**React JS**

# Clase n° 1 – 31/07/2023

**Profesor**: Gabriel Muñoz. Correo electrónico: [gabriel.munoz@bue.edu.ar](mailto:gabriel.munoz@bue.edu.ar).

# Clase n° 2 – 02/08/2023

**Queryselector** es mejor para traer elementos.

**Queryselectorall** trae todos.

Siempre que asociamos eventos a clic se trabaja con un booleano (true o false).

1 solo “=”, le asigna ese valor a la variable. Con 2 “=”, estás preguntando.

# Clase n° 3 – 07/08/2023

Si el usuario presiona ESC o CANCELAR en el prompt, retorna NULL.

**Confirm** 🡪 Si el usuario presiona OK, devuelve TRUE, en cambio devuelve FALSE.

**Javascript** es un lenguaje IMPERATIVO 🡪 Se lee de arriba hacia abajo y de izquierda a derecha.

Es CASE SENSITIVE.

**Convenciones**:

* UpperCamelCase 🡪 La primer letra de cada palabra en mayúscula.
* lowerCamelCase 🡪 Igual que la anterior con la excepción de que la primera letra es en minúscula.
* snake\_case 🡪 Palabras separadas por guión bajo. Primer letra de cada palabra en minúscula.
* Constantes 🡪 En mayúscula.
* trains-case 🡪 Palabras separadas por guión medio. Primer letra con mayúscula.

Una variable es una forma de almacenar un valor para utilizar más tarde. Se puede configurar y restablecer a cualquier tipo.

Se pueden declarar variables sin asignar el dato o asignarlo después.

No se pueden utilizar las palabras claves del lenguaje como nombres de variables.

**LET** es la forma de declarar variables que se vá a utilizar en el curso.

Hasta ECMASCRIPT 5 se utilizaba VAR para declarar variables.

LET se introduce a partir de ECMASCRIPT 6. Es similar a VAR, pero difiere en el scope.

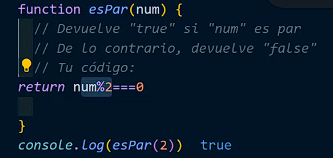
**Método NUMBER** 🡪 Convierte en número un dato.

**Realizar la ejercitación que estará subida al Drive y al repositorio. Carpeta “Ejercitaciones”.**

# Clase n° 4 – 09/08/2023

# Clase n° 5 – 14/08/2023

Se puede simplificar las condiciones incluyéndolas en lo que se declara en la función, obviando el if-else. Sobre todo, en comparaciones.



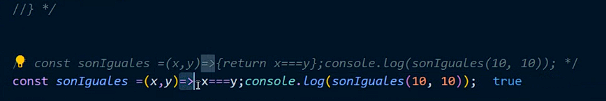
Para elevar al cuadrado 🡪 “num\*\*2”, “num\*num”, “Math.pow(num,exponente)

**Math.round** redonda hacia el número entero más cercano. En caso de encontrarse en “.5”, redondea hacia arriba.

**Math.floor** redondea siempre hacia abajo y **Math.ceil** redondea siempre hacia arriba.

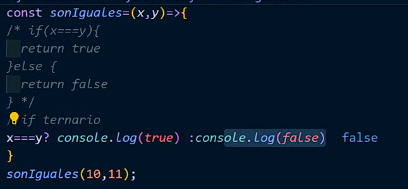
**Math.random()** NO lleva parámetro. Se multiplica por 10 para que dé un número entero y se lo redondea con Math.round.

Backticks 🡪 ALT+96

**Función flecha**

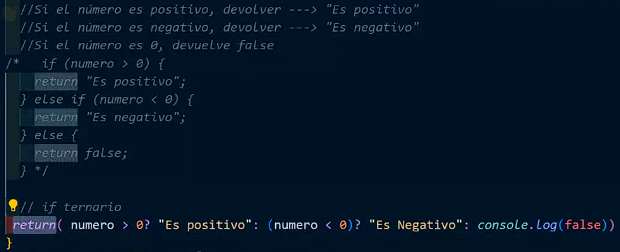
Si hay sólo una línea de ejecución, se puede obviar el return y las llaves. Si recibe sólo un parámetro, se pueden sacar los paréntesis.

**IF ternario**



El RETURN va delante de todo.

IF ternario con más de 1 condición



# Clase n° 6 – 16/08/2023

**Ejercicio**: modificar todas las funciones de la ejercitación a función flecha e if ternario.

En las operaciones, la multiplicación tiene prioridad sobre la división. No es necesario separar en términos.

