

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





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 12. JAVASCRIPT

JQUERY: SELECTORES Y EVENTOS



OBJETIVOS DE LA CLASE

- Conocer el uso de selectores.
- Aprender el manejo de eventos en jQuery.
- Determinar cómo acotar la forma de asociar eventos con jQuery.

GLOSARIO:

Clase 11

Librería: en JS, una librería (o biblioteca) es un archivo de JavaScript que contiene muchas funciones, que realizan alguna tarea útil para tu aplicación o web.

jQuery: es una librería que sirve para manipular el DOM, controlar eventos, agregar animaciones y ejecutar llamadas AJAX, entre otras cosas. Si bien ya vimos el uso del DOM mediante las herramientas nativas de JS, lo que diferencia a jQuery es que es más práctico y potente, además de estar mucho más sistematizado y ordenado desde la forma en que está desarrollado

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 12

¡Para
recordar!



Método ready

Método on

Métodos shortcut

Click

Change

Submit

CRONOGRAMA DEL CURSO

Clase 11



jQuery y Selectores



EJEMPLO EN VIVO



JQUERY

Clase 12



jQuery: Eventos



EJEMPLOS EN VIVO



INCORPORAR JQUERY AL PROYECTO

Clase 13



Efectos y animaciones con jQuery



EJEMPLOS EN VIVO



ANIMACIONES EN JQUERY



HERRAMIENTAS DE LA CLASE

Les compartimos algunos recursos para acompañar la clase

- Guión de clase N° 12 [aquí](#).
- Quizz de clase N° 12 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

MÉTODO READY

jQuery y eventos

jQuery nos brinda un conjunto de métodos para asociar eventos a un selector. Son una

equivalencia al empleo de `addEventListener` y se prefieren si estamos usando la librería para acceder al DOM.

También tenemos una serie de métodos atajo (shortcut) para asociar los eventos de forma más sencilla.

CODER HOUSE



\$(DOCUMENT).READY()

El método ready() de jQuery se emplea para detectar que el DOM está listo para usarse.

Cuando una página se está cargando existe un tiempo de espera hasta que podemos manipular el DOM. Usamos este evento para asegurarnos que podemos acceder a los elementos HTML en el momento oportuno, luego de ser detectados por el cliente.

Es el equivalente al evento nativo de JavaScript [DOMContentLoaded](#)

DOM

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

JQUERY

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

FORMA CORTA DE READY()

La forma recomendada de declarar el método ready() es utilizando la asociación explícita a document. Pero existe una forma corta de definición y dado que el manejador de eventos generalmente es una función anónima, es posible usar funciones flechas.

```
//Forma explicita
$( document ).ready(function() {
    console.log('El DOM esta listo');
});

//Forma corta de ready()
$(function() {
    console.log('El DOM esta listo');
});

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

Ventajas de \$(document).ready ()

- A diferencia del evento **load**, no espera a que se carguen todas las imágenes y recursos externo en la ventana para ejecutarse.
- Permite detectar cuando el DOM está cargado en el browser para efectuar cambios inmediatos sobre la estructura HTML, sin errores.
- Se puede escribir múltiples funciones ready(), ejecutándose en el orden que sea definen cuando el DOM está listo. Pudiendo separar la respuesta al evento en más de un manejador de eventos.

READY VS. LOAD

Si es necesario determinar cuando se cargaron todas las imágenes y otros recursos externos de la página usamos el evento **load**. El evento **ready** ocurre antes que **load**.

```
$( document ).ready(function() {  
    console.log('El DOM esta listo');  
});  
  
window.addEventListener('load', function() {  
    console.log( 'Todos los elementos de la ventana están cargados ' );  
});
```

En cualquier llamado, puedes reemplazar el caracter \$ por jQuery. Es indistinto.

MÉTODO ON

MÉTODO ON

Con el método `on()` podemos asignar eventos a elementos del DOM. Es una opción al método `addEventListener()` de JS Vanilla cuando usamos jQuery

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

