ASOCIACIÓN INFORMÁTICOS UTE – USACH A.G.

DAVID HERNÁNDEZ MATURANA

PALEO INFORMÁTICO
DIRECTOR AGI UTE-USACH

DAVID.HERNANDEZM@USACH.CL

+56998246832

UNIVERSIDAD DE SANTIAGO DE CHILE

Asociación de Informáticos UTE-USACH A.G.

/PRIMEROS PASOS CON JAVASCRIPT

Curso de FrontEnd

Sábado 10 de Junio 2023

/AGENDA

/01 /bienvenida

/02 /revisión tarea nº 4

/03 Condicionales JS

/04 Métodos JS

/05 /Callback Historia

/06 /Promesas

/07 /Then & Catch

/08 /Async - Await

/09 /tarea nº5

Horario

09hrs - Entrada
09:30hrs - Módulo 1
10:00hrs - Módulo 2
10:15hrs - Módulo 3
10:30hrs - Módulo 4
11hrs - Break
11:15hrs - Módulo 5
11:30hrs - Módulo 6
12:00hrs - Módulo 7
12:30hrs - Módulo 8
13:00hr - Cierre

UNIVERSIDAD DE SANTIAGO DE CHILE

Condicionales

JS

En programación, si queremos ejecutar un código cuando se cumple una condición o condiciones particulares, entonces, en tales casos, hacemos uso de algo llamado declaraciones if.

Por ejemplo, imagina cómo podríamos hacer un programa que nos diga si un número es mayor o menor a diez. Si es mayor a 10 debería imprimir una cosa, pero si es menor debería imprimir otra.

```
1 if (<condición>) {
2   // código que se ejecuta si se cumple la condición
3 }
```

UNIVERSIDAD DE SANTIAGO DE CHILE

If / else / else if / operador ternario

If: Ejecuta una sentencia si una condición especificada es evaluada como verdadera. Si la condición es evaluada como falsa, otra sentencia puede ser ejecutada.

Else: La cláusula else (no obligatoria) sirve para indicar instrucciones a realizar en caso de no cumplirse la condición

```
if (condición) sentencia1 [else sentencia2]
```

UNIVERSIDAD DE SANTIAGO DE CHILE

If / else / else if / operador ternario

Else If: Habrá momentos en los que desees probar múltiples condiciones. Ahí es donde entra el bloque else if.

```
if (la condición 1 es verdadera) {
   // el código se ejecuta
} else if (la condición 2 es verdadera) {
  // el código se ejecuta
} else {
   // el código se ejecuta
}
```

UNIVERSIDAD DE SANTIAGO DE CHILE

If / else / else if / operador ternario

JS

Operador ternario: El operador condicional (ternario) es el único operador en JavaScript que tiene tres operandos. Este operador se usa con frecuencia como atajo para la instrucción **if**.

```
condición ? expr1 : expr2
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Métodos en JS

Métodos: Un método es una función la cual es propiedad de un Objeto. Existen dos tipos de métodos: Métodos de Instancia los cuales son tareas integradas realizadas por la instancia de un objeto, y los Métodos Estáticos que son tareas que pueden ser llamadas directamente en el constructor de un objeto.

Numéricos

```
//métodos para números
    // parseFloat() *
    // parseInt() *
    // toFixed()
    // toPrecision()
```

Strings

```
//métodos para strings
    // .charAt()
    //.concat(variable2);
    //.indexOf();*
    //.lastIndexOf();*
    //.replace("texto a encontrar","texto que sustituye el original")
    //.slice("donde empiezo", "donde acabo");
```

Arrays

```
//métodos para arrays
    //.length *
    //.push()
    //.unshift()
```

Ejemplo: https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_tofixed
https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_number_parsefloat1

UNIVERSIDAD DE SANTIAGO DE CHILE

Ciclo for

For: Crea un bucle que consiste en tres expresiones opcionales, encerradas en paréntesis y separadas por puntos y comas, seguidas de una sentencia ejecutada en un bucle.

