

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 14. JAVASCRIPT

AJAX CON JQUERY



OBJETIVOS DE LA CLASE

- Aprender el concepto de AJAX, y sus usos.
- Realizar llamadas simples mediante AJAX.
- Entender el concepto de API.

GLOSARIO:

Clase 13

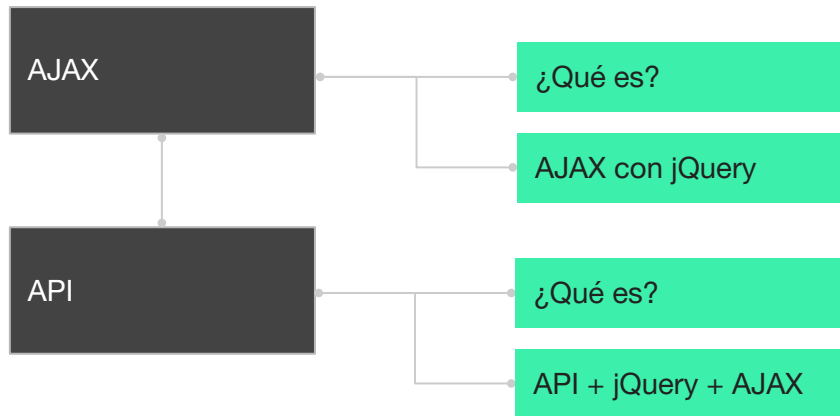
Animaciones con jQuery

- **Función show():** muestra un elemento del DOM. Suponiendo que se encuentre oculto mediante css (display:none), al seleccionarlo con JQuery y ejecutar show(), el elemento aparecerá en la pantalla.
- **Función hide():** es la inversa a show. Suponiendo que se encuentre visible mediante css, al seleccionarlo con JQuery y ejecutar hide(), el elemento desaparecerá de la pantalla.
- **Función fadeIn():** tiene el mismo fin que la función show, solo que en este caso, el elemento aparece con una transición.
- **Función fadeOut():** tiene el mismo fin que la función hide, solo que en este caso, el elemento desaparece con una transición.
- **Función slideDown():** hace aparecer al elemento haciendo una transición hacia abajo. Es una alternativa al **fadeIn()**.
- **Función slideUp():** hace desaparecer al elemento haciendo una transición hacia arriba. Es una alternativa al **fadeOut()**.
- **Función slideToggle():** hace aparecer o desaparecer al elemento mediante slide.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 14

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 13



Efectos y animaciones con jQuery



EJEMPLOS EN VIVO



ANIMACIONES EN JQUERY

Clase 14



AJAX con jQuery



AJAX EN TU PROYECTO



TERCERA ENTREGA DEL PROYECTO FINAL

Clase 15



Introducción a SPA con jQuery



EJEMPLO EN VIVO



PRACTICAR SPA



HERRAMIENTAS DE LA CLASE

Les compartimos algunos recursos para acompañar la clase

- Guión de clase N° 14 [aquí](#).
- Quizz de clase N° 14 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

AJAX



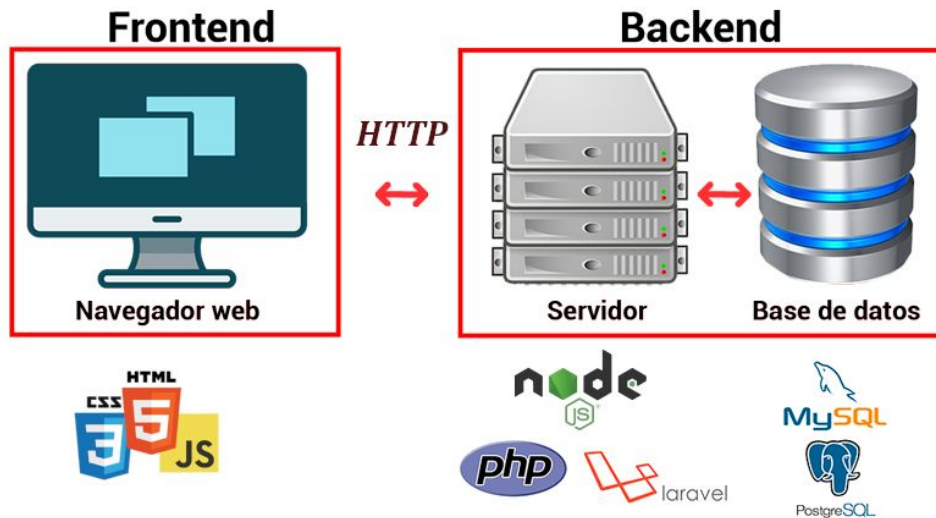
¿QUÉ ES AJAX?