```
// Array de objetos para agregar información al DOM.
const productos = [{ id: 1, nombre: "Arroz", precio: 125 },
{ id: 2, nombre: "Fideo", precio: 70 },
{ id: 3, nombre: "Pan" , precio: 50},
{ id: 4, nombre: "Flan" , precio: 100}];
// Recorremos el array con for..of
for (const producto of productos) {
  //Por cada producto además de los datos agregamos un botón
  $("#app").append(`<div>
    <h4> Producto: ${producto.nombre}</h4>
    <b> $ ${producto.precio}</b>
    <button id="btn${producto.id}">Comprar</button>
  </div>`);
  //Asociamos el evento a botón recién creado.
  $('#btn${producto.id}`).on('click', function () {
    console.log(`Compraste ${producto.nombre}`);
  });
}
```

EJEMPLO APLICADO: ASOCIAR EVENTO A BOTÓN CREADO

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

MÉTODOS SHORTCUT

CLICKO

Este 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 recibir 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);
});
```

CHANGE()

Este 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);
});
```

SUBMITO

```
//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);
});
```

Este método `submit()` es un atajo para `.on("submit", manejador)`.

El evento `submit` se dispara cuando el usuario envía un formulario. Sólo disponible para elementos `<form>`.

MÉTODO TRIGGER

MÉTODO TRIGGER

El método trigger() dispara un evento específico y el comportamiento predeterminado de un evento (como el envío de formularios) para los elementos seleccionados. Se usa para activar eventos de otros elementos.

```
//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");
});
```

EJEMPLO APLICADO:

INPUT HIDDEN

```
// Array de objetos para agregar información al DOM.
const productos = [{ id: 1, nombre: "Arroz", precio: 125 },
{ id: 2, nombre: "Fideo", precio: 70 },
{ id: 3, nombre: "Pan" , precio: 50},
{ id: 4, nombre: "Flan" , precio: 100}];

// Asociamos el evento click en ready luego del DOM Generado
$(document).ready(function () {
    $(".btnComprar").click(function (e) {
        //Obtenemos hijos del padre <div> desde el target
        let hijos = $(e.target).parent().children();
        //Primer input, valor de ID oculto
        console.log(hijos[0].value);
    });
});

// Recorremos el array con for..of
for (const producto of productos) {
    //Por cada producto además de los datos agregamos un botón
    $("#app").append(`<div>
        <input value=" ${producto.id}" type="hidden">
        <h4> Producto:  ${producto.nombre}</h4>
        <b> $  ${producto.precio}</b>
        <button class="btnComprar">Comprar</button>
    </div>` );
}
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE



INCORPORAR JQUERY AL PROYECTO

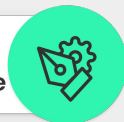
Suma al proyecto integrador los conceptos de jQuery que vimos en las últimas dos clases.

INCORPORAR JQUERY AL 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: Recuerda que jQuery es una librería que simplifica la notación JS. Es posible reemplazar con selectores todos los métodos nativos de acceso al DOM. Así como reemplazar toda definición de eventos de vanilla JS por on o métodos shortcut

Desafío
entregable



>> Consigna: Sumar al proyecto integrador los conceptos de jQuery que vimos en las últimas dos clases:

- Utilizar métodos jQuery para incorporar elementos al DOM.
- Utilizar métodos jQuery para determinar respuesta a ciertos eventos.

>>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 para seleccionar, agregar y definir eventos.

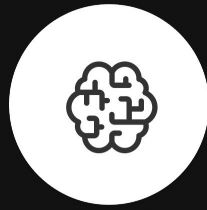
>>Ejemplo:

Manejo de eventos del proyecto: clicks del usuario, cambios en inputs, selectores, etc
Cualquier modificación que necesites hacer sobre el DOM con la página ya cargada: por ejemplo, al seleccionar una opción de un selector aparece una alerta en HTML dando cierta información.
Capturar el evento asociado a presionar ENTER para confirmar el envío de los datos.

CODER HOUSE

¿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



- jQuery |
[GitBooks. jQuery](#)
[Manual Básico de jQuery.](#)
[Tutorial instalación jQuery.](#)
- Documentación |
[Documentación jQuery](#)

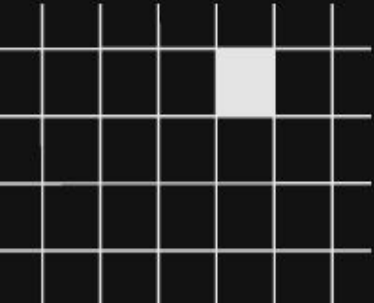
Disponible en [nuestro repositorio](#).

CODER HOUSE



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Manejo de eventos en jQuery.
 - Ready() y load
 - Métodos on y Métodos Shortcut
 - Trigger
- 



OPINA Y VALORA ESTA CLASE