**Number()** 🡪 Es lo mismo que Parseint pero acotado.

**Función esVocal con ternario** 🡪 const esVocal = (letra) => letra = letra.toLowerCase(); letra.length !== 1 ? 'Dato incorrecto' : letra === 'a' || letra === 'e' || letra === 'i' || letra === 'o' || letra === 'u' ? 'es vocal' : 'Dato incorrecto'

**Hacer el pack de ejercitaciones n° 2.**

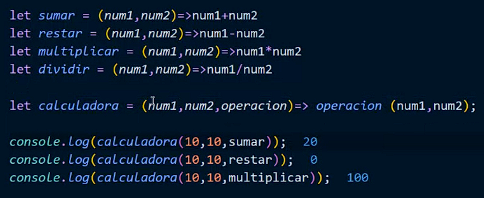
**Bucle SWITCH**



Es importante utilizar el **break** en cada caso para que no siga recorriendo el Switch después de haber matcheado con el caso.

**Callbacks**

Consiste en llamar a una función como parámetro de otra función.

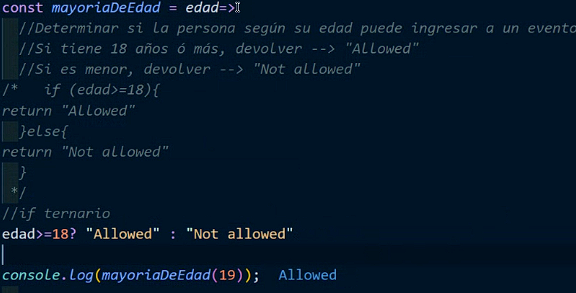


**Buscar el concepto de “callback hell”.**

# Clase n° 7 – 23/08/2023

Al método **Math.max** se le pasan 2 parámetros y devuelve el mayor.

Ejemplo de función flecha con if ternario y return implícito.



Al tener un return en el switch, el break nunca va a entrar. El return tiene un break implícito.

++num devuelve el número. num++ no lo devuelve.

**Hacer los ejercicios (ejercitación 1 y 2) y leer la teoría de React.**

# Clase n° 8 – 28/08/2023

Los métodos son funciones que me permiten ejecutar cosas con diferentes tipos de datos.

**Métodos de arrays**:

Slice (rebanar) 🡪 array.slice(indicedesde.indicehasta). Si no se coloca el segundo argumento, se obtiene hasta el final. NO MODIFICA EL ARRAY ORIGINAL.

Unshift (inserta un dato al principio) 🡪 array.unshift(dato). MODIFICA EL ARRAY ORIGINAL.

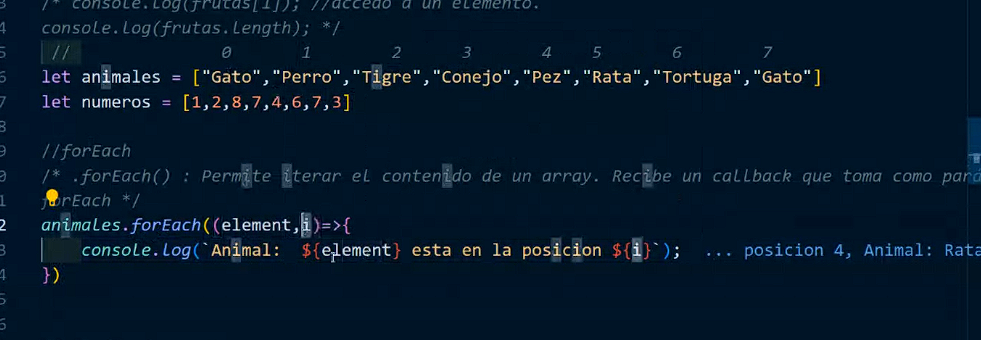
Shift (quita un dato al principio) 🡪 array.shift(). MODIFICA EL ARRAY ORIGINAL.

Pop (elimina un dato al final) 🡪 array.pop(). MODIFICA EL ARRAY ORIGINAL.

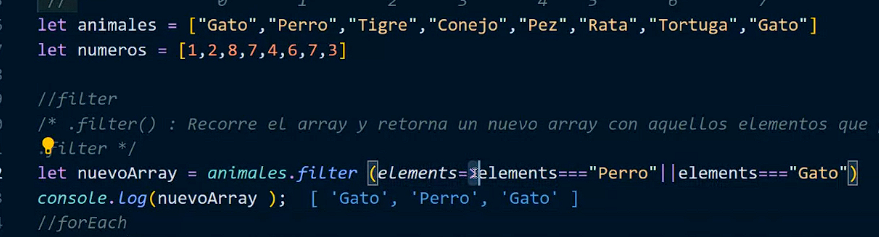
Push (inserta un dato al final) 🡪 array.push(). MODIFICA EL ARRAY ORIGINAL.

