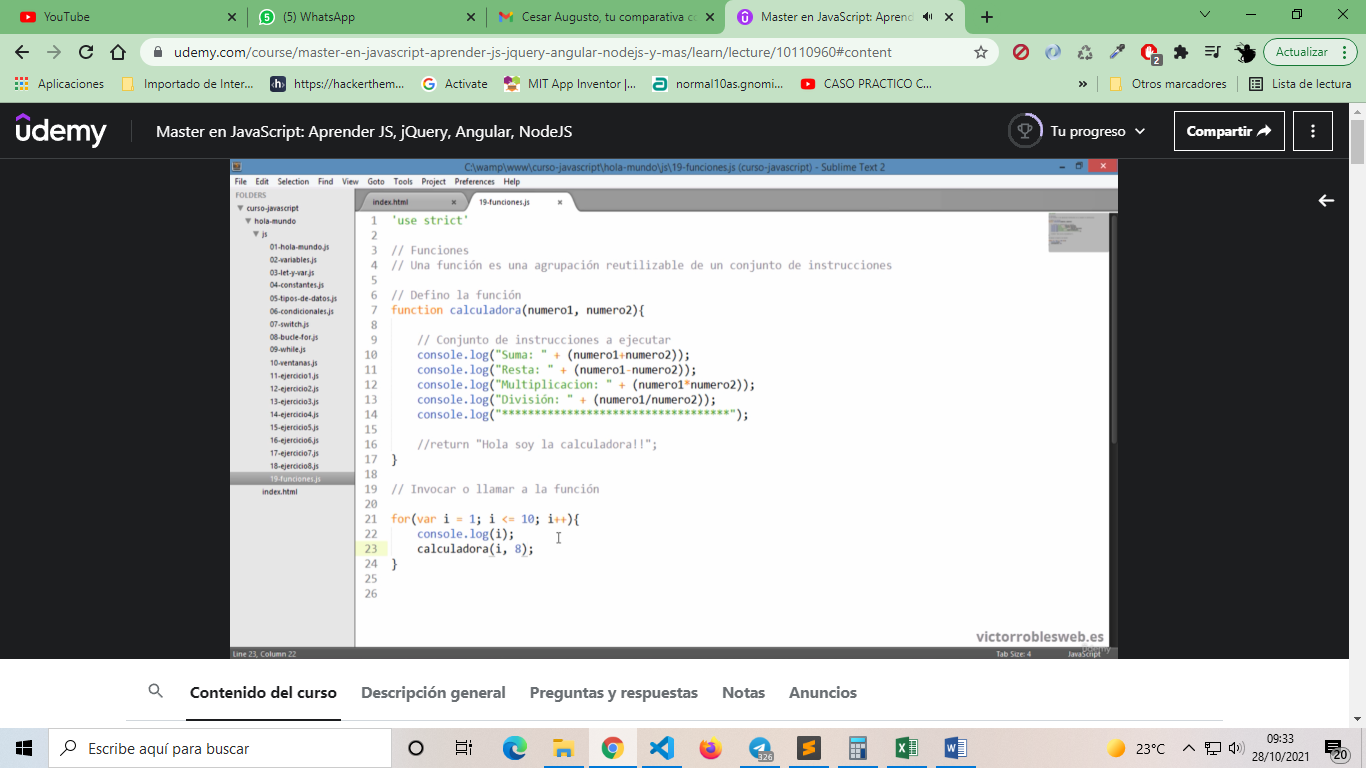
4-Ventanas -> código ventanas.js tenemos Alert, Confirm y Prompt

5-Funcion isNaN devuelve un dato booleano (True/False)

