Guía de ejercicios - APIs (II)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

¡Manos a la obra! - Eventos con jQuery

- Cambia el color de fondo del div utilizando el evento de ratón: mouseenter

```
<div class="miCaja" style="width:100px;height:100px;border: 1px solid
black"></div>
```

- Muestra la posición del elemento (item 1, item 2, entre otros) antes del primer elemento <ul>, además, añade un color de fondo a los impares y otro color de fondo a los pares.

```
<ul>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
</ul>
```

- Muestra el texto que se encuentra en el párrafo con la etiqueta <p>, cuando el usuario haga un click sobre el botón denominado “Mostrar”.

```
<button>Mostrar</button>
<p>Desafío Latam</p>
```

Actividad guiada 1: Utilizando AJAX

Realizar la conexión a siguiente API que cuenta con datos de usuario, como id, email, nombre, apellido y una imagen. En base al siguiente HTML, una vez que el usuario haga click en el botón, se obtendrá la información de la API y será mostrada en el documento web mediante una función de AJAX:

```
<h4>Información de la API</h4>
<div>
   <button type="button">Mostrar Información</button>
   <div class="resultado"></div>
</div>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, incorporar el enlace o CDN de jQuery, que nos permita incluir esta librería en nuestro ejemplo, agregar el extracto del código HTML entregado en el enunciado y enlaza el archivo externo denominado "script.js", como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <h4>Información de la API</h4>
   <div>
      <button type="button">Mostrar Información</button>
      <div class="resultado"></div>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3*:** Ahora en el archivo script.js, se debe agregar la función de AJAX para realizar una petición del tipo GET.

```
$.ajax({
   type:"GET",
   url:"aquí va la url a consultar",
   dataType:"json",
   success: function(data) {
     //si todo sale bien, se agrega la funcionalidad aquí.
   },
   error: function(error) {
     //si todo sale bien, se agrega la funcionalidad aquí.
   },
```

```
});
```

- ***Paso 4*:** Ya dispuesta la estructura de AJAX para una petición del tipo GET, se debe agregar la función principal de jQuery, además el escucha y manejador de evento para el botón, porque la idea es que cuando el usuario haga un click sobre el botón, se realice el llamado a la API, se traiga la información y se muestre.

```
$(document).ready(function(){
   $('button').on('click',function(){
      $.ajax({
         type:"GET",
         url:"https://reqres.in/api/users",
         dataType:"json",
         success: function(data) {
           //si todo sale bien, se agrega la funcionalidad aquí.
         },
         error: function(error) {
            //si todo sale bien, se agrega la funcionalidad aquí.
         },
      });
   });
});
```

- ***Paso 5*:** Con la funcionalidad lista de AJAX más el evento "click" activo en el botón. Agrega dentro de la función "success" de AJAX, lo que debe ocurrir cuando la información que retorne la API esté completa y sea correcta la conexión. Mientras que en la función "error", se desplegará la información para indicar que hubo un error con la conexión a la API. En este caso, vamos a mostrar la información primeramente en un console.log, para ver que trae la API (es recomendable utilizar postman para ver la estructura de la información). En este ejercicio, la información la estamos recibiendo en una variable denominada "datosApi" y dentro de ella se extrae la data original que viene con la API para esa petición en específico.

```
$(document).ready(function(){
   $('button').on('click',function(){
      $.ajax({
         type:"GET",
         url:"https://reqres.in/api/users",
         dataType:"json",
         success: function(datosApi) {
            console.log(datosApi.data);
         }
```

