



APIs

Ajax

{desafío}
latam_



***Crear plugins propios con
jQuery para ampliar las
funcionalidades de una
aplicación web.***

- Unidad 1:
Introducción al lenguaje JavaScript
- Unidad 2:
Funciones y Ciclos
- Unidad 3:
Arrays y Objetos
- Unidad 4:
APIs



Te encuentras aquí



¿Qué aprenderás en esta sesión?

- *Realiza un request a una API utilizando AJAX y procesar el resultado agregando información al DOM.*

**/* Primer acercamiento,
consultas a una API */**

¿Qué es AJAX?

AJAX (Asynchronous JavaScript and XML) se refiere a un grupo de tecnologías que se utilizan para desarrollar aplicaciones web. Al combinar estas tecnologías, las páginas web son más receptivas, puesto que los paquetes pequeños de datos se intercambian con el servidor y las páginas web no se vuelven a cargar cada vez que un usuario realiza un cambio de entrada. AJAX, permite que un usuario de la aplicación web interactúe con una página web sin la interrupción que implica volver a cargar la página web. La interacción del sitio web ocurre rápidamente sólo con partes de la página de recarga y renovación.



¿Qué es AJAX?

AJAX se compone de las siguientes tecnologías para crear un nuevo enfoque al desarrollo de aplicaciones web:

- XHTML y CSS para presentar información.
- DOM (Document Object Model - modelo de objetos de documento) para visualizar e interactuar de forma dinámica la información presentada.
- El objeto XMLHttpRequest para manipular los datos de forma asíncrona con el servidor web.
- XML, HTML y XSLT para el intercambio y la manipulación de datos.
- Se visualiza JavaScript para enlazar solicitudes e información de datos.



Introducción al Concepto de API

- Las APIs son un medio simplificado para conectar tu proyecto a través del desarrollo de aplicaciones, permiten compartir sus datos con clientes y otros usuarios externos.
- Un ejemplo conocido es la API de Google Maps o de servicios de Share de facebook, incluso integraciones con servicios más pequeños que nos permiten obtener valores como el clima, provincias, ciudades, estados de un país o información de carácter global.
- Las APIs públicas son esenciales para el desarrollo de código abierto.



Transferencia de Estado Representacional (REST)

- REST, es un estilo arquitectónico diseñado para proporcionar estándares entre los sistemas informáticos en la web, lo que facilita la comunicación entre éstos.
- Un servidor que opera bajo el paradigma REST, no importa qué tecnología utiliza, solo importa definir de forma correcta los puntos de consulta (endpoints) que quedarán expuestos o abiertos, para que otro sistema a través de una url pueda hacer una petición HTTP para obtener información.

Métodos HTTP

Principales métodos HTTP

- **GET:** Este método se utiliza para obtener un recurso específico o una colección de datos.
- **POST:** Este método se usa para crear recursos nuevos.
- **PUT:** Este método se usa para actualizar un recurso.
- **DELETE:** Este método se usa para eliminar recursos.

Estos conceptos se basan en la operación básica de orientación a objetos llamada CRUD. Te recomendamos investigar sobre estos términos a modo de complemento con esta lectura.



Métodos HTTP

Cabeceras (headers)

- Permiten entregar al servidor información adicional sobre cómo se espera recibir o enviar cierta información.

Por ejemplo:

```
GET /articles/23  
Accept: text/html, application/json
```

- Genera una acción de tipo get y en su cabecera se determina que dicha petición acepta como respuesta texto en formato de HTML o bien un objeto JSON.
- La cantidad de cabeceras o condiciones a la petición son extensas, recomendamos leer la [documentación oficial de Mozilla](#) que indica la gama de posibilidades y condiciones que se pueden exponer en una cabecera.

Métodos HTTP

Rutas (PATH)

La ruta es uno de los elementos más importantes a la hora de hacer una petición HTTP. Es la dirección física en donde se indica que consultar y a dónde hacerlo.

Entendamos el concepto con un ejemplo: <https://pokeapi.co/api/v2/pokemon/pikachu>

- Un protocolo HTTP por el cual se hace la consulta.
- Una ruta base (BASE_URL) en este caso sería pokeapi.co/api/v2/. Esta es la ruta base, la raíz a donde las peticiones irán en todo momento.
- Una ruta relativa (RELATIVE PATH) pokemon/pikachu que indica la acción, busca a un pokémon en específico, en este caso, pikachu.