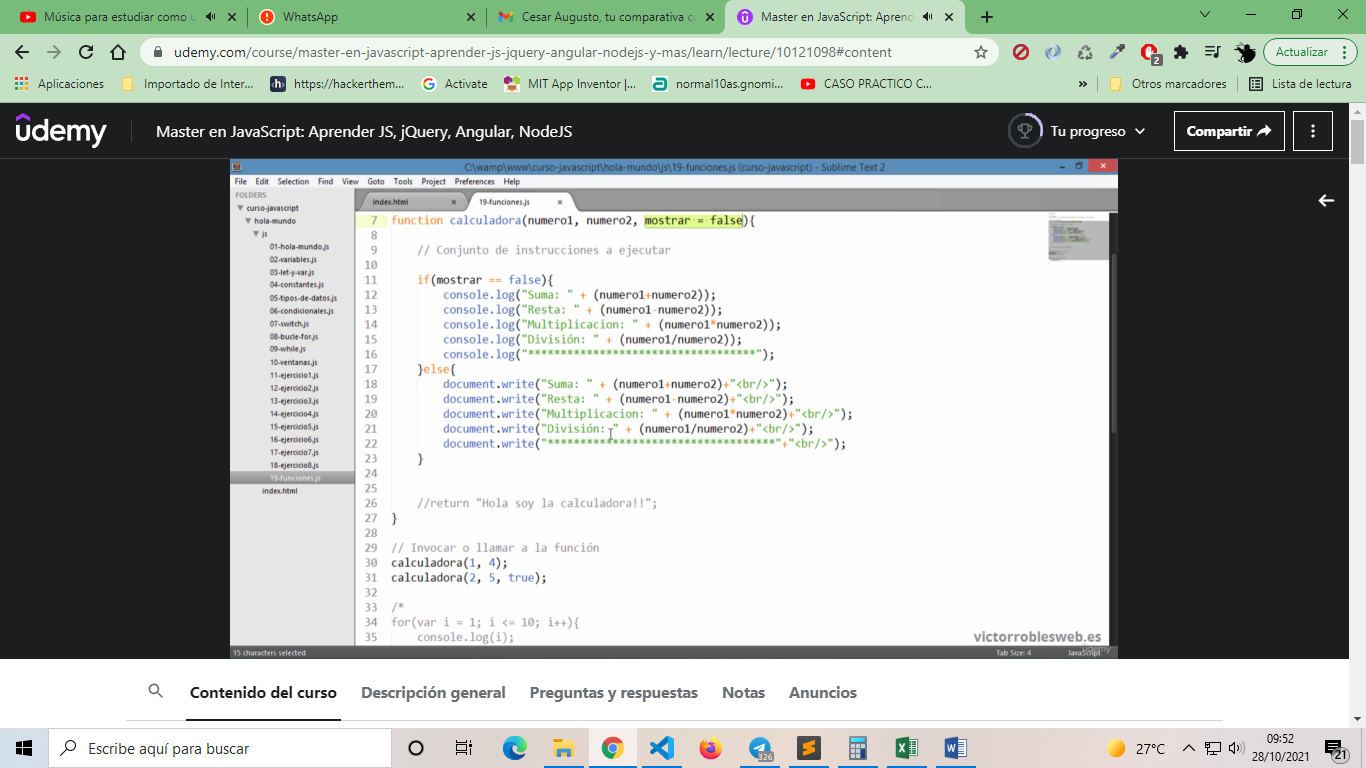
# 6-Funciones



# 7-Funciones con parámetros

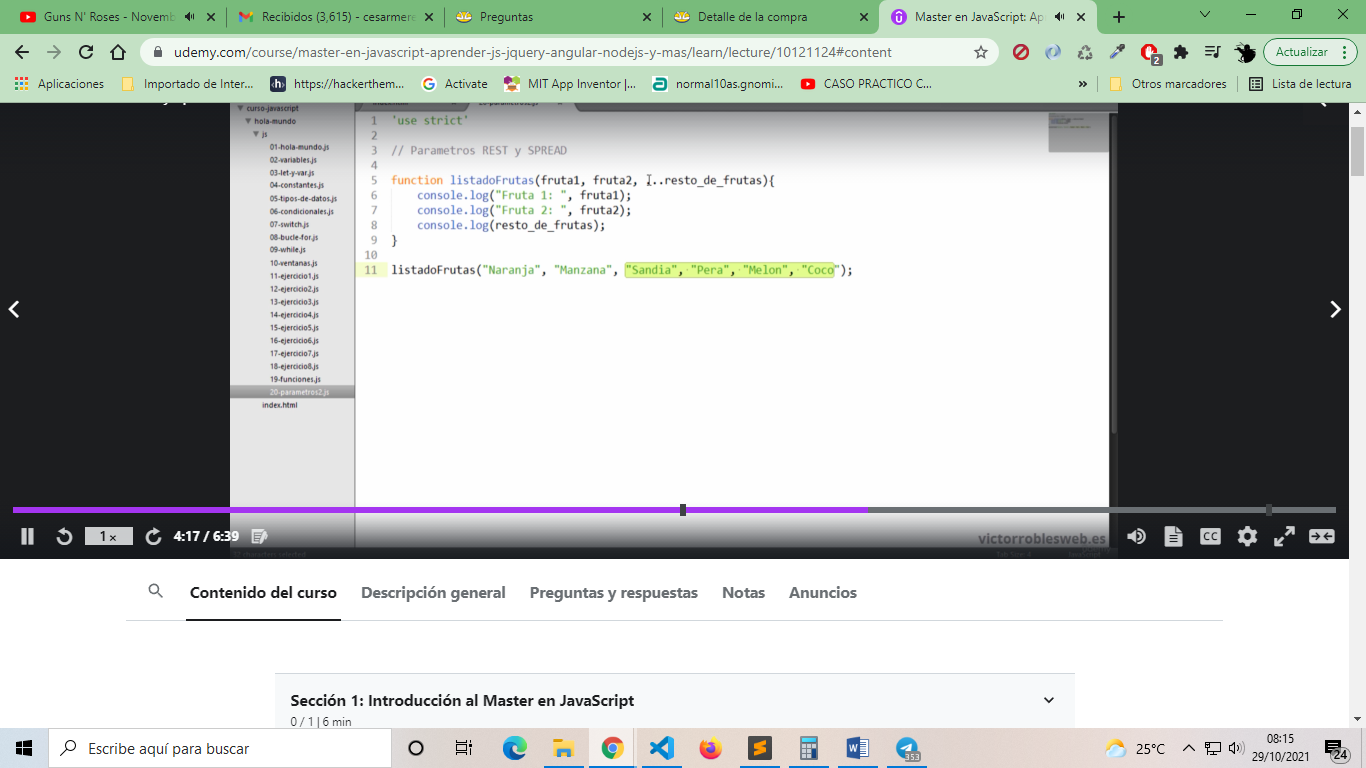


