



Coding Bootcamp

Sprint 1



Temario



Temario

- AJAX.
- JSON.
- AJAX y jQuery.
- AJAX y SEO.
- API.



AJAX

¿Qué es AJAX?



Se dice "Asíncrono" porque no se sabe cuándo se obtendrá una respuesta y no se frena al programa mientras se espera.

- AJAX = Asynchronous JavaScript and XML.
- Es una **técnica** que permite **interactuar** (enviar y recibir información) **con un servidor sin necesidad de recargar la página**.
- Su surgimiento a mediados de la década del 2000 significó un cambio enorme para JavaScript y el desarrollo de aplicaciones web. Fue una revolución.
- Permitió la creación de aplicaciones como Gmail, Google Maps, Facebook, Twitter, etc. ⇨ **Single Page Applications** (SPA) que se parecen a aplicaciones de escritorio.



¿Qué se puede recibir del servidor vía AJAX?

Cualquier cosa, pero en general:

- Texto plano (texto sin un formato específico).
- HTML.
- **JSON** (que sustituyó al XML; igual se le sigue llamando AJAX).



JSON



¿Qué es JSON?

Como puede ser CSV o XML.

- JSON = JavaScript Object Notation.
- Es un **formato de texto** muy simple para **intercambiar información**.
- Es ligero y fácil de leer por humanos.
- Deriva de JavaScript, pero es independiente del lenguaje de programación.
- La información se guarda en pares de clave-valor.

```
{  
  "nombre"      : "María",  
  "apellido"    : "Rodríguez",  
  "edad"        : 36,  
  "nacionalidad" : ["Uruguay", "España"]  
}
```

A diferencia de un objeto de JavaScript, las propiedades deben ir en comillas.



Ejemplos de JSON

Prueben ingresar a las siguientes URLs:

- Open Movie Database: <https://private.omdbapi.com/?apikey=bef9c583&t=cinema+paradiso>
- Star Wars: <https://swapi.dev/api/people/1/?format=json>
- Google Books: <https://www.googleapis.com/books/v1/volumes?q=0596554877>


Estos son servicios que **en lugar de retornar un código HTML retornan código JSON**.

Si quieren hacer pruebas con datos ficticios en formato JSON, entren aquí:

<http://jsonplaceholder.typicode.com>. Sirve probar y prototipar.



Sugerencia: Instalar esta extensión de Chrome.



JSON Formatter

offered by callumlocke.co.uk

★★★★★ (1168) | [Developer Tools](#) | 596,760 users

ADDED TO CHROME

OVERVIEW | REVIEWS | SUPPORT | RELATED

Compatible with your device

Makes JSON easy to read. Open source.

FEATURES

- JSON & JSONP support
- Syntax highlighting
- Collapsible trees, with indent guides
- Clickable URLs
- Toggle between raw and parsed JSON
- Works on any valid JSON page – URL doesn't matter
- Works on local files too (if you enable this in chrome://extensions)
- You can inspect the JSON by typing "json" in the console

[Website](#)
[Report Abuse](#)