Métodos HTTP

Códigos de respuestas (Response Codes)

Cuando uno realiza una petición HTTP, además de traer información en su metadata, podemos observar diversos parámetros, uno de los más importantes son los códigos de respuestas a la petición realizada. Dichos códigos nos permiten entender si la consulta fue realizada con éxito, si hay un problema con la consulta realizada por parte del cliente o bien, el servidor presenta un problema al momento de realizar la consulta.

Algunos códigos de respuestas conocidos:

- 200 (OK)
- 201 (CREATED)
- 204 (NO CONTENT)
- 400 (BAD REQUEST)
- 403 (FORBIDDEN)
- 404 (NOT FOUND)
- 500 (INTERNAL SERVER ERROR)

{desafío}
latam_



Invitamos a consultar la [documentación oficial de Mozilla](#) sobre los contextos de dichas respuestas y cómo utilizar esta información a beneficio de las personas que desarrollan aplicaciones.

Notación de Objetos de JavaScript (JSON)

- JSON es el acrónimo para JavaScript Object Notation y aunque su nombre lo diga, no es necesariamente parte de JavaScript, de hecho es un estándar basado en texto plano para el intercambio de información, por lo que se usa en muchos sistemas que requieren mostrar o enviar información.
- JSON puede representar cuatro tipos primitivos (cadenas, números, booleanos, valores nulos) y dos tipos estructurados (**objetos y arreglos**).



Notación de Objetos de JavaScript (JSON)

Estructura de arreglo de objetos

Estructura básica de un JSON.

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  }
]
```

- Nótese que el archivo inicia []. Esto hace referencia a que es un arreglo.
- Seguidamente, tenemos {} que hace referencia a un objeto.
- Este objeto tiene diversas propiedades y valores.
- Como vimos en la unidad de arreglos y objetos, es posible aplicar métodos para acceder a estos datos según el tipo de estructura.

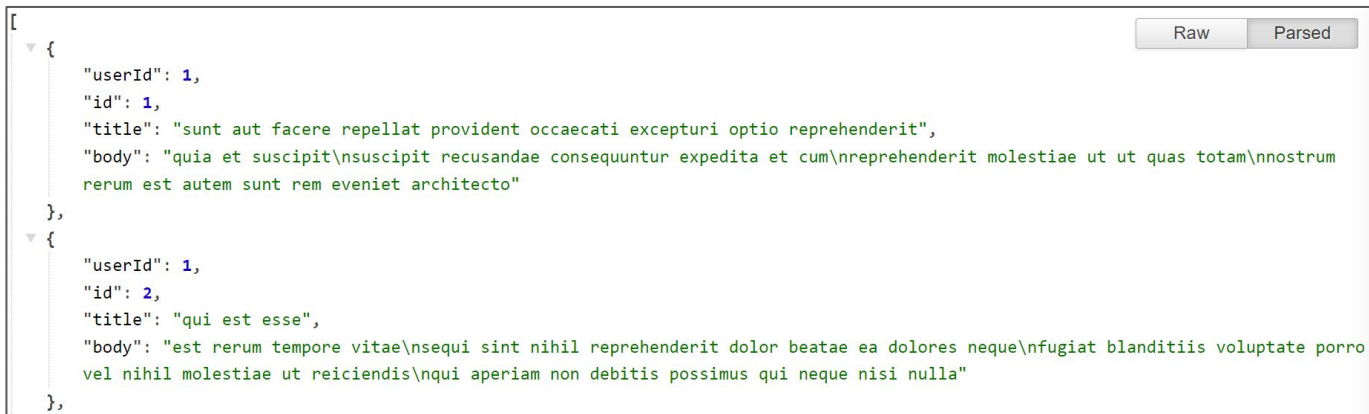
Ejercicio guiado: “Obteniendo datos de la API Json Place Holder”



Obteniendo datos de la API Json Place Holder

Utilizando Ajax

- **Paso 1:** Accedemos al siguiente [enlace](#) y veremos el siguiente resultado.



The screenshot shows a JSON viewer interface with two tabs: 'Raw' and 'Parsed'. The 'Parsed' tab is selected, displaying a JSON array of two user objects. The first object has 'id' 1 and a title 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit'. The second object has 'id' 2 and a title 'qui est esse'.

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
  }
],
```


Obteniendo datos de la API Json Place Holder

- **Paso 2:** Creamos un documento HTML para mostrar los posts.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <ul id="posts"></ul>
    <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
    <script src="ejemplojQuery.js"></script>
  </body>
</html>
```



Obteniendo datos de la API Json Place Holder

- **Paso 2:** Hacemos la consulta y comprobamos el tipo de dato y estructura recibida.