# 8-Funciones con parámetros opcionales

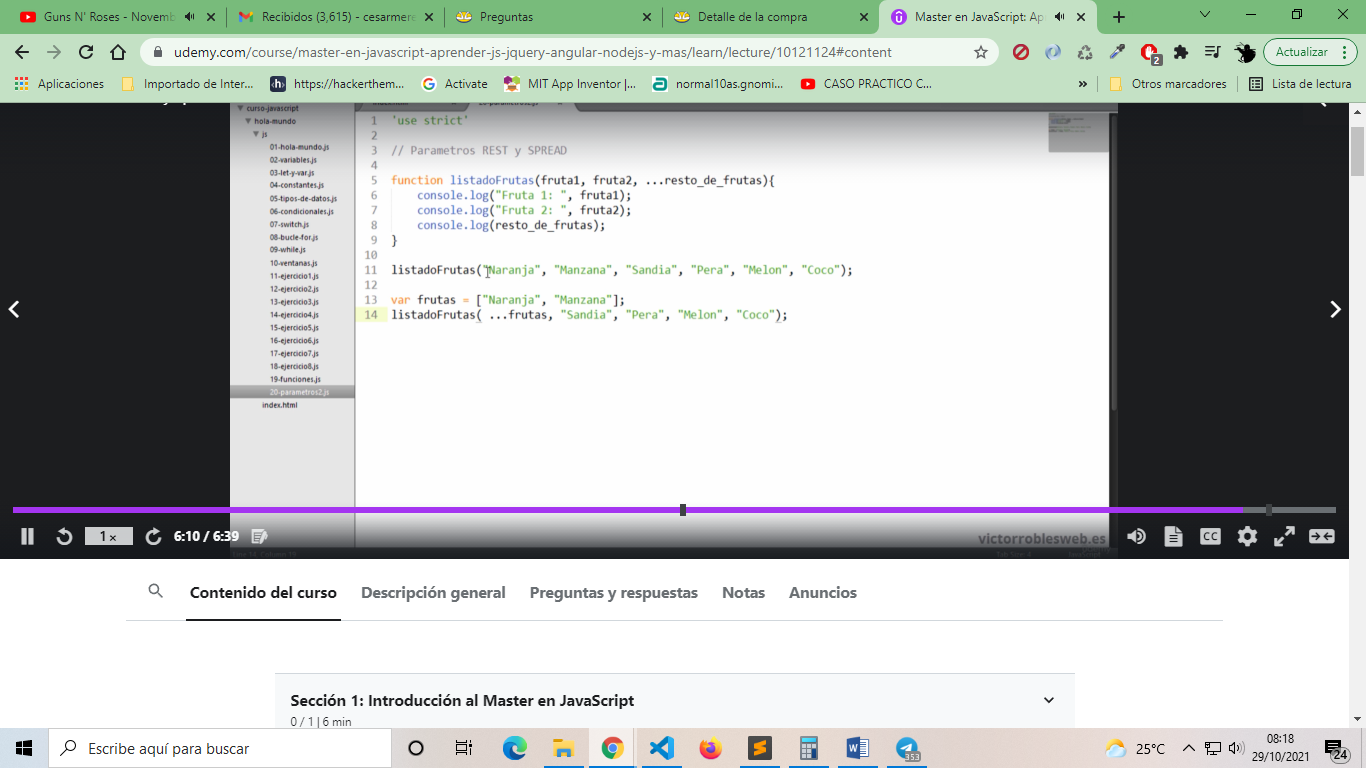


# 9-Funciones con parámetros