



APIs



{desafío}
latam_



**Activen las cámaras los que puedan y
pasemos asistencia**





Inicio

{desafío}
latam_



`/* Consultar una API REST y mostrar los resultados en el DOM
*/`

`/* Realizar un request a una API con JS mostrando por consola
la respuesta */`

`/* Modificar el DOM con los resultados obtenidos a partir de
un pedido */`

`/* Manejar errores con try and catch en caso de problemas
con la API */`

Objetivos

Activación de conceptos

Contesta la pregunta correctamente y gana un punto

Instrucciones:

- Se realizará una pregunta, el primero en escribir “YO” por el chat, dará su respuesta al resto de la clase.
- El docente validará la respuesta.
- En caso de que no sea correcta, dará la oportunidad a la segunda persona que dijo “Yo”.
- Cada estudiante podrá participar un máximo de 2 veces.
- Al final, el/la docente indicará el 1º, 2º y 3º lugar.
- Esta actividad no es calificada, es solo una dinámica para recordar los conceptos clave para abordar esta sesión.





Activación de conceptos



¿Qué muestra el siguiente código?

```
const arreglo1 = [1,2,3,4,5]  
arreglo1.push('hola')  
console.log(arreglo1)
```



Activación de conceptos



¿Qué muestra el siguiente código?

```
const arreglo1 = [1,2,3,4,5]  
arreglo2 = arreglo1  
arreglo2[0] = 5  
console.log(arreglo1)
```




Activación de conceptos



¿Qué significa que un objeto sea mutable?



Activación de conceptos



¿Qué muestra el siguiente código?

```
const arreglo6 = [1,2,3,4,5]  
arreglo6.splice(2, 1)  
console.log(arreglo6)
```



Activación de conceptos



¿Qué muestra el siguiente código?

```
const arr1 = [4,1,2,3]
const ordenado = arr1.sort((x,y)
=> y - x)
console.log(ordenado)
```



Activación de conceptos



¿Cuál es la diferencia entre `.forEach` y `.map`? ¿Qué devuelve cada uno?



Activación de conceptos



Primer lugar:



Segundo lugar:



Tercer lugar:



Desarrollo

{desafío}
latam_



/* APIs REST */

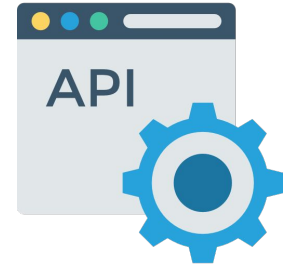
APIs

¿Qué son las API?

Una API es como una página web pero sirve para ser consultada por programas en lugar de personas. En esta clase aprenderemos a escribir estos programas con JS y utilizaremos los datos obtenidos para actualizar la página

Hay APIs para distintos propósito:

- Datos del clima
- Servicios de geolocalización
- Indicadores financieros
- Actualizar redes sociales
- Automatización de acciones en campañas de marketing



Adicionalmente es relativamente sencillo crear una API cuando tienes nociones de backend.

APIs

Cliente servidor



El cliente puede ser el navegador, una herramienta como Thunder Cliente o Postman o un script o programa hecho por nosotros en JavaScript u otro lenguaje de programación.

El servidor es otro programa que podremos acceder vía una dirección de internet utilizando el cliente



Utilizaremos el término Request para referirnos a hacer un pedido.

