



ASOCIACIÓN INFORMÁTICOS UTE – USACH A.G.



DAVID HERNÁNDEZ MATURANA

**PALEO INFORMÁTICO
DIRECTOR AGI UTE-USACH**

DAVID.HERNANDEZM@USACH.CL

+56998246832



/PRIMEROS PASOS CON JAVASCRIPT

Curso de FrontEnd

Sábado 11 de Junio 2022





/AGENDA



/01

/bienvenida

/02

/revisión tarea n° 4

/03

Condicionales JS

/04

Métodos JS

/05

/Callback Historia

/06

/Promesas

/07

/Then & Catch

/08

/Async - Await

/09 /tarea n°5



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

Revisión de tarea 4

- **Función recursiva que realice lo siguiente:**
 - Mostrar en consola los números del 1 al 100 de 1 en 1.
 - Mostrar en consola los números del 1000 al 500 de 2 en 2.
- **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.



Condicionales

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 }
```

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]
```

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  
}
```


If / else / else if / operador ternario

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
```

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()
```

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)

For Of

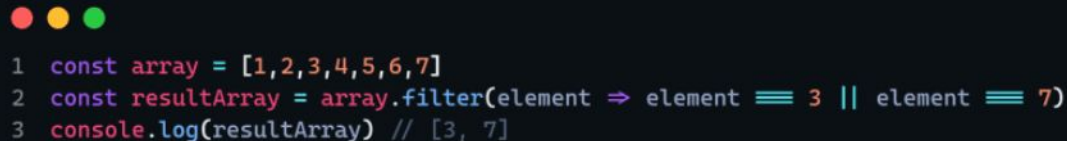
For Of: La sentencia `for of` ejecuta un bloque de código para cada elemento de un objeto iterable, como lo son: `String`, `Array`, objetos similares a array (por ejemplo, `arguments` or `NodeList`), `TypedArray`, `Map`, `Set` e iterables definidos por el usuario.

```
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)

For Each

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

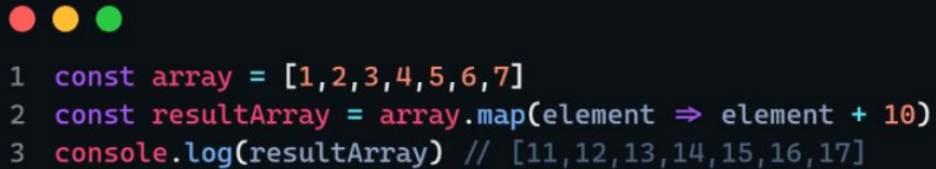


```
1 const array = [1,2,3,4,5,6,7]
2 const resultArray = array.filter(element => element === 3 || element === 7)
3 console.log(resultArray) // [3, 7]
```

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

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?

Filter

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

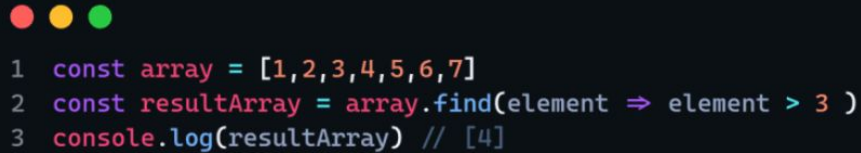


```
1 const array = [1,2,3,4,5,6,7]
2 const resultArray = array.filter(element => element === 3 || element === 7)
3 console.log(resultArray) // [3, 7]
```

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

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?

Some

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

```
1 const array = [1,2,3,4,5,6,7]
2
3 // Los elementos son mayores que 4
4 const isGreaterThanFour = array.some(element => element > 4)
5 console.log(isGreaterThanFour) // true
6
7 // Los elementos son menores que 0
8 const isLessThanTen = array.some(element => element < 0)
9 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)

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);
```

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 asíncronico 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
});
```

Async / Await

Se utiliza para trabajar con promesas de forma asíncrona.

```
function scaryClown() {  
  return new Promise(resolve => {  
    setTimeout(() => {  
      resolve('😱');  
    }, 2000);  
  });  
}  
  
async function msg() {  
  const msg = await scaryClown();  
  console.log('Message:', msg);  
}  
  
msg(); // Message: 😱 ← after 2 seconds
```



Tarea N°5

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 pokemones tipo fuego.

Hacer un repositorio en GitHub y enviarlo a:
contacto@softwarelibrechile.cl

Tarea N°5

```
//1- Del siguiente arreglo de strings retornar otro arreglo con todo a mayúsculas.
let pokeones = ['Pikachu','Charmander','Bulbasaur','Squirtle']