Rest y Spread

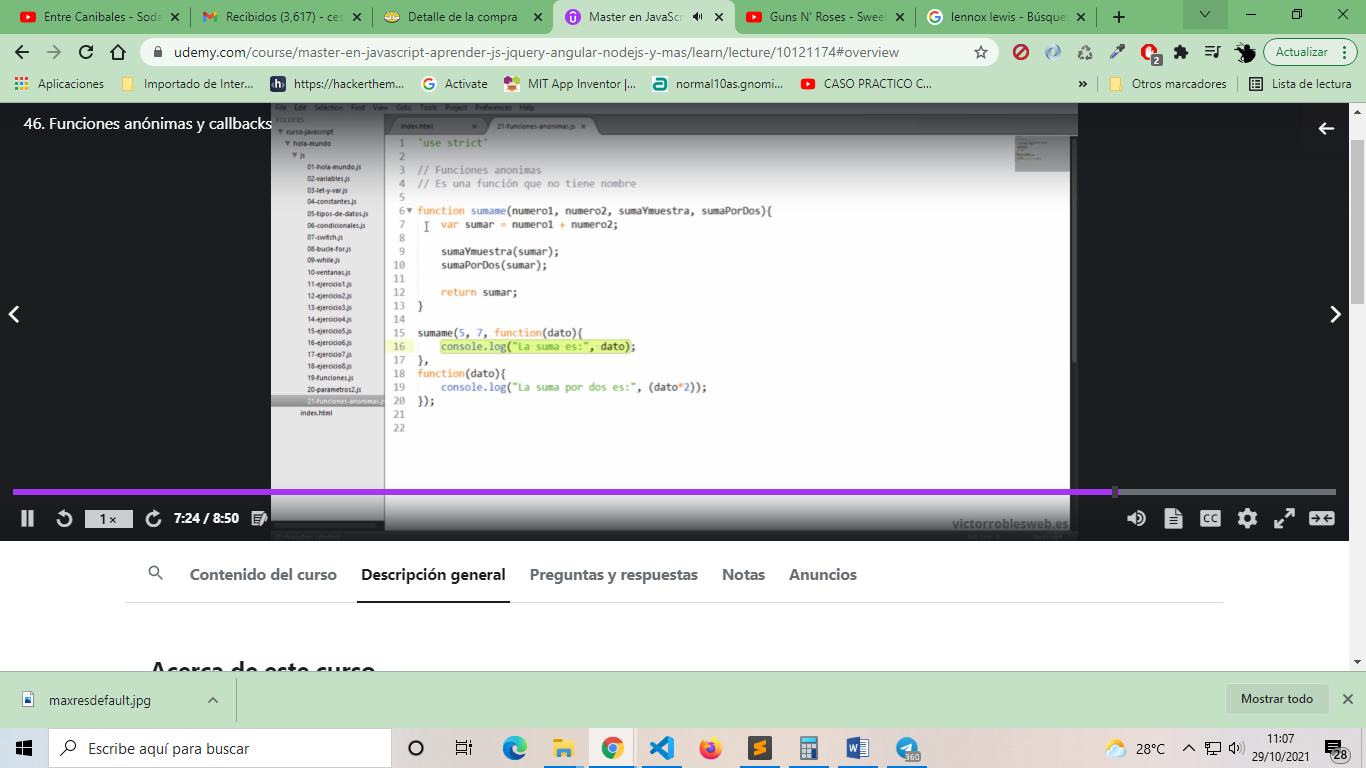
Rest: nos permite capturar un número indefinido de parámetros