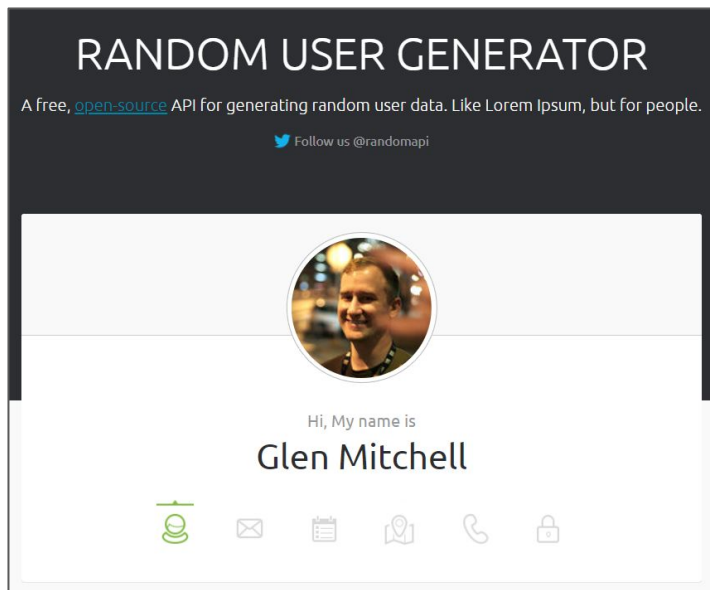
APIs

Usando nuestra primera API

<https://randomuser.me/> es una página que genera data al azar de personas. En <https://randomuser.me/api> podemos encontrar una API que cumple la misma función y podemos utilizar para exterminar.

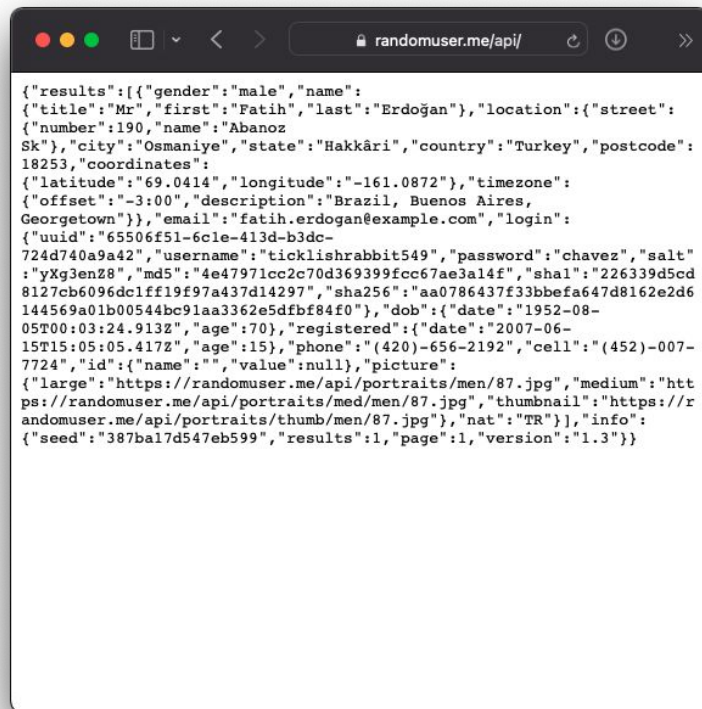
Entraremos a <https://randomuser.me/api> con el navegador.

- Entrar a la página equivale a hacer un pedido (**request**)
- La dirección (<https://randomuser.me/api>) recibe el nombre de endpoint
- Al entrar el navegador automáticamente procesará la respuesta y nos mostrará el resultado.



Observando la respuesta desde el navegador

- Al realizar el request a <https://randomuser.me/api> con el navegador veremos algo similar a la imagen, recordemos que los datos son generados al azar.
- El navegador nos muestra la respuesta, la cual viene en un formato llamado **JSON** (JavaScript Object Notation)
- Todas las APIs con las que trabajaremos devuelven los resultados en este formato pero existen otros.



- Podemos copiar y pegar los resultados obtenidos en <https://jsonformatter.curiousconcept.com/> al revisarlos ordenados nos daremos cuenta que este formato JSON nos es familiar, se compone de objetos y arreglos, esta información la podemos obtener utilizando lo que hemos aprendido hasta ahora.
- Antes de trabajar con la respuesta aprendamos a obtenerla directamente con JavaScript

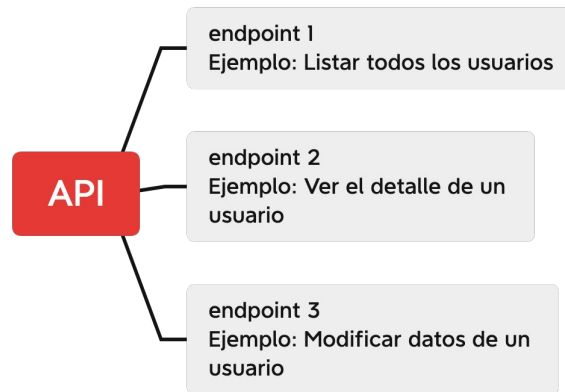
```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "Mr",
        "first": "Fatih",
        "last": "Erdoğan"
      },
      "location": {
        "street": {
          "number": 190,
          "name": "Abanoz Sk"
        },
        "city": "Osmaniye",
        "state": "Hakkâri",
        "country": "Turkey",
        "postcode": 18253,
        "coordinates": {
          "latitude": "69.0414",
          "longitude": "-161.0872"
        }
      },
    },
  ],
}
```

...