```
        error: function(error) {
            //si todo sale bien, se agrega la funcionalidad aquí.
        },
      });
   });
});
```

- ***Paso 6*:** Al ejecutar el código anterior y si la conexión fue exitosa, en la consola del navegador el resultado debe ser:

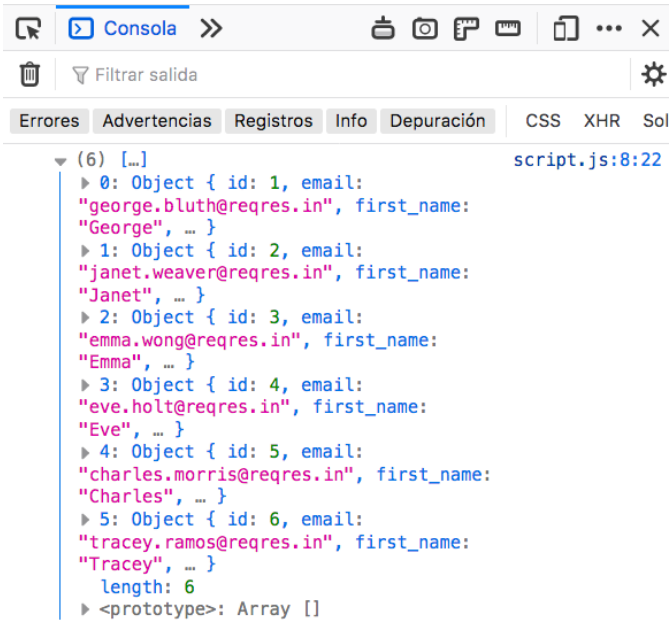

Imagen 1. Información proveniente de la API.
Fuente: Desafío Latam

- ***Paso 7*:** Ahora como ya tenemos la información proveniente de la API lista para usar, vamos a mostrar esa información dentro de la etiqueta con la clase "resultado". Por lo tanto, se debe captar el elemento por la clase "resultado" para luego mediante el método "append" de jQuery podremos agregar una simple etiqueta de párrafo para la data que retorna la API. Esta información viene encapsulada dentro de un arreglo, y dentro del arreglo son distintos objetos que contienen el contenido, entonces se hace necesario el uso de forEach para recorrer el arreglo.

```
$(document).ready(function(){
   $('button').on('click',function(){
      $.ajax({
         type:"GET",
         url:"https://reqres.in/api/users",
         dataType:"json",
```

```
        success: function(datosApi) {
           console.log(datosApi.data);
           datosApi.data.forEach(element => {
              $('.resultado')
                 .append(`<p>${element.id}-${element.email}-
                       ${element.first_name}-${element.last_name}-
                       ${element.avatar}</p>`);
            })
        },
        error: function(error) {
            //si algo sale mal, se agrega la funcionalidad aquí.
        },
     });
  });
});
```

- ***Paso 8*:** Al ejecutar el código anterior en nuestro navegador web y hacer clic sobre el botón "Mostrar Información", el resultado debería ser:

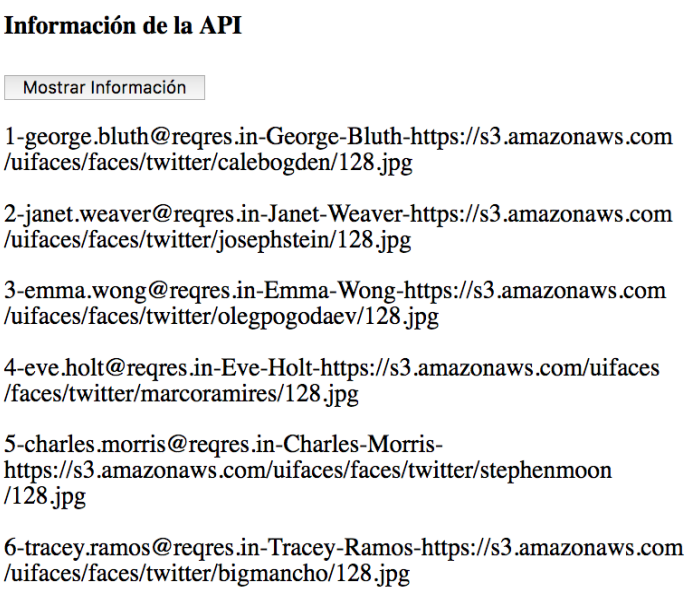

Imagen 2. Mostrando información proveniente de la API en el documento web.
Fuente: Desafío Latam

¡Manos a la obra! - AJAX

- Utiliza Postman para realizar una conexión, traer y mostrar la información de la siguiente API: https://reqres.in/api/users.

¡Manos a la obra! - Utilizando AJAX

- Para la siguiente dirección web (perteneciente a una API): https://jsonplaceholder.typicode.com/users. Realiza la respectiva conexión mediante AJAX y luego muestra la información solo cuando un usuario haga click sobre un botón.

Soluciones

Eventos con jQuery

1. Cambia el color de fondo del div utilizando el evento de ratón: mouseenter

```
$(document).ready(function() {
     let miCaja = $('.miCaja');
     miCaja.on('mouseenter',function(){
          $(this).css('background-color','green');
     });
});
```

2. Muestra la posición del elemento (item 1, item 2, entre otros) antes del primer elemento <ul>, además, añade un color de fondo a los impares y otro color de fondo a los pares.