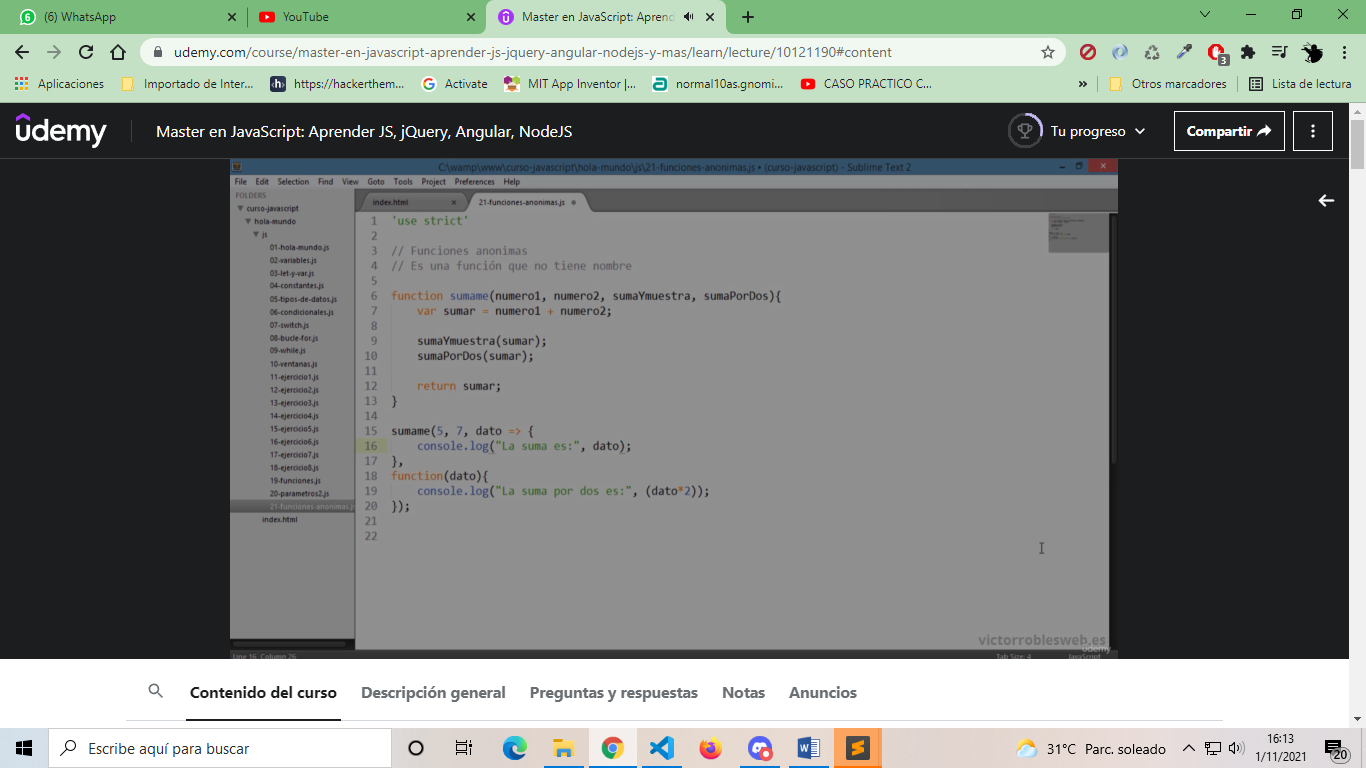
# 10-Spread:



11-Funciones Anónimas: es una función que no tiene nombre. Se usa mucho para hacer callback (es una función que se ejecuta dentro de otra)



# 12-Función Fecha



# 13-Funciones de texto

# 13-1-Convertir un número a texto (PGM relacionados 13-funciones-texto.js y 13v-funcionesTexto.js y plantilla.js)

let dato=numero.toString();

# 13-2-Convertir en minúscula y mayúscula

let dato1=texto1.toUpperCase(); <-mayúscula

let dato2=texto2.toLowerCase(); <-minúscula

13-3-Calcular la longitud de un texto:si el texto no tiene nada o si se asigna un numero devuelve o 0 o undefined. Si tiene datos devuelve el número de caracteres.

let nombre = "Mereles cesar augusto";

console.log(nombre.length);

# 13-4-Función para buscar una palabra en una cadena (indexOf)

indexOf = devuelve en qué posición esta la palabra buscada. Si no la encuentra devuelve -1

let texto = "Nahiara Nicel Mereles";

console.log("(indexOf) está en la posición: "+texto.indexOf("Nicel"));

let texto1 = "Nachato cesar ignacio cesar Mereles";

console.log("(lastIndexOf) esta en la posicion: "+texto1.lastIndexOf("cesar"));

13-5-Función search hace lo mismo que usar indexOf

13-6-Función match

Devuelve un array con las palabras si no encuentra devuelve null

let cadena = "Cesar Augusto Mereles es hincha de River Plate Argentina River";

let busqueda = cadena.match(/River/g);

# 13-7-Extraer una parte de un string función substr

let sacarString = cadena.substr(6 =>(posición desde donde se comienza a sacar),7 =>(Numero de caracteres a querer sacar);

# 13-8-Para extraer una letra concreta de un String función charAt

let letra = cadena.charAt(0)

# 13-9-Busqueda de una palabra al comienzo del string

let busqueda\_startWith = cadena.startsWith("Cesar"); Si lo encuentra devuelve True o caso contrario False. Para buscar en el final de la cadena tenemos el método endsWith. Otro método utilizado para la búsqueda también es el includes. También devuelve True si encuentra y False caso contrario

let ejemplo = cadena.includes(“Palabra buscada”)

# 13-10-Métodos para reemplazar texto por otro

let textoNuevo = cadena.replace("hincha","Super hincha");

# 13-11-Método para sacar una porción de string a partir de una posición. (slice)

let slice\_metodo = cadena.slice(10);

13-12-Método Split -> convierte en un array un string

let split\_metodo = cadena.split(" "); Dejando un espacio toma todo el string y no le separa. Este método es muy utilizado en buscadores.

