



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

/Métodos JS

/03

/Promesas

/04

/Then & Catch (Manejo de errores)

/05

/Async - Await

/06

API'S

/07

Métodos HTTP

/08

Fetch / Axios

/09

Tarea 6

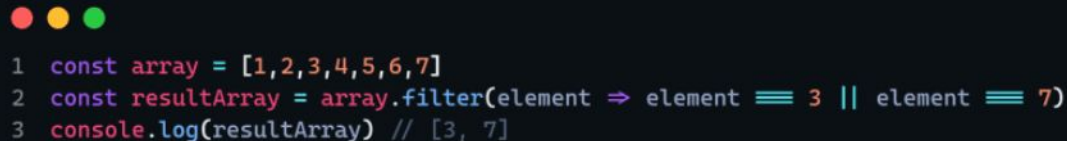


Horario

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

For Each

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

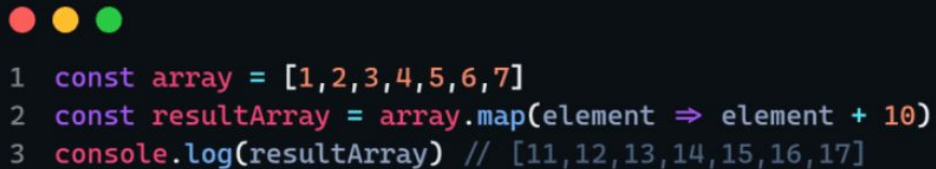


```
1 const array = [1,2,3,4,5,6,7]
2 const resultArray = array.filter(element => element === 3 || element === 7)
3 console.log(resultArray) // [3, 7]
```

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.

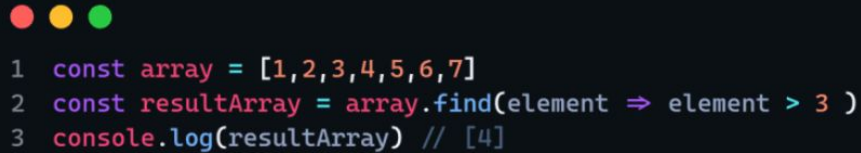


```
1 const array = [1,2,3,4,5,6,7]
2 const resultArray = array.filter(element => element === 3 || element === 7)
3 console.log(resultArray) // [3, 7]
```

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

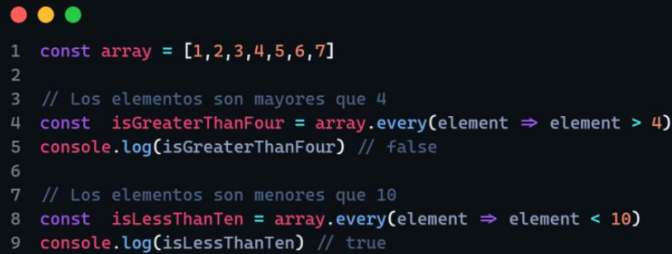
Some: El método `some()` comprueba si al menos un elemento del array cumple con la condición implementada por la función proporcionada.

```
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
```





API'S

Es el acrónimo de **Application Programming Interface**, o en español, Interfaz de Programación de Aplicaciones. Se puede definir como una interfaz que favorece la comunicación entre dos sistemas o plataformas diferentes, permitiendo agregar diversas funciones a sitios web y aplicaciones.

La finalidad de las API es unir sus productos y sus servicios con otros sin saber cómo se implementan, lo que ayuda al desarrollo de las apps además de ahorrar tiempo y dinero.

Para los ejemplos prácticos utilizaremos:
<https://pokeapi.co/>



Métodos de petición HTTP

HTTP define un conjunto de **métodos de petición** para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados *HTTP verbs*. Cada uno de ellos implementan una semántica diferente, pero algunas características similares son compartidas por un grupo de ellos.



200

OK

Métodos de petición HTTP

GET

El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

PUT

El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

POST

El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

DELETE

El método DELETE borra un recurso en específico.

Fetch / Axios

La **API Fetch** proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global `fetch()` que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

Axios es un cliente HTTP *basado en promesas* para `node.js` y el navegador. Es *isomorfo* (puede ejecutarse en el navegador y `nodejs` con el mismo código base). En el lado del servidor usa el módulo nativo `http` de `node.js`, mientras que en el lado del cliente (navegador) usa `XMLHttpRequests`.

Fetch / Axios

Fetch

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

Axios

```
axios.get('http://webcode.me').then(resp => {  
  
  console.log(resp.data);  
});
```

Tarea N°6

1. Consumir el siguiente endpoint <https://pokeapi.co/api/v2/pokemon/> y mostrar en el front lo siguiente:
 - a) Cards que contengan los 20 primeros pokemones (imagen y nombre del pokemon)
 - b) Utilizar Async / Await para trabajar las promesas de forma asíncrona
 - c) Usar Axios o Fetch para realizar la solicitud al endpoint mencionado
 - d) Ocupar Try / Catch para el manejo de errores

Crear un repositorio en GitHub y colocarlo en el siguiente form de google:

★ softwarelibrechile.cl/tarea-6

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

/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%
12 clases	9 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, de a 2 personas, ambas obteniendo la misma nota.

/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%

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!>