JAVASCRIPT

CLASE 12
Material complementario

jQuery Eventos

CODER HOUSE

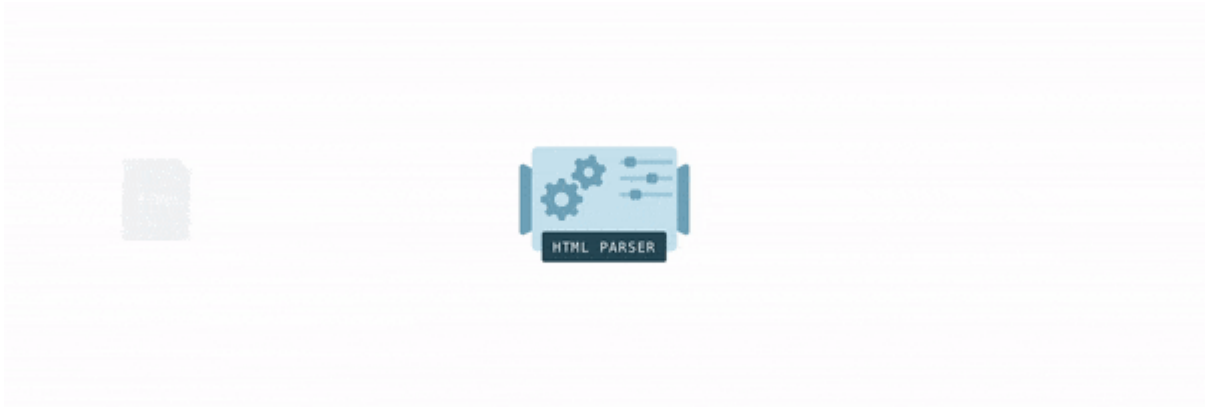
jQuery: eventos

Como analizamos en la clase 9, los eventos nos permiten detectar acciones del usuario sobre la aplicación, capturar entradas y efectuar salidas en respuesta. Definir los eventos a escuchar en la aplicación es responsabilidad del/la programador/a y, en ocasiones, definir el control del mismo evento sobre un conjunto de elementos identificados por clase o etiqueta puede implicar el recorrido de los nodos DOM utilizando un bucle.

Para simplificar la manera en la que definimos eventos con jQuery, podemos combinar los selectores con métodos de definición de eventos, los cuales pueden ser el método ***on()*** o los llamados métodos ***shortcut()***.

También aprendimos algunos eventos especiales como load, el cual podemos usar para detectar cuándo todos los elementos dentro de la pestaña del navegador fueron cargados.

Pero ahora nos concentraremos en una situación particular: dijimos que el DOM es generado automáticamente por el navegador, según la estructura de la página HTML actual, pero este proceso no es inmediato e implica una "traducción" del documento HTML a la estructura DOM, como podemos apreciar en la imagen siguiente:



El tiempo en el que el DOM se genera puede variar en función de la extensión del

documento HTML a interpretar, y más si optamos por incluir HTML dinámico. Para garantizar que los accesos al DOM se realicen una vez que este fue construido correctamente, existe un evento en jQuery conocido como ***ready*** que podemos emplear.

Método ready

El método ***ready()*** en jQuery puede ser empleado para detectar que el DOM está listo para usarse. Cuando una página se está cargando, existe un tiempo de espera hasta que podemos manipular todos los nodos del DOM. Veamos un ejemplo de codificación del método ***ready***.

```
$( document ).ready(function()
{
   console.log( "El DOM está listo" );
});
```

Este evento se emplea para asegurarnos que podemos acceder a los elementos HTML en el momento oportuno, luego de ser interpretados por el cliente. Si nuestro script incluye nuevos elementos al DOM mientras la página se carga, y es necesario acceder a los nodos recién creados, podemos usar ***ready*** para garantizar que las referencias a los nuevos nodos se realicen adecuadamente.

Existe un evento equivalente en JavaScript, con un comportamiento similar a ***ready,*** llamado DOMContentLoaded:

```
window.addEventListener('DOMContentLoaded', function () {
    console.log('El DOM está listo');
});
```

El método ***ready*** puede escribirse de tres formas distintas: una explícita, una forma acotada (forma shortcut) y una acotada con función flecha. Veamos un ejemplo de dicha notaciòn:

```
//Forma explicita
$( document ).ready(function() {
```

```
    console.log('El DOM está listo');
});
//Forma corta de ready()
$(function() {
    console.log('El DOM está listo');
});
//Forma corta con arrow function
$(() => {
    console.log('El DOM está listo');
});
```

Ready y load

En algunas aplicaciones, puede ser necesario determinar cuándo se cargan las imágenes y otros recursos externos de la página, para poder acceder a ellos y/o modificarlos adecuadamente. En esta situación, el evento load puede ser bastante útil:

```
$( document ).ready(function() {
    console.log('El DOM está listo');
});

window.addEventListener('load', function() {
    console.log( 'Todos los elementos de la ventana están cargados'
);
});
```

Pero debemos identificar qué ready se dispara antes que load, ya que la carga del DOM implica la interpretación de cada nodo del documento HTML actual, sin preocuparse por los tiempos de carga de los recursos asociados a la etiquetas (la carga de una imagen en una etiqueta <img> o un vídeo <iframe>).

Ventajas del método ready

- A diferencia del evento load, no espera a que se carguen todas las imágenes y

recursos externos en la ventana para ejecutarse, siendo entonces el evento recomendado para modificar el DOM durante la carga inicial de la aplicación.