# 13-13-Eliminar espacios en un string utilizamos el método trim

# 14-Definición de un arreglo (PGM arreglos.js mezclado con etiquetas HTML)

const arreglos = ["cesar","river","campeon",43,true];

const lenguajes = new Array("PHP","JS","JAVA","PYTHON");

# 15-Foreach

//elemento->es el arreglo | índice -> es el índice del arreglo | arr -> es el array original

lenguajes.forEach((elemento,índice,arr) => { }

# 16-Array Multidimensionales (pgm arreglos-multi.js y arregloFunciones.js – funciones find y findIndex- pgm ejerciciosArreglo.js usa funciones de ordenado como son SORT y REVERSE)

//defino un array multidimensional

const categorias = ['accion','terror','comedia'];

const peliculas = ["avenger", "mentiros mentiroso", "la liga de la justicia"];

const cine =[categorias,peliculas];

# 17-Para acceder a un elemento del array multidimensional

console.log(cine[0][2]); -> esto devuelve en la posición 0 (Fila) “Categorias” 2 (Columna) en este caso el dato que devuelve es comedia

console.log(cine[1][0]);

17-1-Funciones para array (push,pop,indexOf,splice,join y split): arreglo.push(valor) para agregar un elemento al array. Para eliminar un elemento del array uso la función pop(). Ejemplo PGM 17-1-operaciones-array.js

Para buscar un elemento en el array funciones como indexOf. Arreglo.indexOf(‘ELEMENTO BUSCADO’). Si no encuentra dicho elemento devuelve -1. Para eliminar uso el método splice. Por ejemplo tengo que buscar el elemento que quiero eliminar y le paso a una variable y luego invoco al arreglo.splice(valor\_buscado,1). El 1 indica la cantidad de elementos a eliminar.

La función join se utiliza para convertir un array en string. Otra función para convertir un texto en arreglo es la función Split. Su sintaxis es

let cadena = “text1, text2, text3”

const cadena\_array = cadena.split(“, “) -> código ejemplo 17-1-operaciones-array.js

17-1-1-Recorrer un array usando for in -> código ejemplo 17-1-operaciones-array.js

# 17-2-Funciones de búsqueda en un array (find, findIndex y some)

PGM 17-2-arregloFunciones.js

Find -> si no encuentra devuelve undefined. Esto busca el contenido

findIndex -> devuelve el índice del elemento buscado

some-> es para utilizar con array con valores numéricos en caso que cumpla la condición devuelve true

18-DOM (document object model)->podemos modificar el html de una página.

## Para seleccionar un id concreto

let caja = document.getElementById("miCaja");

## Para capturar el código que tiene la etiqueta html

let caja = document.getElementById("miCaja").innerHTML; <-innerHTML se utiliza para capturar el contenido del div.

## Para agregar un valor nuevo a la etiqueta HTML

let cajaModificado = document.getElementById("miCajaModificado");

cajaModificado.innerHTML= "Incrusto un texto desde JS";

console.log(cajaModificado);

## Agregando un estilo

let pEstilo = document.getElementById("estilo");

pEstilo.style.background="red";

Para agregar un clase al elemento html uso el atributo className

pEstilo.className = "ClaseNueva";

## Otra forma de seleccionar un elemento HTML tipo CSS

let tipoquery = document.querySelector("#tipoquery");

tipoquery.style.background="green";

## //Seleccionar elementos HTML por etiqueta

let todosLosDiv = document.getElementsByTagName("div");

## //Para seleccionar el contenido de un elemento html

let contenido\_div = todosLosDiv[3].textContent;

La función textContent y innerHTML hacen lo mismo con la diferencia que innerHTML te permite modificar el contenido de la etiqueta HTML

# Realizar operaciones en el DOM -> createElement, createTextNode y appendChild

## Crear un nodo

Creamos un nodo invocando al método createElement del objeto de document. Su sintaxis es

Let nuevoNodo = document.createElement(‘ETIQUETA HTML ELEGIDO’);

## Creando un nodo de tipo Text

Este se crea a partir del método createTextNode del objeto document su sintaxis es la siguiente

Let nuevoNodoText = document.createTextNode(‘texto del contenido en el nodo creado’);

Debemos crear la dependencia en el árbol del DOM por lo cual un nodo creado debe definirse como hijo de un nodo ya existente. La sintaxis es

nombreNodoPadre.appenchild(nombreNodoHijo)

# Seleccionar un elemento por su elemento TAG

let todosLosDiv = document.getElementsByTagName("div");

# Seleccionar un elemento por su clase HTML

Usamos el método getElementsByClassName del objeto document. Su sintaxis es:

let divRojos = document.getElementsByClassName('rojo');

# Eventos

Es una función que se ejecuta cuando sucede algo

## Función escuchadora de eventos--> addEventListener

let btn = document.querySelector('#boton');

btn.addEventListener('click',()=>{

cambiarColor();

});

### Evento click

Evento mouseOver-> cuando pasa el puntero de mouse sobre el elemento

Evento mouseOut->cuando sale de foco el puntero sobre un elemento

# Eventos del Teclado

Focus: cuando estas dentro del foco de input

Blur: cuando estas fuera del foco

Keydown: se ejecuta cuando pulsas una tecla. Para capturar cosas del teclado usamos los métodos String.fromCharCode(event.keyCode). De esta manera sé que tecla estoy pulsando.