Additional Information

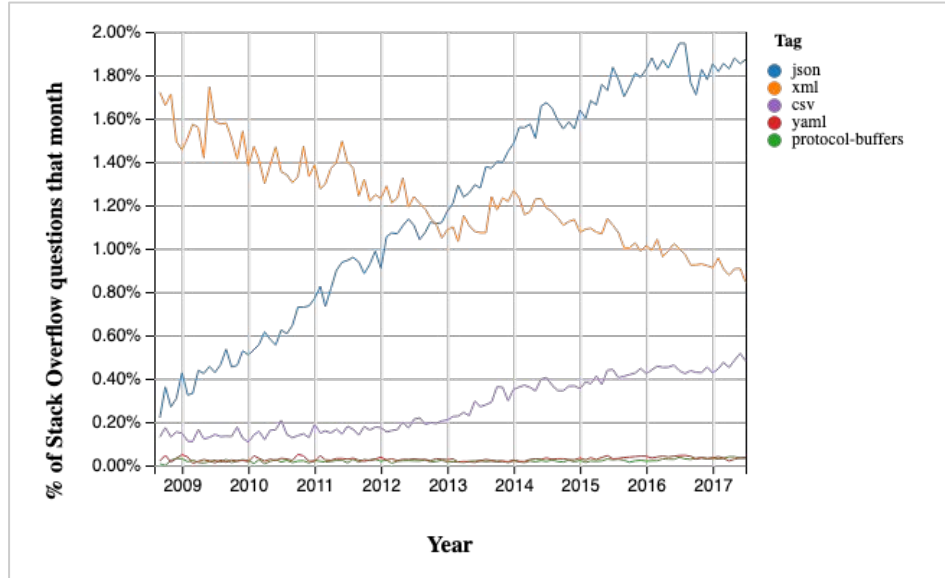
Version: 0.6.0
Updated: October 5, 2014
Size: 27.62KIB
Language: English

https://graph.facebook.com/cocacola

```
{
  "id": "40796308305",
  "name": "Coca-Cola",
  "picture": "http://profile.ak.fbcdn.net/hprofile-ak-ash2/174560_407963083",
  "link": "http://www.facebook.com/coca-cola",
  "likes": 46963810,
  "cover": {
    "cover_id": "10151829640053306",
    "source": "http://a8.sphotos.ak.fbcdn.net/hphotos-ak-ash3/s720x720/529413_10151829640053306_446360541_n.jpg",
    "offset_y": 0
  },
  "category": "Food/beverages",
  "is_published": true,
  "website": "http://www.coca-cola.com",
  "username": "coca-cola",
  "founded": "1886",
  "description": "Created in 1886 in Atlanta, Georgia, by Dr. John S. Pembe"
```



JSON es cada vez más el formato por defecto a la hora de compartir información entre sistemas web.



Fuente: <https://twobithistory.org/2017/09/21/the-rise-and-rise-of-json.html>



AJAX y jQuery

AJAX y jQuery (1/4)

⚠ Notar que también es posible hacer llamadas AJAX usando JavaScript "puro" usando [Fetch](#), aunque no funciona en ninguna versión de Internet Explorer.



jQuery permite realizar llamadas AJAX de forma muy simple utilizando una función llamada `ajax`. Ver documentación [aquí](#).

```
$.ajax(...);
```

Recordar que la variable llamada `$` es el objeto jQuery. En este caso se está accediendo al método `ajax` de dicho objeto.

En las siguientes slides se muestran ejemplos del parámetro que debe recibir la función `ajax`.



AJAX y jQuery (2/4) – Ejemplo

```
$.ajax({  
  url: "https://swapi.dev/api/people/1/?format=json",  
  success: function(datosObtenidos) {  
    console.log("Nombre del personaje: " + datosObtenidos.name);  
  }  
});
```

El método [ajax](#) recibe como parámetro un objeto, el cual por lo menos debe contener dos propiedades: `url` y `success`.

- `url` es un string que indica la URL a donde se realizará el *request*.
- `success` es una función que se llama en caso de que el *request* sea exitoso.

AJAX y jQuery (3/4) – Ejemplo



```
$.ajax({  
    method      : "GET",  
    url         : "https://swapi.dev/api/people/1/?format=json",  
    dataType    : "json",  
    success     : function(datosObtenidos) {  
        console.log("Nombre del personaje: " + datosObtenidos.name);  
    },  
    error       : function() {  
        alert("Ocurrió un error durante la llamada");  
    }  
});
```

Este es el mismo ejemplo, pero un poco más completo.

AJAX y jQuery (4/4) – Developer Tools



Probar el código anterior en la Consola (en una página que tenga jQuery instalado) y observar la pestaña “Network” en las Developer Tools de Chrome.

