**Práctico Nº 2 - JavaScript**

1.- Construir 2 clases (Persona y Empleado)

Persona debe tener:

* nombre (pública)
* edad (pública)
* altura (privada)
* peso (privada)

Empleado es una persona y tiene un sueldo (privado).

Ambas clases deben tener un método público que permita imprimir un resumen de su información.

Cree instancias de las clases y compruebe su funcionamiento.

Compruebe el polimorfismo creando una función que reciba por parámetro cualquiera de los objetos e imprima su información.

Agregue un método a la clase Empleado que permita llevar la cantidad de instancias creadas.

2.- Desarrolle un módulo fecha que pueda indicarnos la fecha y hora actual. Cree algún código que requiera el módulo y pruebe su funcionamiento en el navegador.

3.- Programe un array listaPersonas que tenga como contenido objetos personas que tengan un nombre, apellido, documento, altura y peso.

Defina una función que retorne dichos objetos.

Ej.

let listaPersonas = [{nombre:”juan”,apellido:”Gomez”,documento:”22112277”,altura:175,peso:80},{…},{…}]

Luego recorra la lista usando un bucle for para imprimir el nombre y apellido de cada persona.

Realice el mismo procedimiento usando los bucles for..in, for..of y forEach

Modifique la función creadora de objetos para que los objetos creados incorporen una función que retorne el índice de masa corporal (IMC).

Vuelva a recorrer la lista de personas imprimiendo también el IMC.

4.- Suponga que tiene una lista de valores con elementos duplicados. Por ej. [‘jose’,’juan’,’ana’,’jose’,’carlos’,’rosa’,’ana’]

Escriba al menos 2 alternativas para generar una lista sin los valores duplicados.

5.- En base al siguiente formato JSON

[

{

"id" : "978-0641723445",

"label" : ["book","hardcover"],

"name" : "The Lightning Thief",

"author" : "Rick Riordan",

"series\_t" : "Percy Jackson and the Olympians",

"genre\_s" : "fantasy",

"inStock" : true,

"price" : 12,

"pages\_i" : 384

}

,

{

"id" : "978-1857995879",

"label" : ["book","paperback"],

"name" : "Sophie's World : The Greek Philosophers",

"author" : "Jostein Gaarder",

"genre\_s" : "fantasy",

"inStock" : true,

"price" : 3,

"pages\_i" : 64

}

,

{

"id" : "978-1933988177",

"label" : ["Article","paperback"],

"name" : "Lucene in Action, Second Edition",

"author" : "Michael McCandless",

"genre\_s" : "IT",

"inStock" : true,

"price" : 30,

"pages\_i" : 475

}

]

a.- Cree una página que pueda cargar un nuevo registro de libro en el array de objetos basado en el formato JSON dado.

b.- Agregue una funcionabilidad que permita al usuario ver todos los libros registrados.

c.- Cree otra función que permita al usuario ver los libros mayores a un determinado precio.

d.- Cree otra funcionalidad que permita ver al usuario los libros que pertenecen a una determinada etiqueta.

6.- dado el siguiente objeto:

const personas = [

{

id: 1,

datos: {

nombre: "Jose",

edad: 39,

empleado: true,

hobbies: ["tenis"],

},

},

{

id: 2,

datos: {

nombre: "Manuel",

edad: 23,

empleado: true,

},

},

{

id: 3,

datos: {

nombre: "Abigail",

edad: 25,

empleado: false,

hobbies: ["basquet", "tenis", "futbol"],

},

},

{

id: 4,

datos: {

nombre: "Fabiana",

edad: 12,

empleado: true,

hobbies: ["tenis", "golf"],

},

},

];

Para cada punto realice una función flecha que retorne:

a.- los datos de la persona 3

b.- Un array con todos los datos (solo el detalle, no el id)

c.- Un array de las personas mayores a 24 años

d.- Un array de personas ordenado por el nombre de la persona

e.- Un array de personas que al menos tenga 2 hobbies

f.- Un array con todos los diferentes hobbies que existen en personas

g.- Un array de los nombres de personas que sean empleados.

**Ajax**

1.- Utilice Ajax para consumir la api <https://randomuser.me/api/> que retorna en cada petición los datos de perfil de un usuario fake.

Primero verifique en el navegador la api y compruebe que información retorna.

Utilice el visor de Json del sitio <http://jsonviewer.stack.hu/> para pegar el texto plano de json y con el visor facilitar el análisis de la respuesta.

Realice una página con un botón que permita mostrar los principales datos e imagen pequeña del perfil recuperado en forma de lista. Agregue a cada entrada un botón para eliminar esa entrada cuando sea necesario.

2.- Realice una página que permita mostrar la predicción del tiempo para ciudades de la argentina. Solicite por medio de un formulario la ciudad al usuario y actualice los datos de la predicción sin recargar el contenido de la página.

1. Acceda a la pagina <https://rapidapi.com/community/api/open-weather-map>
2. Registrese.
3. Busque el endpoint de predicción para las próximas 3 horas.
4. Verifique como funciona el endpoint (que parámetros usa y cual es el formato del resultado.
5. Note que para utilizar la api se debe proveer una API KEY que debe enviarse en los encabezados del request.

Utilice Ajax con lo aprendido de la api para recuperar la información y completar el ejercicio.

3.- Realice una página que muestre de forma aleatoria un chiste.

La pagina debe tener un botón que permita recuperar y mostrar un chiste diferente.

Verifique el uso de la api <https://rapidapi.com/LemmoTresto/api/joke3> para obtener un chiste aleatorio.

Luego utilice la api del traductor de Google para traducir el chiste recuperado.

Puede utilizar la api de <https://rapidapi.com/googlecloud/api/google-translate1> (recomendado)

o sacar una cuenta de Google Cloud gratuita por 3 meses (Es necesario crear una cuenta de facturación proporcionando medios de pago)

**Complementario**

Utilizando la API <http://hp-api.herokuapp.com/api/characters> de personajes de Harry Potter realice una página que presente cada uno de los personajes en un card que contenga la imagen, nombre y casa (house).

a.- Use una imagen por defecto para los personajes que no tengan imagen.

b.- Realice una interfaz para filtrar por raza (species), género o ambos.

c.- Actualice las cards para que los personajes que están muertos aparezcan con una X roja sobre la card