```
$(document).ready(function() {
      $('ul li').each(function(index){
            $('ul').before("<p>"+index+": "+$(this).text()+"</p>");
      });
      $('ul li:odd').css('background-color','green');
      $('ul li:even').css('background-color','blue');
});
```

3. Muestra el texto que se encuentra en el párrafo con la etiqueta <p>, cuando el usuario haga un click sobre el botón denominado “Mostrar”.

```
$(document).ready(function() {
      $('button').on('click',function(){
          $('p').show();
      });
});
```

Ajax

1. Utiliza Postman para realizar una conexión, traer y mostrar la información de la siguiente API: https://reqres.in/api/users.

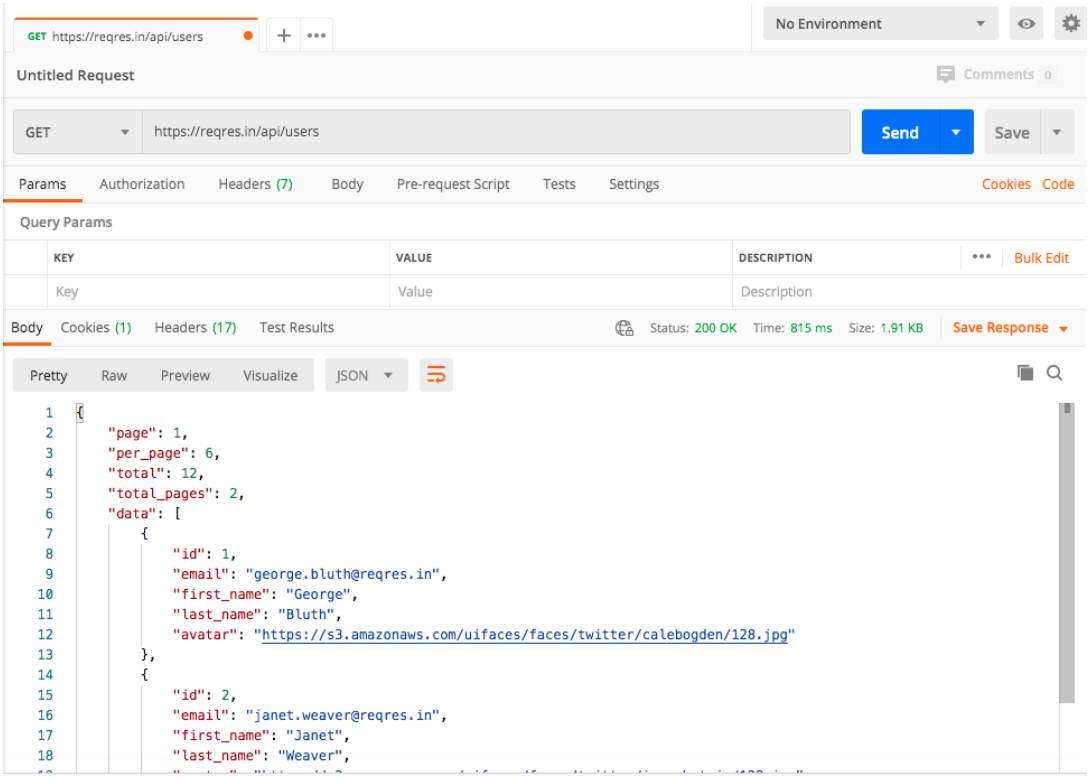

Imagen 3. Solución ejercicio propuesto
Fuente: Desafío Latam

Utilizando Ajax

1. Para la siguiente dirección web (perteneciente a una API): https://jsonplaceholder.typicode.com/users. Realiza la respectiva conexión mediante AJAX y luego muestra la información solo cuando un usuario haga clic sobre un botón.

```
$(document).ready(function(){
      $('button').on('click',function(){
      $.ajax({
            type:"GET",
            url:"https://jsonplaceholder.typicode.com/users",
            dataType:"json",
            success: function(datosApi) {
            console.log(datosApi);
            datosApi.forEach(element => {
                  $('.resultado').append(`<p>${element.id}-
${element.name}-${element.username}-${element.email}-
${element.phone}</p>`);
            })
            },
            error: function(error) {
            console.log(error);
            },
      });
      });
 });
```