The screenshot shows the Chrome Developer Tools interface with the Network tab selected. The top bar includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. Below the tabs, there are icons for a red dot, a crossed-out circle, a video camera, and a funnel. The 'View' section shows a list icon, a tree icon, and checkboxes for 'Preserve log', 'Disable cache', and 'Offline'. The 'No throttling' option is also visible. A filter input field is present, followed by checkboxes for 'Regex' and 'Hide data URLs'. The 'All' filter is selected, and a list of filter categories is shown: XHR, JS, CSS, Img, Media, Font, Doc, WS, Manifest, and Other. The main table displays network requests with the following columns: Name, Status, Type, Initiator, Size, Time, and Waterfall. A single request is listed: '?format=json' with a status of 200, type of xhr, initiator of 'jquery.js:9664', size of 486B, and time of 346ms. The Waterfall column shows a green bar representing the request duration. At the bottom, a summary bar indicates '1 requests | 486B transferred'.

Name	Status	Type	Initiator	Size	Time	Waterfall
<input type="checkbox"/> ?format=json	200	xhr	jquery.js:9664	486B	346ms	

1 requests | 486B transferred



Un par de notas...



Loaders y mensajes de error

Cuando se hace un pedido al servidor (*request*) es posible que el **servidor demore** en contestar (y no se puede saber cuánto demorará). Incluso podría ocurrir que el **servidor nunca conteste** porque está caído. Por este motivo, es necesario tomar medidas que generen una mejor experiencia de usuario.


Por ejemplo:


- Usar “loaders”.
 - Ejemplo 1: <http://loading.io>
 - Ejemplo 2: <https://getbootstrap.com/docs/4.5/components/progress/>
 - Ejemplo 3: <https://fontawesome.com/how-to-use/on-the-web/styling/animating-icons>
- Mostrar mensajes de error si la respuesta no es la deseada.
- Dar la opción de reintentar el request.






AJAX y SEO (1/2)

 Existen sitios web donde el **Search Engine Optimization** (SEO) es fundamental. Por ejemplo: un sitio institucional, un blog, MercadoLibre, PedidosYa o cualquier sitio de e-commerce. Son sitios web cuyo contenido se quiere que aparezca en los primeros lugares de los resultados de búsqueda de Google (u otro *search engine*).

 En estos casos, hay que tener mucho cuidado si nuestro sitio web consiste en un único HTML, el cual levanta todo el contenido a través de llamadas **AJAX**. A los “ojos” de Google, nuestro sitio tendría sólo una página *indexable*: `index.html`.

 Esto no siempre es un problema. En el caso de un sitio web privado, de uso interno en una empresa, el SEO no es relevante.



AJAX y SEO (2/2)

Un sitio web para el cual el SEO es importante, intentará tener la mayor cantidad de páginas *indexadas* en los motores de búsqueda.

Por ejemplo, **PedidosYa** se preocupó de que cada restaurante tenga su propia página, con su propia URL y que dicho contenido pueda ser *indexado* por Google:

- <https://www.pedidosya.com.uy/cadenas/la-pasiva>
- <https://www.pedidosya.com.uy/cadenas/chiviteria-marcos>
- <https://www.pedidosya.com.uy/restaurantes/montevideo/pastas-blanes-menu>

De esta forma, Google *indexa* cada una de estas páginas por separado, generando más contenido en el buscador, y aumentando las probabilidades de que alguien encuentre a PedidosYa en Google.

🤔 ¿Qué hacemos si queremos tener una **Single Page Application** y además nos preocupa el SEO? Afortunadamente se puede tener lo mejor de los dos mundos, pero requiere [trabajo extra](#).