```
$(function () {  
    // Realizamos la consulta a la API  
    $.ajax({  
        url: "https://jsonplaceholder.typicode.com/posts",  
        type: "GET",  
        dataType: "json",  
        success: function (posts) {  
            console.log(posts);  
        },  
    });  
});
```

Obteniendo datos de la API Json Place Holder

- **Paso 2:** Hacemos la consulta y comprobamos el tipo de dato y estructura recibida.

[illegible]

Nótese que recibimos la información como un arreglo de objetos. Entonces podemos aplicar la lógica para recorrer los elementos de dicho arreglo.

Obteniendo datos de la API Json Place Holder

- **Paso 3:** Hacemos el recorrido del arreglo y lo mostramos como elemento de tipo lista en el HTML.

```
$(function () {  
    // Realizamos la consulta a la API  
    $.ajax({  
        url: "https://jsonplaceholder.typicode.com/posts",  
        type: "GET",  
        dataType: "json",  
        success: function (posts) {  
            // Iteramos sobre los posts  
            posts.forEach((post) => {  
                // Generamos un elemento li con los datos del post  
                const li = $("- ").text(post.title);  
  
                // Agregamos el elemento li a la lista  
                $("#posts").append(li);  
            });  
        },  
    });  
});

```

Obteniendo datos de la API Json Place Holder

Revisemos el `forEach`

posts es un arreglo

Creamos una variable para cada post

Utilizamos la sintaxis de punto para acceder a la clave `title` de cada objeto `post` y así obtenemos su valor.

Guardamos en una constante la generación de un ``

```
posts.forEach((post) => {  
  // Generamos un elemento li con los datos del post  
  const li = $("- ").text(post.title);  
  // Agregamos el elemento li a la lista  
  $("#posts").append(li);  
});

```

Accedemos al elemento HTML con el id `posts` y le asignamos el valor de la constante `li`.

Obteniendo datos de la API Json Place Holder

El objeto post

- **post** es un objeto que posee un conjunto de claves y valores.
- Al ser un objeto accedemos a cada clave y valor mediante la sintaxis de punto.

```
ejemplojQuery.js:12
{userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati excepturi optio reprehenderi
▼ t', body: 'quia et suscipit\nsuscipit recusandae consequuntur ...strum rerum est autem sunt rem evenie
t architecto'} i
  body: "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae
  id: 1
  title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
  userId: 1
  ► [[Prototype]]: Object
```

Ejercicio guiado - Obtener datos financieros



Ejercicio guiado

Obtener datos financieros

Consultar la siguiente API que nos muestra la información del dólar, UF, UTM y otros datos financieros en formato JSON. Si la consultas a través del navegador, verás su estructura; a través de POSTMAN podremos trabajar con ella, para obtener más adelante sólo aquellos datos que nos aporten a la información que queremos mostrar.

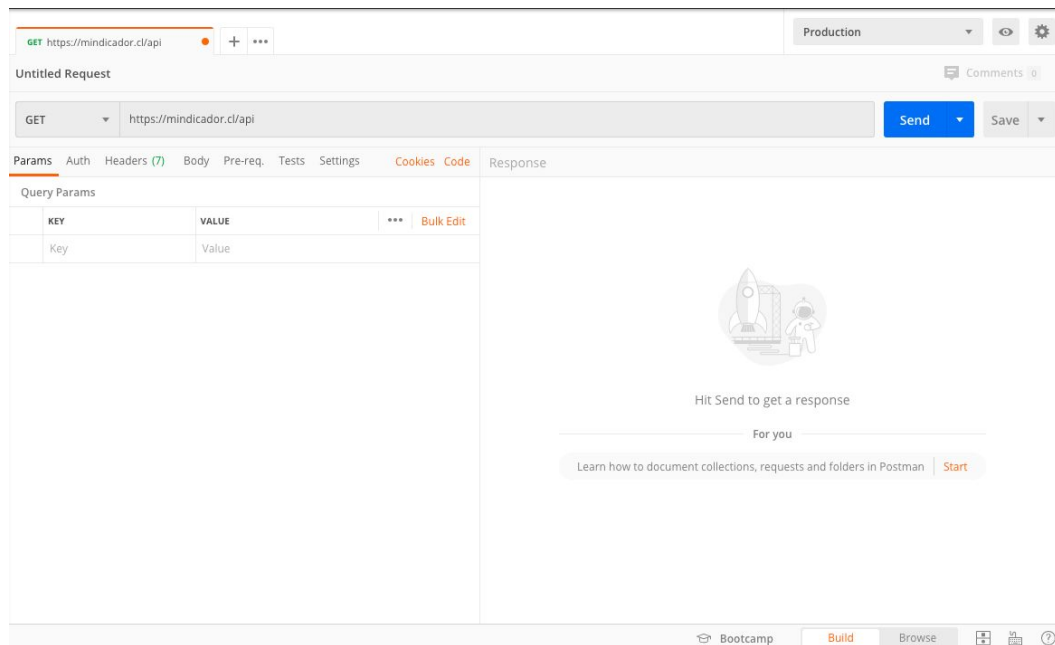
Por ahora consultar la URL y validar que nos responda con un código 200 (OK):



Ejercicio guiado

Obtener datos financieros

- **Paso 1:** Para esto copiaremos el siguiente endpoint <https://mindicador.cl/api> en Postman.



Ejercicio guiado

Obtener datos financieros

- **Paso 2:** Ahora si apretamos el botón de "Send", ejecutaremos el request y obtendremos el resultado de la API

The screenshot shows a REST client interface with the following details:

- Request:** GET `https://mindicador.cl/api`
- Status:** 200 OK, Time: 858 ms, Size: 2.35 KB
- Response Body (JSON):**