//2- Del siguiente arreglo de objetos, retornar otro arreglo con los pokemones tipo fuego.
let pokemones = [
  {
    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'
  }
]
```

Canales de comunicación

Slack (principal)

www.softwarelibrechile.cl/slack

WhatsApp

www.softwarelibrechile.cl/whatsapp

Programas

Visual Studio Code

www.softwarelibrechile.cl/vscode

Trello

www.trello.com

Codesandbox

codesandbox.io

Programa

- 01 - Fundamentos
- 02 - Estilos y diagramación
- 03 - Estilos y diagramación
- 04 - Introducción a JS
- 05 - Primeros pasos a JS
- 06 - Funciones, API's, Manejo de errores
- 07 - Programación JS
- 08 - ReactJS
- 09 a 12 - Práctico grupal

Curso FrontEnd: Calendario de Clases

MES	DIA
MAYO	07
	14
	28
JUNIO	04
	11
	18
	25
JULIO	02
	09
	23
	30
AGOSTO	06

Curso FrontEnd: Contenidos

CLASE	CAPÍTULO	CONTENIDOS
SÁBADO 1	FUNDAMENTOS	<ol style="list-style-type: none">1. BIENVENIDA AGI2. ORGANIZACIÓN DE TAREAS3. TRELLO4. TOMA DE REQUERIMIENTOS BASE5. METODOLOGÍAS DE TRABAJO6. HTML, CSS, JS7. BOOTSTRAP INSPECTOR ELEMENTOS.8. TAREA N° 1

Curso FrontEnd: Contenidos

SÁBADO 2	ESTILOS Y DIAGRAMACIÓN CSS PARTE 1	<ol style="list-style-type: none">1. REVISIÓN TAREA N°12. BEM, SASS, FLEXBOX3. PREFIJOS, NAVEGADOR, MEDIAQUERYS4. Posicionamiento, layout5. Cajas (tipos, posición)6. Responsividad7. Tarea N° 2
SÁBADO 3	ESTILOS Y DIAGRAMACIÓN CSS PARTE 2	<ol style="list-style-type: none">1. Revisión Tarea N° 22. GRID3. Introducción a GIT4. Visionamiento de código (GITHUB)5. Despliegue (GITHUB Pages)6. Tarea N° 3

Curso FrontEnd: Contenidos

SÁBADO 4	INTRODUCCION JS	<ol style="list-style-type: none">1. Revisión Tarea N° 32. ¿Qué es JS?3. ¿Cómo usar JS?)4. JQUERY, AJAX5. Tipos de datos6. Funciones7. Tarea N° 4
SÁBADO 5	PRIMEROS PASOS CON JS	<ol style="list-style-type: none">1. Revisión Tarea N° 42. ARRAYS, CALLBACK3. Ciclos4. Tarea N° 5

Curso FrontEnd: Contenidos

SÁBADO 6	FUNCIONES, API'S, MANEJO DE ERRORES	<ol style="list-style-type: none">1. Revisión Tarea N° 52. Tarea N° 6
SÁBADO 7	PROGRAMACIÓN JS	<ol style="list-style-type: none">1. Revisión Tarea N° 62. herencias3. Tarea N° 7
SÁBADO 8	INTRODUCCIÓN A REACTJS	<ol style="list-style-type: none">1. Revisión Tarea N° 72. Definición proyecto final3. Tarea N° 8
SÁBADO 9	PRÁCTICO (GRUPAL)	<ol style="list-style-type: none">1. Revisión Tarea N° 82. Proyecto final}3. Tarea N° 9

Curso FrontEnd: Contenidos

SÁBADO 10	PRÁCTICO (GRUPAL)	<ol style="list-style-type: none">1. Revisión Tarea N° 92. Presentación y entrega
SÁBADO 11	PROYECTO FINAL (GRUPAL)	<ol style="list-style-type: none">1. Presentación y entrega
SÁBADO 12	ENTREGA CERTIFICADOS APROBACIÓN	<ol style="list-style-type: none">1. Presentación y entrega2. Entrega Certificados3. Cierre

Curso FrontEnd: Relatores



Scarlet Melgarejo

Director de front-end en PropulsoW, Ayudante en Desafío Latam, voluntario Mozilla Chile ayudante en taller Joomla dictado en la Usach el 2011 - 2015 2015.

<https://www.linkedin.com/in/scarlett-melgarejo-venegas-38805626/>



Gonzalo Flemming

(Tech Lead Frontend en Global 66 | Profesor de Programación en Desafío Latam

<https://www.linkedin.com/in/gfleming-garrido/>



Sebastian Becerra

Director de Operaciones en PropulsoW, Ingeniero Informático Duoc UC, voluntario Mozilla Chile ayudante en taller Joomla dictado en la Usach el 2014-2015

<https://www.linkedin.com/in/sebaebc/>



Cristian Pavés

Front End en Propulsow y estudiante de Ingeniería Informática y telecomunicaciones en Duoc UC

<https://www.linkedin.com/in/cristian-pavez-0307811b8/>

Curso FrontEnd: Relatores



Dariana Gómez Reyes

Ing. de Sistemas y
FrontEnd Developer en
Entel.

<https://www.linkedin.com/in/dariana-g%C3%B3mez/>



Oriana Sanabria

Frontend Developer en CoreBiz

<https://www.linkedin.com/in/orianasabri/>



José Manuel Luman

Technical Leader en Ecomsur

<https://www.linkedin.com/in/jos%C3%A9-manuel-luman-salazar-363371175/>

<gracias!>