AJAX significa JavaScript asincrónico y XML (Asynchronous JavaScript and XML). Es un conjunto de técnicas de desarrollo que permiten que las aplicaciones web funcionen de forma asincrónica, pudiendo procesar tareas en segundo plano.

Un proceso asíncrono tiene la característica de no generar tiempo de espera por la respuesta. Como resultado, cualquier app o web que use AJAX puede enviar y recibir datos del servidor sin la necesidad de volver a cargar toda la página.

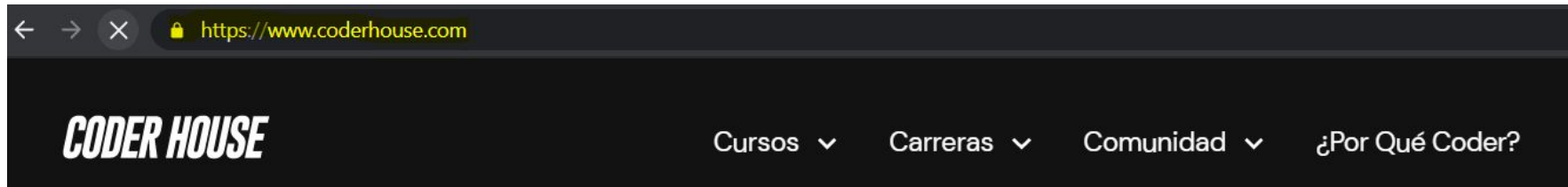
COMUNICACIÓN CON EL SERVIDOR

Antes de ver cómo utilizar AJAX, debemos entender cómo se envían los datos al servidor mediante AJAX. Se utiliza el protocolo HTTP para comunicarnos con el servidor. Con HTTP empleamos métodos (llamados métodos de petición) para acceder o enviar recursos e información al servidor.



MÉTODO GET

El método GET sirve para solicitar una representación de un recurso específico del servidor. Las peticiones que usan el método GET son para acceder a datos, páginas o imágenes*/ , entre otros. Cada vez que utilizamos el navegador para acceder a una dirección web utilizamos el método GET.



MÉTODO POST

El método POST sirve para enviar datos al servidor. Es posible determinar que información queremos enviar en una cabecera identificada como Content-Type. Cada vez enviamos un formulario de contacto utilizamos el método POST.

Suscribite a nuestro Newsletter

miEmail@gmail.com



COMPARACIÓN GET Y POST

Aspecto	Con GET	Con POST
Los datos son visibles en la url	Sí	No
Los datos pueden permanecer en el historial del navegador	Sí	No
Una url puede ser guardada conteniendo parámetros de un envío de datos	Sí	No
Existen restricciones en la longitud de los datos enviados	Sí (no se puede superar la longitud máxima de una url)	No
Se considera preferible para envío de datos sensibles (datos privados como contraseñas, números de tarjeta bancaria, etc.)	No (los datos además de ser visibles pueden quedar almacenados en logs)	Sí (sin que esto signifique que por usar post haya seguridad asegurada)
Codificación en formularios	application/x-www-form-urlencoded	application/x-www-form-urlencoded ó multipart/form-data. Se usa multipart para envío de datos binarios, por ejemplo ficheros.
Restricciones de tipos de datos	Sí (sólo admite caracteres ASC-II)	No (admite tanto texto como datos binarios p.ej. archivos)
Se considera preferible para disparar acciones	No (podría ser accedido por un robot que dispararía la acción)	Sí (sin que esto garantice que no pueda acceder un robot)
Riesgo de cacheado de datos recuperados en los navegadores	Sí	No
Posibles ataques e intentos de hackeo	Sí (con más facilidad)	Sí (con menos facilidad)

AJAX CON JQUERY

Para emplear los métodos AJAX con jQuery se utiliza un conjunto de métodos que generalmente reciben dos parámetros:

- Una URL (absoluta o relativa) del archivo que va a procesar la llamada.
- Un array de parámetros (puede no ser necesario en caso de usar GET).