Keypress: nos devuelve la tecla presionada

Keyup: nos permite capturar el evento cuando levanto el dedo de la tecla

Evento Load -> este evento se lanza cuando todos los elementos del DOM estiquetas html ya está cargadas. Sintaxis:

window.addEventListener('load',()=>

## Como seleccionar el contenido de un elemento HTML usando document.querySelector y su propiedad value

let tipoquery = document.querySelector("#tipoquery").value;

El atributo value se usa con el querySelector y esto nos devuelve el contenido de un elemento html

# Timers

setInterval: nos permite ejecutar una función callback cada x segundo.

Settimeout: se ejecuta solo una vez. Codigo de ejemplo 22-temporizadores.js

Sintaxis

Let tiempo = setInterval(()=>{

},tiempo);

Páginas de index.

INDEX.HTML -> códigos (tipos de datos, operadores, ventanas, ejercicios de programación, funciones, parámetros, funciones anónimas, funciones de texto, plantillas, arreglos, arreglos multidimensionales, operaciones con array, funciones para arreglos, ejercicios con arreglos )

INDEX2.HTML -> DOM

# INDEX3.HTML -> EVENTOS, TEMPORIZADORES

# Definición de un objeto (Json): son array asociativos en javascript

Json signifa javascript object notation

Let

let objeto = {

clave: valor

}

let peliculas = {

titulo: "Batman vs Superman",

anno: 2017,

pais:"EEUU"

}

# -Definición de un arreglo con objetos dentro

Let arreglo-con-objeto = [ { clave : valor } ]

const colores = [

{

color:"verde",

cantidad:15,

pais:'Argentina'

},

{

color:'rojo',

cantidad:20,

pais:'Paraguay'

}

]

# -Imprimir un atributo del objeto

Console.log(Objeto. Atributo)

# -Cargar un elemento a un objeto

Objeto. Atributo = valor-asignado

# Recorriendo un arreglo con objetos json y creando un párrafo e insertándolo en un div creado en el HTML

let coloresPais = document.querySelector('#colores');

for(let index in colores){

let parrafo = document.createElement('p');

parrafo.append(colores[index].color+" - "+colores[index].cantidad+" - "+colores[index].pais);

coloresPais.append(parrafo);

}

# LocalStorage: es como una memoria que permanece mientras navego en la página web.

//como guardar datos en el localStorage

localStorage.setItem('titulo','curso de php');

//como recuperar un elemento y lo inserto un elemento HTML

document.querySelector('#colores').innerHTML=localStorage.getItem('titulo');

## Para almacenar datos tipos objetos tengo que convertirlos json string. El localstorage no permite guardar el objeto json en formato nativo.

Metodo stringify: convierte el objeto en un json string

localStorage.setItem('datos', JSON.stringify(datosPersonales));

Recuperar el objeto guardado. Tengo que convertir el json que esta string a un json objeto usable para ello uso el método parse.

# Metodo parse

let datosPersonalesRecuperados = JSON.parse(localStorage.getItem('datos'));

console.log(datosPersonalesRecuperados);

document.querySelector('#colores').append(datosPersonalesRecuperados.nombre);

# Ejercicio completo con un formulario para agregar un elemento con un formulario, listarlo y luego borrarlo usando los métodos nativos de localStorage.

Creo un formulario html con un campo input y un botón submit. Capturo el formulario con los métodos document.querySelector e invoco el método addEventListener (escuchador de eventos) donde capturo el evento submit. Luego capturo el input con el document.querySelector con su valor cargado por teclado.

let peliculas=document.querySelector('#addpelicula').value;

Al presionar el botón submit este guarda en el localStorage usando el método setItem

localStorage.setItem(peliculas,peliculas);

## JSON

Es un formato ligero de intercambios de datos

## ¿Qué es REST?

El término ***REST*** (***Representational State Transfer***) o Transferencia de Estado Representacional, se originó en el año 2000, descrito en la [tesis](https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional) de Roy Fielding, padre de la especificación HTTP.

Un servicio ***REST*** es un conjunto de restricciones con las que podemos crear un estilo de arquitectura de software, la cual podremos usar para crear aplicaciones web respetando el protocolo HTTP.

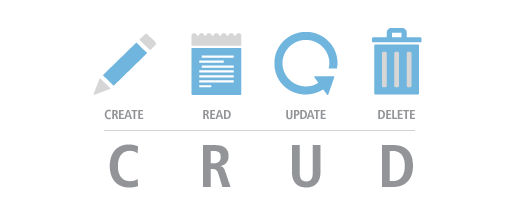
Hoy en día la mayoría de las empresas utilizan ***API REST*** para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web.