```
for (var i = 0; i < 9; i++) {
   n += i;
   mifuncion(n);
}
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

UNIVERSIDAD DE SANTIAGO DE CHILE

For Of

For Of: La sentencia sentencia for of ejecuta un bloque de código para cada elemento de un objeto iterable, como lo son: String, Array, objetos similares a array (por ejemplo, arguments or NodeList), TypedArray, Map, Set e iterables definidos por el usuario.

```
for (variable of iterable) {
  statement
}
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

UNIVERSIDAD DE SANTIAGO DE CHILE

For Each

For Each: El método forEach() ejecuta la función indicada una vez por cada elemento del array.

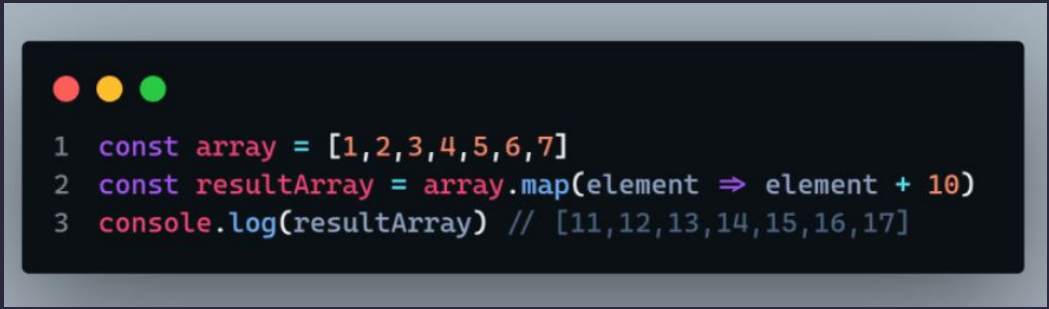
```
1  const array = [1,2,3,4,5,6,7]
2  const resultArray = array.filter(element => element === 3 || element === 7)
3  console.log(resultArray) // [3, 7]
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

UNIVERSIDAD DE SANTIAGO DE CHILE

Map

Map: El método map() crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.



```
1  const array = [1,2,3,4,5,6,7]
2  const resultArray = array.map(element => element + 10)
3  console.log(resultArray) // [11,12,13,14,15,16,17]
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

Diferencias

¿Qué diferencias hay entre Map y ForEach?

UNIVERSIDAD DE SANTIAGO DE CHILE

Filter

Filter: El método filter() crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada.

```
1  const array = [1,2,3,4,5,6,7]
2  const resultArray = array.filter(element => element === 3 || element === 7)
3  console.log(resultArray) // [3, 7]
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

UNIVERSIDAD DE SANTIAGO DE CHILE

Find

Find: El método find() devuelve el valor del primer elemento del array que cumple la función de prueba proporcionada.

```
1  const array = [1,2,3,4,5,6,7]
2  const resultArray = array.find(element => element > 3 )
3  console.log(resultArray) // [4]
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

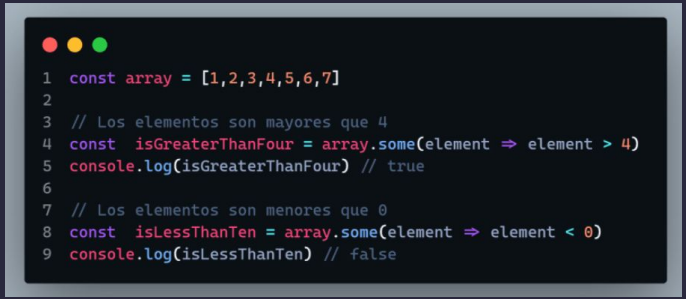
Diferencias

¿Qué diferencias hay entre Filter y Map?

UNIVERSIDAD DE SANTIAGO DE CHILE

Some

Some: El método some() comprueba si al menos un elemento del array cumple con la condición implementada por la función proporcionada.



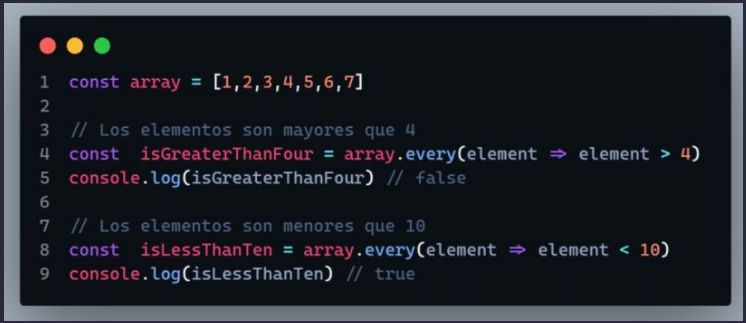
```
const array = [1,2,3,4,5,6,7]