Estos parámetros son procesados por servidor para realizar operaciones, y luego retornar cierta información al cliente en modo de respuesta.

Todo esto puede ocurrir mientras en la App o web el usuario sigue en la misma pantalla, es decir, que no se recarga la ventana.

FORMATO DE ENVÍO DATOS Y RESPUESTA

¿En qué formato se recibe o envía información al servidor?

Se recibe mediante texto plano. Una forma de estandarizar este proceso y recibir información más compleja es utilizar JSON. Como ya vimos, es un formato basado en texto plano para representar datos estructurados en la sintaxis de objetos de JavaScript. De esa forma podemos recibir texto plano, texto en HTML, arrays, etc.

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
},
]
```

MÉTODOS AJAX CON JQUERY

LLAMADA AJAX: GET

Para ejecutar AJAX en JS mediante JQuery, se usa el método GET para solicitar un dato puntual. Puede ser, por ejemplo, solicitar datos a la dirección

<https://jsonplaceholder.typicode.com/posts>

```
//Declaramos la url que vamos a usar para el GET
const URLGET =
"https://jsonplaceholder.typicode.com/posts"
//Agregamos un botón con jQuery
$("#body").prepend('<button id="btn1">GET</button>');
//Escuchamos el evento click del botón agregado
$("#btn1").click(() => {
    $.get(URLGET, function (respuesta, estado) {
        if(estado === "success"){
            let misDatos = respuesta;
            for (const dato of misDatos) {
                $("#body").prepend(`<div>
                                <h3>${dato.title}</h3>
                                <p> ${dato.body}</p>
                                </div>`);
            }
        }
    });
});
```

```
//Declaramos la url que vamos a usar para el GET
const URLGET =
"https://jsonplaceholder.typicode.com/posts"
//Declaramos la información a enviar
const infoPost = { nombre: "Ana", profesion:
"Programadora" }
//Agregamos un botón con jQuery
$("body").prepend('<button id="btn1">POST</button>');
//Escuchamos el evento click del botón agregado
$("#btn1").click(() => {
    $.post(URLGET, infoPost , (respuesta, estado) => {
        if(estado === "success") {
            $("body").prepend(`<div>
                                Guardado:${respuesta.nombre}
                                </div>`);
        }
    });
});
```

LLAMADA AJAX: POST

Para ejecutar AJAX en JS mediante JQuery, se usa el método POST para enviar datos al servidor. Puede ser por ejemplo, enviar datos a la dirección

<https://jsonplaceholder.typicode.com/posts>

EJEMPLO CON ARCHIVO JSON ESTÁTICO