API



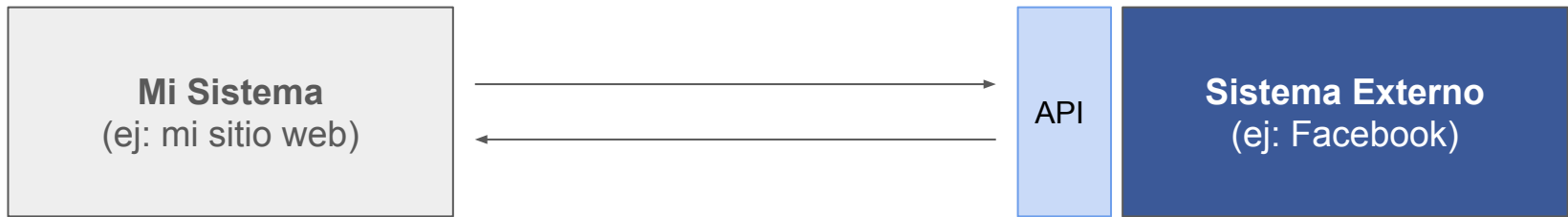
¿Qué es una API? (1/4)

- API = Application Programming Interface.
- Es una **interface** que le permite a 2 **sistemas independientes** comunicarse entre sí.
 - Uno de los sistemas **“provee”** la API (es dueño de la misma).
 - El otro **“consume”** la API.
- Quien provee la API debe especificar qué servicios se proveen y cómo se usan. La buena práctica es que toda API está acompañada de una **documentación**.
- Notar que la definición anterior no habla de Internet y de hecho el concepto de API va más allá de sistemas conectados a una red.



¿Qué es una API? (2/4)

Diagrama de ejemplo:



En este ejemplo, el sistema externo provee una API y nuestro sistema la consume.

El sistema externo debe especificar qué servicios se proveen y cómo se usan. En este ejemplo:

<https://developers.facebook.com/docs/graph-api>.



¿Qué es una API? (3/4)

- Es común que la información intercambiada con una API tenga formato JSON.
- Las APIs pueden ser:
 - **Públicas:** cualquiera puede acceder a ellas. Ej: La API de Google Maps, Star Wars.
 - **Privadas:** sólo determinados actores pueden acceder a las mismas. Ej: una empresa crea un API para sus clientes o crea una API para sus sucursales.
 - **De acceso restringido:** para poder acceder a ellas es necesario autenticarse (es lo que ocurre en la mayoría de los casos). Ej: Open Exchange Rates (<https://openexchangerates.org/>).
 - **Gratis:** se pueden usar gratuitamente sin restricción de uso. Ej: Facebook (<https://developers.facebook.com/docs/graph-api>).
 - **Parcialmente gratis (freemium):** hasta cierto uso se pueden usar gratis, luego es necesario pagar. Ej: Monkey Learn (<http://www.monkeylearn.com/>). Esta es una startup uruguaya!
 - **Pagas:** es necesario pagar para usarlas. Ej: Twilio (<https://www.twilio.com/sms>).

¿Qué es una API? (4/4)



Links interesantes:

- IFTTT: <https://ifttt.com>.
- Zapier: <https://zapier.com>.
- RapidAPI: <https://rapidapi.com>.

Y recuerden, casi siempre hay una API para todo...

Lectura interesante:

<https://medium.com/routific/don-t-build-it-in-house-there-is-an-api-for-that-c929b8677137>

The screenshot shows the 'ORAL-B DEVELOPER PROGRAM' page. It features a header with the program name, a sub-header 'CONNECT TO THE WORLD'S FIRST BLUETOOTH CONNECTED TOOTHBRUSH', and a brief description: 'Write apps that work with iOS and Android and connect directly with the Oral-B Bluetooth Connected Toothbrush'. Below this, there's a section titled 'SDK' which explains that the SDK provides access to core features like device state, pressure, brushing time, current mode, and cached session data. An illustration shows a smartphone connected to an Oral-B toothbrush via Bluetooth. A 'DOWNLOAD SDK' button is visible on the right.