// Los elementos son mayores que 4
const  isGreaterThanFour = array.some(element => element > 4)
console.log(isGreaterThanFour) // true

// Los elementos son menores que 0
const  isLessThanTen = array.some(element => element < 0)
console.log(isLessThanTen) // false
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

UNIVERSIDAD DE SANTIAGO DE CHILE

Every

Every: Determina si todos los elementos en el array satisfacen una condición.



```
1  const array = [1,2,3,4,5,6,7]
2
3  // Los elementos son mayores que 4
4  const  isGreaterThanFour = array.every(element => element > 4)
5  console.log(isGreaterThanFour) // false
6
7  // Los elementos son menores que 10
8  const  isLessThanTen = array.every(element => element < 10)
9  console.log(isLessThanTen) // true
```

1. ¿A qué se aplica este método? (Tipo de dato)
2. ¿Qué recibe este método? (Parámetro)
3. ¿Qué se obtiene? (Retorno o resultado)

Promesas

Una Promise (promesa en castellano) es un objeto que representa la terminación o el fracaso de una operación asíncrona.

Esencialmente, una promesa es un objeto devuelto al cuál se adjuntan funciones callback, en lugar de pasar callbacks a una función.

```
function exitoCallback(resultado) {
  console.log("Archivo de audio disponible en la URL " +
resultado);
}

function falloCallback(error) {
  console.log("Error generando archivo de audio " + error);
}

crearArchivoAudioAsync(audioConfig, exitoCallback, falloCallback);
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Constructor new promise

El constructor Promise se utiliza principalmente para ajustar funciones que aún no admiten promesas.

```
const myFirstPromise = new Promise((resolve, reject) => {
  // hacer algo asincrónico que eventualmente llame a:
  //
  // resolver (algúnValor) // cumplido
  // or
  // rechazar ("motivo de falla") // rechazado
});
```

Then / Catch

Se utiliza para el manejo de promesas. El método catch() retorna una Promise y solo se ejecuta en los casos en los que la promesa se marca como Reject. Se comporta igual que al llamar Promise.prototype.then(undefined, onRejected) (de hecho, al llamar obj.catch(onRejected) internamente llama a obj.then(undefined, onRejected)).

```
p.catch(onRejected);

p.catch(function(reason) {
   // rejection
});
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Async / Await

Se utiliza para trabajar con promesas de forma asíncrona.

UNIVERSIDAD DE SANTIAGO DE CHILE

Operadores lógicos

Operador	Nombre	Ejemplo	Descripción
+	Suma	`5 + 6`	Suma dos números
-	Substracción	`7 - 9`	Resta dos números
*	Multiplicación	`6 * 3`	Multiplica dos números
/	División	`4 / 8`	Divide dos números
%	Módulo: el resto después de la división	`7 % 2`	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Incremento.	`a++`	Suma 1 al contenido de una variable.
--	Decremento.	`a--`	Resta 1 al contenido de una variable.
-	Invierte el signo de un operando.	`-a`	Invierte el signo de un operando.

UNIVERSIDAD DE SANTIAGO DE CHILE

Operadores lógicos

```
console.log( 0 == 0);                    // true
console.log( "" == 0 );                  // true
console.log( false == 0);                // true
console.log( undefined != 0);            // true
console.log( null == 0);                 // false


console.log('1' === 1);                  // false
console.log(2 === 2);                    // true
console.log('abc' !== 123);              // true
```

Tarea Nº4 (opcional)

Desplegar la tarea 3 en GitHub Pages

Crear un repositorio en GitHub y enviarlo mediante formulario:
https://www.softwarelibrechile.cl/G23-S1-04-Tarea

UNIVERSIDAD DE SANTIAGO DE CHILE

Tarea Nº5

Hacer las funciones que sean necesarias para:

Obtener el promedio de notas de un alumno considerando que la suma de notas debe ser el retorno de una función y el promedio el retorno de otra función. Las notas son: 6,8,9,2,5,10.

Crear un repositorio en GitHub y enviarlo mediante formulario: https://www.softwarelibrechile.cl/G23-S1-05-Tarea

UNIVERSIDAD DE SANTIAGO DE CHILE

Tarea Nº6

1. Mostrar en consola la secuencia de Fibonacci:
   a. Entre los números 0 y 1000.
   b. Números pares entre 0 y 1000.
   c. Números impares entre 0 y 1000.

   Hint: Puedes usar recursividad o algún ciclo o método iterador visto en clase

2. Del siguiente arreglo de strings retornar otro arreglo con todo a mayúsculas.

3. Del siguiente arreglo de objetos, retornar otro arreglo con los pokemon tipo fuego.

   Nota: Arreglos en la siguiente diapositiva.

Enviar la tarea usando el formulario: **softwarelibrechile.cl/G22-S2-06-Tarea**

UNIVERSIDAD DE SANTIAGO DE CHILE

//2•– Del siguiente arreglo de strings retornar otro arreglo con todo a mayúsculas.

```
let pokemon =

[
    'Pikachu',
    'Charmander',
    'Bulbasaur',
    'Squirtle'
]
```

//3•– Del siguiente arreglo de objetos, retornar otro arreglo con los pokemon tipo fuego.

```
let pokemon = [

    {
        nombre: 'Pikachu',
        tipo: 'Electrico'
    },
    {
        nombre: 'Charmander',
        tipo: 'Fuego',
    },
    {
        nombre: 'Bulbasaur',
        tipo: 'Planta'
    },
    {
        nombre: 'Squirtle',
        tipo: 'Agua'
    },
    {
        nombre: 'Charmeleon',
        tipo: 'Fuego'
    },
    {
        nombre: 'Weedle',
        tipo: 'bicho'
    },
    {
        nombre: 'Charizard',
        tipo: 'Fuego'
    }
]
```

/Porcentajes a cumplir en el curso

A continuación les dejamos información acorde al % de asistencia, entregas de tarea y proyecto final, todo esto de **carácter obligatorio** para obtener el certificado de aprobación.

/CLASES	/TAREAS	/PROYECTO FINAL
ASISTENCIA 80%	**ENTREGAS 80%**	**ENTREGA 100%**
11 clases	8 tareas	1 proyecto
Debe existir asistencia mínima de 9 clases	Deben entregarse mínimo 7 tareas o más	El proyecto final se hará de forma grupal, 2 personas, ambas obteniendo la misma nota.

UNIVERSIDAD DE SANTIAGO DE CHILE

/APROBACIÓN Y ENTREGA CERTIFICADO

/TAREAS

El promedio de las notas de la tareas equivale a un

/50%

+

/PROYECTO FINAL

La nota en el proyecto final equivale a un

/50%

=

/NOTA FINAL

La nota debe ser igual o mayor a 6

/100%

<gracias!>