```
//Declaramos la url del archivo JSON local
const URLJSON = "data/datos.json"
//Agregamos un botón con jQuery
$("body").prepend('<button id="btn1">JSON</button>');
//Escuchamos el evento click del botón agregado
$("#btn1").click(() => {
$.getJSON(URLJSON, function (respuesta, estado) {
    if(estado === "success"){
        let misDatos = respuesta;
        for (const dato of misDatos) {
            $("body").prepend(`<div>
                                <h3>${dato.name}</h3>
                                <p> ${dato.email}</p>
                                </div>`)
        }
    }
});
```

ALGUNOS USOS DE AJAX CON JS

- Agregar o modificar productos a un carrito de compras.
- Enviar un formulario sin refrescar la página.
- Agregar o quitar productos de una wishlist.
- Enviar un comentario en un blog.
- Notificaciones en Facebook, Twitter.
- Ventanas de chats.

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

***¡Ya estás llegando al fin de la cursada!
Recordá que luego de la corrección de tu proyecto
final, se notificará por Slack y email si quedaste en el
TOP10***

***No cuelgues, que tenés hasta 2 semanas desde que te
notificamos para solicitar los beneficios.***

¡5/10 MINUTOS Y VOLVEMOS!

CODER HOUSE

API

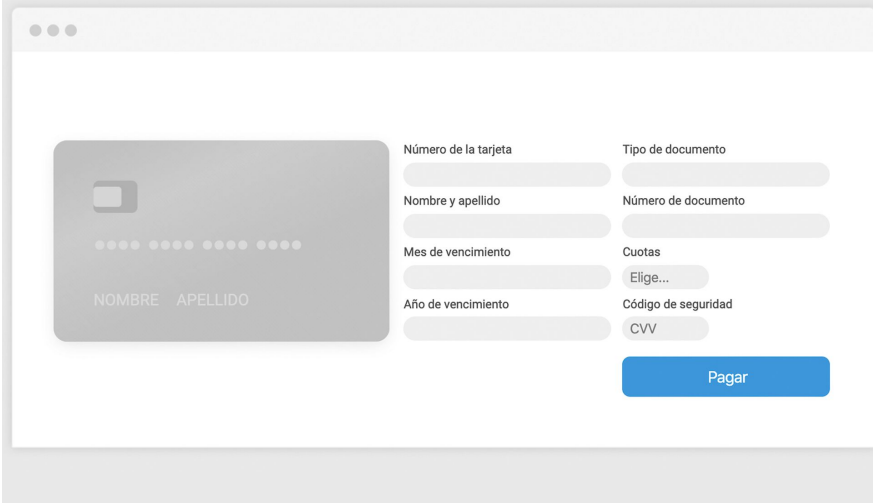
¿QUÉ ES UNA API?

API o Application Programming Interfaces, es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas. El uso de una API es el mecanismo más útil para conectar dos softwares entre sí. De esta manera, podemos garantizar el intercambio de mensajes o datos en formato estándar.

Nos conectamos al API para obtener o enviar información.

EJEMPLO DE API

Por ejemplo: una web de e-commerce que permite pagar con MercadoPago, está integrando la API de MP, mediante la cual una web X puede cobrar una compra. Esa web usará las funciones y reglas de la API de MP para integrarla a sus necesidades del e-commerce.

A mockup of a payment form for MercadoPago. On the left is a gray card placeholder with a chip icon, a masked number '0000 0000 0000 0000', and the labels 'NOMBRE' and 'APELLIDO'. To the right of the card are several input fields: 'Número de la tarjeta', 'Nombre y apellido', 'Mes de vencimiento', 'Año de vencimiento', 'Tipo de documento', 'Número de documento', 'Cuotas' with a dropdown 'Elige...', 'Código de seguridad', and 'CVV'. A blue 'Pagar' button is at the bottom right.

API + JQUERY + AJAX

Una de las formas de comunicarse con una API (siempre y cuando esta lo contemple) es utilizar AJAX en jQuery.

En la práctica sería muy similar a realizar una llamada AJAX común, solo que tendremos que cumplir los requisitos que la API defina para cada llamada.

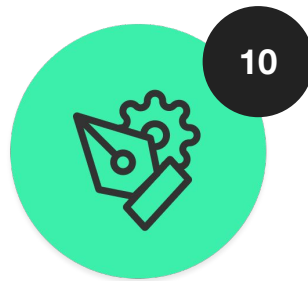
Por ejemplo: si fuera una API de una plataforma de pagos, podremos usar llamadas para enviar pagos, para consultar cuotas, etc.

EJEMPLO APLICADO: POST AL API

```
$( document ).ready(function() {  
    //Declaramos la url del API  
    const APIURL = 'https://jsonplaceholder.typicode.com/posts ' ;  
    //Declaramos la información a enviar  
    const infoPost = { nombre: "Ana", profesion: "Programadora" }  
    //Agregamos un botón con jQuery  
    $("body").prepend('<button id="btn1">ENVIAR API</button> ');  
    //Escuchamos el evento click del botón agregado  
    $("#btn1").click(() => {  
        $.ajax({  
            method: "POST",  
            url: APIURL,  
            data: infoPost,  
            success: function(respuesta) {  
  
                $("body").prepend(`<div>${respuesta.nombre}</div>`);  
  