APIs

Endpoints

- Dentro de una API puede haber uno o más endpoints, así como dentro de un sitio web pueden haber múltiples páginas web. Cada endpoint provee un servicio distinto. Cada endpoint tiene una URL distinta.
- Un endpoint se compone de una acción + una URL
- Lo acción y URL de cada endpoint junto a lo que devuelve se especifica en la documentación de cada API.



/* Obteniendo los datos desde Javascript */

APIs

Haciendo un request con fetch

Para conectarnos a la API de randomUser desde JavaScript crearemos una página web nueva y dentro de script agregaremos lo siguiente.

```
async function getRandomUser(){  
  const res = await fetch("https://randomuser.me/api")  
  const data = await res.json()  
  console.log(data);  
}  
  
getRandomUser()
```

Viendo la respuesta en la consola del navegador

Si ejecutamos el código anterior podremos observar como se muestra la información por consola:



```
>> ▶ async function getRandomUser(){
    const res = await fetch("https://randomuser.me/api")
    const data = await res.json()
    console.log(data);
}...

← ▶ Promise { <state>: "pending" }

▼ Object { results: (1) [...], info: {...} }
  ▶ info: Object { seed: "791d20dd5b4b0de0", results: 1, page: 1, ... }
  ▼ results: Array [ {...} ]
    ▶ 0: Object { gender: "female", email: "elena.blanchard@example.com", phone: "02-83-25-28-48", ... }
      length: 1
    ▶ <prototype>: Array []
    ▶ <prototype>: Object { ... }
```


APIs

Revisando el código

```
async function getRandomUser(){  
  const res = await fetch("https://randomuser.me/api")  
  const data = await res.json()  
  console.log(data);  
}  
  
getRandomUser()
```

1. Por ahora ignoremos todo lo que diga *async* y *await*
2. *Fetch* hace un request al endpoint
3. *res.json* Transforma los resultados para que podamos leerlos fácilmente como un objeto de Javascript
4. Si quisiéramos consultar otra API solo tendríamos que cambiar el endpoint (y el nombre de la función para ser coherentes)

APIs

Revisando el código

```
async function getRandomUser(){  
  const res = await fetch("https://randomuser.me/api")  
  const data = await res.json()  
  console.log(data);  
}
```

getRandomUser()

Se obtiene el contenido en formato JSON de la respuesta

Se obtiene la respuesta(res) de la consulta


Ejercicio

Crea una página web nueva y dentro del tag script haz un request a uno de los endpoints mostrados en la siguiente página <https://api.gael.cloud/#publica> u otro endpoint público de tu interés y lee la respuesta en la consola del inspector de elementos.

Ejercicio ¡Manos al teclado!



`/* Consultar una API REST y mostrar los resultados en el DOM
*/`

`/* Realizar un request a una API con JS mostrando por consola
la respuesta */` 

`/* Modificar el DOM con los resultados obtenidos a partir de
un pedido */`

`/* Manejar errores con try and catch en caso de problemas
con la API */`

Objetivos

APIs

Async y Await

- JavaScript no espera a nadie. Fetch es lento, requiere de conectarse con una página web por lo que si no esperamos la respuesta no obtendremos resultado alguno.
- Prueba quitando await de fetch y recarga la página, obtendrás un error.

```
async function getRandomUser(){  
  const res = fetch("https://randomuser.me/api")  
  const data = await res.json()  
  console.log(data);  
}  
  
getRandomUser()
```

APIs

Async y Await

Lo que realmente está pasando es que `fetch` y `.json()` del resultado de `fetch` devuelven un tipo de dato especial llamado **promesa**, `await` espera que una promesa se resuelva. Una regla importante es que `await` solo puede ser utilizada dentro de una función declarada `async`.

```
async function getRandomUser(){
  const res = await fetch("https://randomuser.me/api")
  const data = await res.json()
  console.log(data);
}

getRandomUser()
```