Find (recorre el array y retorna la primer coincidencia del elemento que se busca) 🡪 nuevoarray = array.find(element => element < 3).

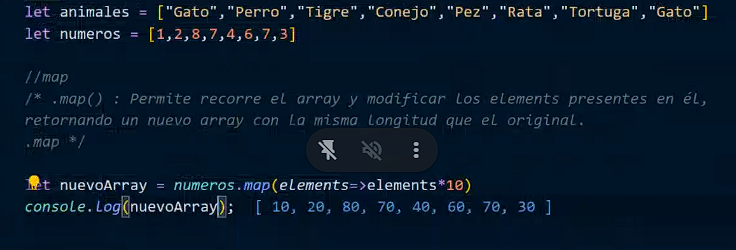
Foreach:



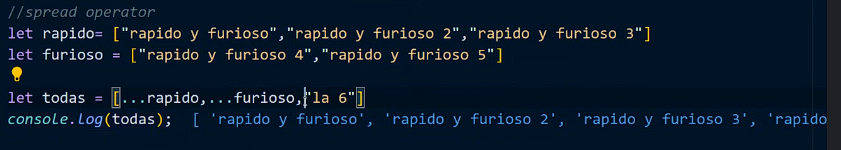
Filter. Retorna un nuevo array con los elementos que están dentro de la condición.



**Map (muy importante)**



Spread operator



También se puede añadir al array “…rapido” e incluir todos los datos de ese array.

**Realizar la ejercitación JS 2 y métodos de arrays.**

# Clase n° 9 – 30/08/2023

**Métodos de arrays**

Splice (índice de inicio, cantidad a borrar, dato a insertar, dato a insertar…) 🡪 elimina datos y reemplaza por otros datos. Si se coloca 0 en el segundo argumento, no borra nada.



**Objetos**:

Poseen pares clave:valor separados por COMA (,) y encerrado todo entre llaves ({ }).

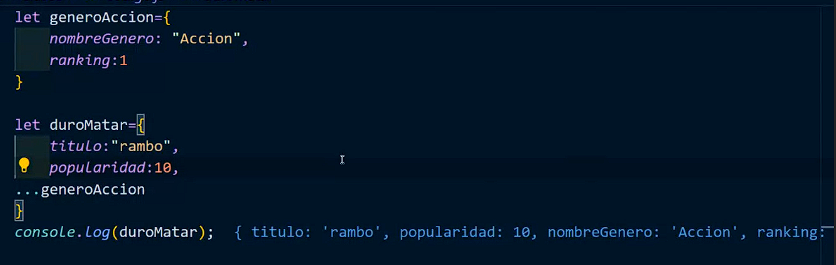
Se puede acceder a sus valores con notación de punto (objeto.clave).

Se puede acceder a sus valores con notación de corchete (objeto[“clave”]).

Un método es una función que está dentro de un objeto (Por ejemplo, objeto Math).

THIS no funciona dentro de un método utilizado con función flecha.

**Spread operator en objetos:**



**Destructuración**

Extraer o destructurar datos de algo que está guardado de forma estructurada.

Lo extraigo y lo guardo en una variable.



En la destructuración de un objeto, el nombre de la variable a asignar el dato debe ser el mismo que la clave del objeto.



Si se quiere asignar otro nombre a la variable, se coloca “:nombre” luego del nombre de la clave del objeto.

# Clase n° 10 – 04/09/2023

Cada navegador posee un motor v8 que permite ejecutar código Javascript.

Método GET 🡪 Pide información.

Método POST 🡪 Envío información nueva.

Método PUT 🡪 Actualizo información existente.

Método DELETE 🡪 Elimino información.

MIDDLEWARES 🡪 Intermediarios entre cliente y servidor. (Seguridad, por ejemplo).

JSON 🡪 Lenguaje que permite comunicarme entre cliente-servidor-API. JavaScript Object Notation.

El “.then” toma la información que le precede y lo guarda en una variable y posteriormente hace algo con eso.

# Clase n° 12 – 25/09/2023

El “package.JSON” es la documentación de mi API. Todo lo que tengo instalado y las versiones respectivas.

La carpeta “node modules” contiene todos los módulos que instalé con npm.

“gitignore” contiene todas las cosas que no voy a subir al repositorio.

React 🡪 Trabaja con lenguaje JSX (mezcla entre HTML y Javascript).

# Clase n° 14 – 02/10/2023

“React.strictmode” es para que te tire avisos en consola para optimizar el proyecto.