```
1 {
2   "version": "1.6.0",
3   "autor": "mindicador.cl",
4   "fecha": "2020-04-22T01:00:00.000Z",
5   "uf": {
6     "codigo": "uf",
7     "nombre": "Unidad de fomento (UF)",
8     "unidad_medida": "Pesos",
9     "fecha": "2020-04-21T04:00:00.000Z",
10    "valor": 28664.96
11  },
12  "ivp": {
13    "codigo": "ivp",
14    "nombre": "Indice de valor promedio (IVP)",
15    "unidad_medida": "Pesos",
16    "fecha": "2020-04-21T04:00:00.000Z",
17    "valor": 29751.84
18  },
19  "dolar": {
20    "codigo": "dolar",
21    "nombre": "Dólar observado",
22    "unidad_medida": "Dólar"
```

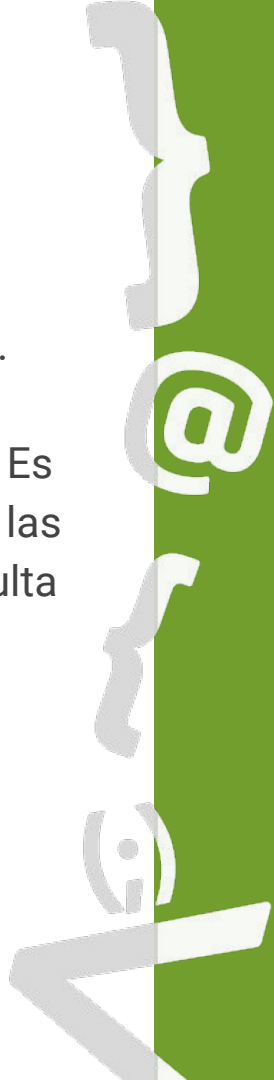


Ejercicio guiado

Obtener datos financieros

Si todo está correcto, deberías ver la estructura de la respuesta y código http 200.

Para conocer más APIs que pronto estarás utilizando, visita: [Public APIs](#). Es importante que sepas que no todas las APIs son públicas, también existen las privadas que requieren de una API-KEY declarada en los parámetros de la consulta para la devolución de datos o bien la ejecución de una función interna.



Resumen de la sesión

- En resumen, las APIs le permiten habilitar el acceso a sus recursos y, al mismo tiempo, mantener la seguridad y el control, además de cómo habilitar el acceso y a quienes dependen de ti.
- La seguridad de las APIs tiene que ver con una buena gestión de ellas. Para conectarse a las APIs y crear aplicaciones que utilicen los datos o las funciones que estas ofrecen, se puede utilizar una plataforma de integración distribuida que conecte todos los elementos, incluidos los sistemas heredados y el Internet de las cosas (IoT).



Próxima sesión...

- *Crea un script que realice peticiones asíncronas utilizando la librería jQuery y AJAX para resolver un problema planteado.*