Según Fielding las restricciones que definen a un sistema ***API REST*** o ***RESTful*** como también se le conoce, son:

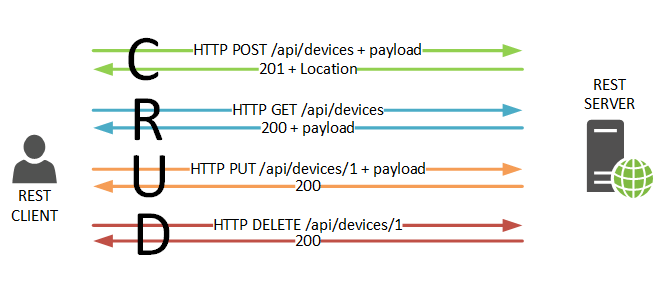
* **Cliente-Servidor**, esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se olvida de entender cómo son usados los datos que envía al cliente.
* **Sin Estado**, significa que cada petición recibida por el servidor debería ser independiente, es decir, no es necesario mantener sesiones.
* **Cacheable**, debe admitir un sistema de almacenamiento en caché, lo que evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.
* **Interfaz Uniforme**, significa una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio ***REST*** debe tener una única dirección (URL) web, a estas rutas se les da el nombre de ***endpoints***.
* **Sistema de Capas**, el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

Si creas aplicaciones web con JavaScript, las API REST probablemente serán la forma principal de obtener datos para tus aplicaciones, así como la de enviar datos hacia una base de datos.

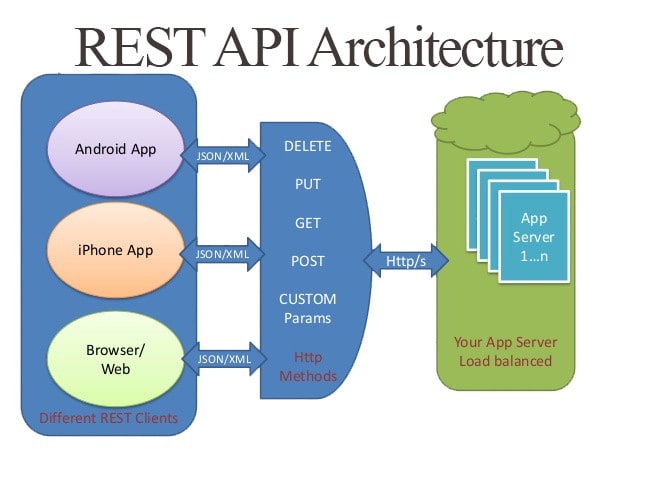
Así que... ...sí las API REST, son el nuevo CRUD del desarrollo moderno, con la ventaja de que puedes desacoplar tu la lógica de tu cliente de la del servidor y la base de datos.



| **Operaciones CRUD** | **Peticiones REST** |
| --- | --- |
| INSERT | POST |
| SELECT | GET |
| UPDATE | PUT |
| DELETE | DELETE |



Otra ventaja de este modelo de desarrollo es que puedes usar un mismo origen de datos para distintos clientes, por ejemplo un cliente nativo (iOS, Android), un cliente de escritorio (Windows) o un cliente web (Navegadores).



API RES (transferencia de estado representacional REST) => aplication programing interface – INTERFAZ DE PROGRAMACION DE APLICACIONES: son un conjunto de rutinas que nos da acceso de un software específico. Ejemplos API de cámaras web, API de geolocalización

# Fetch

Petición Ajax es una llamada a un servicio rest o api rest backend que está en la nube. Fetch esta basado en promesas. Este nuevo standard nos da una nueva alternativa para interactuar con el protocolo http para realizar llamadas al servidor. Al hacer la solicitud el servidor nos manda un response y eso lo convertimos en json.

fetch('https://jsonplaceholder.typicode.com/users') -> necesitamos dos promesas

.then(response => response.json())

# Jquery

librería en javascript

$-> el signo dólar se refiere al objeto jquery como tal

$(document) <- esto es un selector

$(document).ready <- es un evento (ready)

# AJAX -> EN JQUERY

Es una tecnología que nos permite hacer peticiones asíncronas a un servidor y recoger los resultados sin necesidad de que la pagina recargue.

load-> este método hace una petición Ajax por get y lo incrusta dentro de un div o cualquier elemento dentro de la página. Se puede incrustar un trozo de HTML o recoger JSON. En una aplicación monolítica clásica donde tenemos un frontend y un backend mezclados.

## Métodos GET y POST

draggable()->permite mover un elemento en la pagina web

# POO – TYPESCRIPT

Typescript: es un set desarrollado en Microsoft

Ssh: permite conexión a servidores

Wget: nos permite descargar ficheros de la web