- Si usamos HTML dinámico, podemos usar ready para la asociación de evento y acceso inicial sobre los nuevos elementos incluidos, evitando así errores de referencias, consecuencia de intentar acceder a un nuevo elemento agregado al DOM recientemente,
- Se pueden escribir múltiples métodos ready(), ejecutándose estos cuando el DOM está listo, en el orden que fueron definidos, pudiendo separar la respuesta al evento en más de un manejador de eventos.

Definiendo eventos con jQUERY

Existen distintas formas de asociar evento a elementos seleccionados con jQuery: una de ellas es a través del método *on*, y otra mediante los llamados métodos *shortcut*. A pesar de ser formas equivalentes, varían en su sintaxis, identificados a la segunda forma como de tipado más acotado.

Método ON

Con el método *on()* podemos asignar eventos a elementos del DOM, seleccionados previamente usando jQuery; es la opciòn de la librería al método addEventListener() de JS Vanilla. El primer parámetro comprende el identificador del evento, mientras que el segundo al manejador de eventos, el cual al igual que en JS puede ser una función tradicional, anónima, flecha con cuerpo, o flecha sin cuerpo. Veamos un ejemplo de uso de *on* para los eventos click y dobleclick:

```
//Agregamos un botón al body como primer elemento
$('body').prepend('<button id="btnjQuery">CLICK</button>');
//Asociamos el evento click al botón creado
$('#btnjQuery').on('click', function () {
```

```
    console.log("Respuesta a un click");
});
//Asociamos el evento doble click al botón creado
$('#btnjQuery').on('dblclick', () => {
    console.log("Respuesta al doble click");
});
```

Métodos shortcut

Estos métodos son atajos que podemos emplear desde cualquier selector para definir los eventos a escuchar. Son una equivalencia en notación al métodos on y en términos funcionales es lo mismo, pero ofrece al programador/a la posibilidad de emplear un sintaxis màs acotada.

Analicemos ahora los métodos shortcut màs empleados:

Shortcut click

El método click() es un atajo para .on("click", manejador). El evento click se dispara cuando el puntero del ratón está sobre el elemento, y el botón del ratón se presiona y se suelta. Cualquier elemento HTML puede emplear este evento:

```
//Agregamos dos botones con jQuery
$("body").prepend('<button id="btn1">BUTTON</button>');
$("body").prepend('<button id="btn2">BUTTON</button>');
//Asociamos el evento click para btn1
$("#btn1").click(function () {
    console.log(this);
});
//Evento click para btn2 con Arrow function y parámetro e
$("#btn2").click((e) => {
    console.log(e.target);
});
```

Shortcut change

El método change() es un atajo para .on("change", manejador). El evento change se dispara cuando el valor del elemento cambia. Este evento está limitado a los elementos <input>, <textarea> y <select>:

```
//Agregamos inputs con jQuery
$("body").prepend(`<input type="text"   class="inputsClass">
                   <input type="number" class="inputsClass">
                   <select class="inputsClass">
                        <option value="1" selected >ID 1</option>
                        <option value="2">ID 2</option>
                        <option value="3">ID 3</option>
                    </select>`);
//Asociamos el evento change a todos los inputs
$(".inputsClass").change(function (e) {
    console.log(e.target.value);
    console.log(this.value);
});
```

Una de las ventajas de esta notaciòn es la simplicidad que ofrece para definir la escucha del mismo evento sobre elementos que comparten una clase determinada.

Shortcut submit

El método submit() es un atajo para .on("submit", manejador). El evento submit se dispara cuando el usuario envía un formulario. Sólo está disponible para elementos <form>:

```
//Agregamos un formulario con jQuery
$("body").prepend(`<form id="myForm">
                       <input type="text"  >
                       <input type="number">
                       <input type="submit">
```

```
                    </form>`);
//Asociamos el evento submit al formulario
$("#myForm").submit(function (e) {
    //Prevenimos el comportamiento de submit
    e.preventDefault();
    //Obtenemos hijos del formulario
    let hijos = $(e.target).children();
    //Primer input type="text"
    console.log(hijos[0].value);
    //Primer input type="number"
    console.log(hijos[1].value);
});
```

Todo evento admitido por un elemento del DOM puede asociarse empleado la notaciòn shortcut de jQuery.

Trigger

Un trigger es una instrucción que permite ejecutar, de forma automática, cierto comportamiento en la aplicación, Se utiliza principalmente para realizar acciones en el programa, sin la intervención del usuario. Los triggers pueden ser útiles si usamos eventos en nuestra aplicación, sobre todo cuando necesitamos tener más de un evento asociado a un único evento; es decir que un evento de usuario podría disparar eventos de otros elementos.

En jQuery contamos con el método trigger() para disparar un evento específico, forzando la interpretación del manejador de eventos correspondiente en esa instrucción. Podemos usarlo de la siguiente manera:

```
//Agregamos un botón y un input
$("body").prepend('<button id="btn1">BUTTON</button>');
$("body").prepend('<input  id="ipt1" type="text">');
//Asociamos el evento change al ipt1
$("#ipt1").change((e) => {
    alert("El valor es " + e.target.value);
});
//Asociamos el evento click para btn1 y usamos trigger
$("#btn1").click(() => {
    //Usamos trigger para disparar el evento change de ipt1
    $("#ipt1").trigger("change");
});
```