            }  
        });  
    });  
});  
});
```

¡IMPORTANTE!

- Tanto para el uso de jQuery y APIs siempre vas a necesitar el frontend (en este caso JS), y un backend, ya que estas interactuando entre un cliente y un servidor.
- Una API no es un lenguaje, es una metodología de trabajo para programar un conjunto de reglas bajo una necesidad funcional.
- Una API puede estar programada en diferentes lenguajes: php, asp, etc; y puede llamarse mediante otros: JS, HTTP Request, PHP, etc.



AJAX EN TU PROYECTO

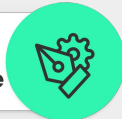
Tomando como ejemplo la llamada AJAX con JSON, incorpora al proyecto del simulador al menos una llamada AJAX.

AJAX EN TU PROYECTO

Formato: Página HTML y código fuente en JavaScript. Debe identificar el apellido del alumno/a en el nombre de archivo comprimido por “claseApellido”

Sugerencia: Usamos GET para obtener información del servidor y POST para enviar. getJSON es la forma simplificada de obtener la información de un archivo JSON.

Desafío
entregable



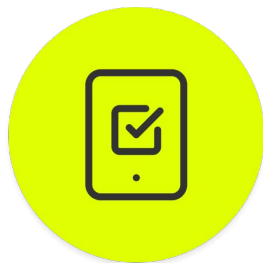
>> Consigna: Tomando como ejemplo la llamada AJAX con JSON, incorpora al proyecto del simulador al menos una llamada AJAX, que mediante la interacción del usuario (un botón o envío de formulario), se realice la llamada y la misma responda un JSON estático. incluir esa respuesta en el HTML.

>>Aspectos a incluir en el entregable:

Archivo HTML y Archivo JS, referenciado en el HTML por etiqueta `<script src="js/miarchivo.js"></script>`, que incluya la definición de un algoritmo en JavaScript con métodos jQuery con al menos una petición ajax

>>Ejemplo:

- Al confirmar el envío del formulario, sin refrescar la página, mostrar alerta de "envío exitoso"
- Consultar usuarios desde [JSONPlaceholder](#) y mostrarlos en el HTML.
- Consultar personajes desde la [API de Harry Potter](#) y mostrarlos en el HTML



TERCERA ENTREGA DEL PROYECTO FINAL

Deberás **incorporar jQuery para controlar elementos, sumar efectos y animaciones, así como optimizar diseño HTML y CSS,** en función de la tercera entrega de tu proyecto final.

TERCERA ENTREGA DEL PROYECTO FINAL

Formato: Página HTML y código fuente en JavaScript. Debe identificar el apellido del alumno/a en el nombre de archivo comprimido por “claseApellido”.

Sugerencia: En la tercera entrega buscamos que puedan definir el alcance del simulador y sus interacciones esenciales, refinando algunos aspectos trabajados en las entregas anteriores, empleando las herramientas ofrecidas por la librería jQuery.

Proyecto
Final



2

>>Objetivos Generales:

1. Refinar la codificación del simulador y su interacción con el usuario.
2. Construir estructura y estilo final de la aplicación.

>>Objetivos Específicos:

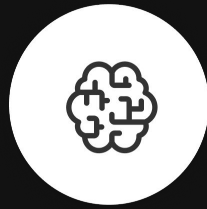
1. Definir estructura completa del simulador e interfaces con HTML.
2. Declarar reglas de estilo y/o utilizar un framework CSS para estilar la aplicación.
3. Emplear jQuery para definir modificación del DOM, efectos y animaciones.

>>Se debe entregar:

- Maquetación del HTML y eventos a manejar en él.
- Incorporación de jQuery para acceder y agregar elementos al DOM.
- Implementación de efectos y animaciones.
- Estilos base CSS del simulador

¿PREGUNTAS?





¡PARA PENSAR!

¿Te gustaría comprobar tus conocimientos de la clase?

Te compartimos a través del chat de zoom
el enlace a un breve quiz de tarea.

Para el profesor:

- *Acceder a la carpeta "Quizzes" de la camada*
 - *Ingresar al formulario de la clase*
 - *Pulsar el botón "Invitar"*
 - *Copiar el enlace*
- *Compartir el enlace a los alumnos a través del chat*



RECURSOS:

Material
ampliado



- Asincronismo |

Los apuntes de Majo (Página 32)

- AJAX |

¿Qué es AJAX?

- Herramienta REST API |

REST API JSONPLACEHOLDER

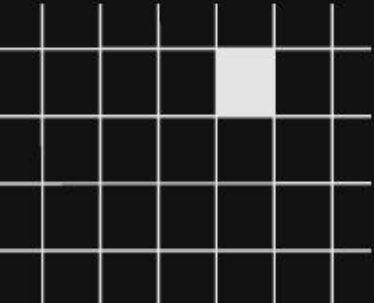
- Plugin Visual Code

Live Server



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Concepto de AJAX.
 - Ejemplo de llamadas AJAX.
 - Concepto de APIs.
- 



OPINA Y VALORA ESTA CLASE