APIs

Modificando el DOM con datos de una API

```
async function getRandomUser(){
  const res = await
fetch("https://randomuser.me/api")
  const data = await res.json()
  console.log(data)
  const element =
document.querySelector(".user")
  element.innerHTML = data.results[0]['email']
}

getRandomUser()
```

Una vez que tengamos la respuesta solo tenemos que modificar el DOM, la parte que puede llegar a ser un poco más difícil es moverse a través de todos los campos que tiene la respuesta. En este ejemplo es un objeto que dentro tiene la clave resultados con un arreglo, en el primer valor del arreglo está el usuario y para este ejemplo agregamos el email del usuario al DOM

Ejercicio

Adicional al email muestra el teléfono (phone), nombre completo, la ciudad y la imagen

Ejercicio ¡Manos al teclado!



/* Consultar una API REST y mostrar los resultados en el DOM
*/

/* Realizar un request a una API con JS mostrando por consola
la respuesta */ ✓

/* Modificar el DOM con los resultados obtenidos a partir de
un pedido */ ✓

/* Manejar errores con try and catch en caso de problemas
con la API */

Objetivos

/* Sentencia Try Catch */

APIs

Try and Catch

- Si algo falla dentro de un bloque try se ejecutará el bloque catch

```
try{  
  error()  
}  
catch{  
  console.log("hola")  
}
```

- De esta forma podemos lograr que si el método fetch no recibe una respuesta porque la API está caída mostrar un mensaje de aviso al usuario o llamar a otra API.

APIs

Sentencia Try Catch

```
async function getSomething() {  
  try {  
    const res = await fetch("https://estapaginanooexiste.cl");  
    const data = await res.json();  
    console.log(data);  
  } catch (e) {  
    alert(e.message);  
  }  
}  
  
getSomething();
```

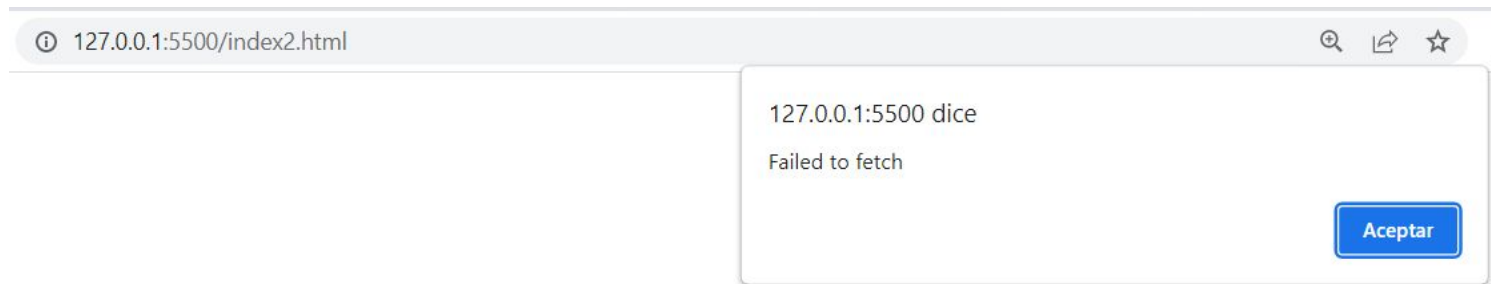
Las APIs son como páginas web y pueden estar caída o con otro tipo de problemas, para esto try y catch son ideales.

Para probarlo y no esperar que el servicio falle probaremos con una página que no existe.

APIs

Sentencia Try Catch

- Si ejecutamos el código en un navegador, veremos como aparece una ventana emergente con el mensaje de error correspondiente a lo que haya sucedido dentro del bloque `try`.



Ejercicio


Envolver en un bloque try y un bloque catch nuestra consulta a la API de randomuser.me


Ejercicio ¡Manos al teclado!




/* Consultar una API REST y mostrar los resultados en el DOM



/* Realizar un request a una API con JS mostrando por consola la respuesta */ 

/* Modificar el DOM con los resultados obtenidos a partir de un pedido */ 

/* Manejar errores con try and catch en caso de problemas con la API */ 

Objetivos



Cierre

{desafío}
latam_



¿Existe algún concepto que no
hayas comprendido?

Reflexionemos

- Revisar la guía que trabajarán de forma autónoma.
- Revisar en conjunto el desafío.

¿Qué sigue?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam