# El evento **hover** sobre las imágenes, está causando un efecto de desplazar los demás componentes de forma abrupta. ¿Qué tendríamos que hacer para hacer una animación más suave al usuario?

Para que no sea de forma abrupta tenemos que utilizar una transición en CSS.

En el caso de mis imágenes podemos poner:

transition: all 0.5s ease;

All: Cualquier propiedad que se quiera cambiar.

0.5s: Duración entre el estado original y como va a acabar.

Ease: La velocidad en la que lo hará.

También se puede poner:

transition-property: transform;

transition-duration: 0.5s;

transition-timing-function: ease;

Más información:  
[CSS Transitions (w3schools.com)](https://www.w3schools.com/css/css3_transitions.asp)

# Suponiendo que tenemos un backend. ¿Qué tendrías que hacer para que la búsqueda mostrara los resultados a través de un API rest sin refrescar la pantalla?

En estos casos se utiliza una tecnología llamada AJAX de JavaScript.

Crearía un formulario que cuando le den al botón enviar llame a una función en JavaScript que estuviera escuchando ese botón.

La función de JavaScript recuperaría el valor del input y haría una llamada al API REST conAJAX utilizando el método GET (ya el POST es para guardar datos y el DELETE para borrarlos)

En el callback de la llamada AJAX se procesa la respuesta y se mostraríamos los resultados en la página con el .innerHTML

Un ejemplo que podemos encontrar sería:

<script>

function loadDoc() {

const xhttp = new XMLHttpRequest();

xhttp.onload = function() {

document.getElementById("demo").innerHTML = this.responseText;

}

xhttp.open("GET", "demo\_get2.asp?fname=Henry&lname=Ford");

xhttp.send();

}

</script>

Como se puede ver si crea una instancia del objeto XMLHttpRequest (AJAX) que cuando carga obtiene el elemento con id “demo” y le pone la respuesta.

Debajo se puede ver la petición GET con AJAX y como la envía.

En cuanto a las funciones usadas para que la búsqueda muestre los resultados a través de una API, el XMLHTTPRequest(); crea un objeto predefinido de Javascript. Un objeto tiene propiedades y métodos, este objeto nos permite comunicar con el servidor para obtener información sobre él.

Xhttp.open() es un método que tiene dos parametros obligatorios(método,URL). Este método Realiza una petición de apertura de comunicación con un método que puede ser principalmente GET o POST.

Con GET los parámetros de la petición van incluidos en la url. Con POST los parámetros de la petición van en las cabeceras de HTTP. url puede ser una ruta relativa o completa.

xmlhttp.send(); envia la petición al servidor

Fuente del ejemplo: [W3Schools Tryit Editor](https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_get2)

Más información: [AJAX Introduction (w3schools.com)](https://www.w3schools.com/js/js_ajax_intro.asp)

# 3. Caso tengamos que informar al usuario sobre las cookies. Cómo controlaríamos para saber si el usuario ha aceptado las cookies o no. y en caso de afirmativo, como podríamos volver a pregunta pasados 30 días?

Lo primero sería mostrarle un mensaje para que las acepte o no (modal). En el botón aceptar pondríamos una función de JavaScript.

Esta función establecerá una cookie con un nombre especifico por ejemplo: cookiesAceptadas (se aconseja el inglés) con una fecha de expiración de 30 días.

Por otra parte en todas las páginas de la web comprobaríamos si esa cookie existe si no existe volveríamos a mostrar el mensaje.

Se puede encontrar más información aquí:  
[JavaScript Cookies (w3schools.com)](https://www.w3schools.com/js/js_cookies.asp)

Un ejemplo que podemos encontrar en esta misma página para insertar la coockie al usuario sería:  
function setCookie(cname, cvalue, exdays) {

const d = new Date();

d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

let expires = "expires="+ d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

Y para obtenerla:

function getCookie(cname) {

let name = cname + "=";

let decodedCookie = decodeURIComponent(document.cookie);

let ca = decodedCookie.split(';');

for(let i = 0; i <ca.length; i++) {

let c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1)

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length); // SI EXISTE LA COOKIE

}

}

return ""; // SI NO EXISTE LA COOKIE

}

La función setcookie() **define una cookie que se enviará junto con el resto de las cabeceras HTTP** . Una cookie se utiliza a menudo para identificar a un usuario.

New date() Cuando es llamado como un constructor, retorna una cadena que representa la fecha y hora actual,

Gettime() Retorna el valor númerico de la fecha especificada como el número de milisegundos transcurridos desde el 1 de Enero de 1970, 00:00:00 UTC.

toUTCString();Convierte una fecha a una cadena usando la zona horaria UTC.

Con document.cookie puedes crear, leer y eliminar cookies.

**expires:** es un atributo opcional que especifica la fecha de caducidad de la cookie.´

Get cookie es una función que permite coger aquella cokkie que queremos recuperar.