jQuery 3

Contenido

[1.INTEGRACION DE JQUERY. 3](#_Toc56707451)

[2. SELECTORES. 3](#_Toc56707452)

[2.1 SELECTOR DE ID. 4](#_Toc56707453)

[2.2 SELECTOR DE CLASE. 4](#_Toc56707454)

[CAMBIAR BORDE AL DARLE CLICK 5](#_Toc56707455)

[2.3 SELECCIÓN DE ETIQUETAS. 6](#_Toc56707456)

[2.4 SELECTOR DE ATRIBUTOS. 7](#_Toc56707457)

[2.5 SELECTOR DE MAS DE 1 ELEMENTO. 7](#_Toc56707458)

[2.6 FIND. 7](#_Toc56707459)

[2.7 PARENT. 8](#_Toc56707460)

[3. EVENTOS. 9](#_Toc56707461)

[3.1 MOUSEOVER Y MOUSEOUT. 9](#_Toc56707462)

[3.2 HOVER. 10](#_Toc56707463)

[3.3 CLICK Y DOBLE CLICK. 10](#_Toc56707464)

[3.4 FOCUS Y BLUR. 11](#_Toc56707465)

[3.5 MOUSEDOWN , MOUSEUP Y MOUSEMOVE 11](#_Toc56707466)

[3.5.1 EJEMPLO MOUSEMOVE(PUNTERO RATON CON UN DIV). 12](#_Toc56707467)

[4. EFECTOS Y ANIMACIONES. 13](#_Toc56707468)

[4.1 TRABAJO CON EL DOM Y LOS TEXTOS. 13](#_Toc56707469)

[4.1.1 AÑADIR A UN ENLACE TEXTO DEL href. 13](#_Toc56707470)

[4.1.2 MEJORA EJERCICIO. 14](#_Toc56707471)

[4.1.3 AÑADIR ATRIBUTOS. 16](#_Toc56707472)

[4.1.4 ELIMINAR ATRIBUTO. 16](#_Toc56707473)

[4.2 EFECTOS. 17](#_Toc56707474)

[4.2.1 OCULTAR Y MOSTRAR.hide() y show(). 17](#_Toc56707475)

[4.2.2 OCULTAR Y MOSTRAR.fadein() Y fadeout(), fadeTo(). 18](#_Toc56707476)

[4.2.3 toogle() . 19](#_Toc56707477)

[4.2.4 slideUp() y slideDown(). 19](#_Toc56707478)

[4.3. ANIMACIONES. 20](#_Toc56707479)

[4.3.1 METODO animate() . 20](#_Toc56707480)

[4.3.2 SECUENCIA DE ANIMACIONES. 21](#_Toc56707481)

[4.3.3 CALLBACKS EN ANIMACIONES. 22](#_Toc56707482)

[5. PETICIONES AJAX. 23](#_Toc56707483)

[5.1 LOAD. 23](#_Toc56707484)

[5.2 METODO GET. 24](#_Toc56707485)

[5.3 METODO POST 25](#_Toc56707486)

[5.4 METODO $.AJAX . 27](#_Toc56707487)

[6. JQUERY UI. 28](#_Toc56707488)

# 1.INTEGRACION DE JQUERY.

Vamos a descargar jQuery:

<https://code.jquery.com/>

<!--

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"

        integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0=" crossorigin="anonymous">

  </script>

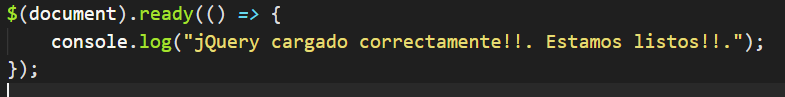
 -->

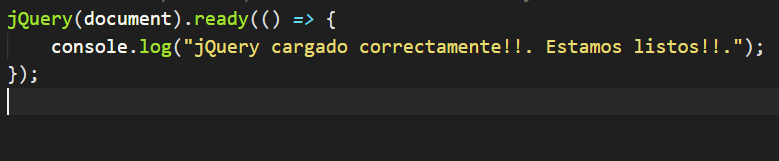
 <script type="text/javascript" src="js/jquery-3.5.1.min.js">

</script>

# 2. SELECTORES.

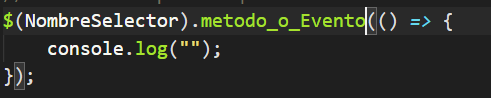
Vamos a usar el $ para referirnos al objeto jQuery, seria similar a poner jQuery:





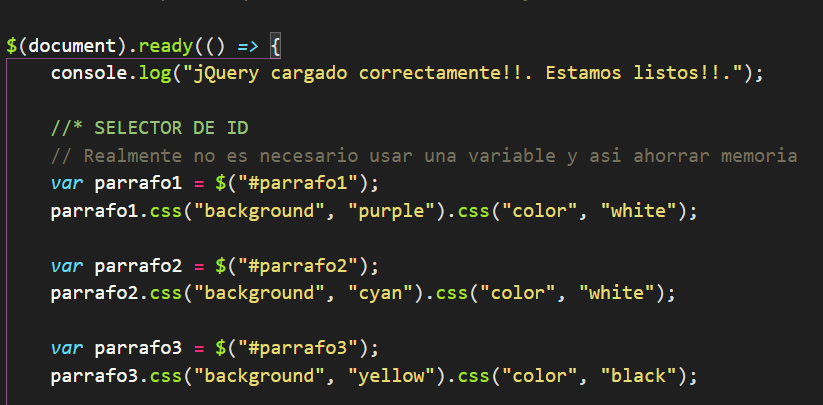
Por comodidad usamos $

**SINTAXIS**

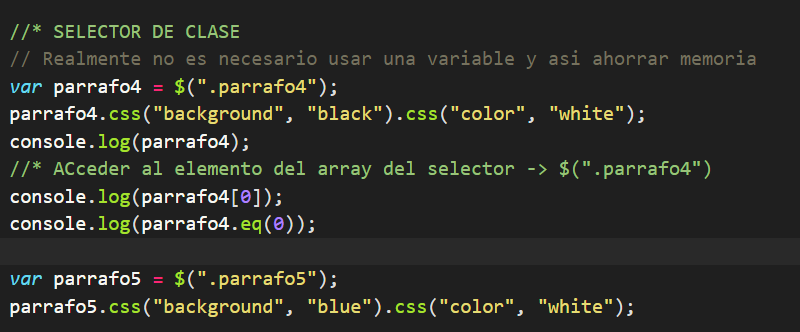


## 2.1 SELECTOR DE ID.

Cuando usemos jQuery lo primero que debemos de **comprobar es si el documento esta listo**. Asi que **seleccionamos el documento y el evento ready**, que recibe una función de callback.

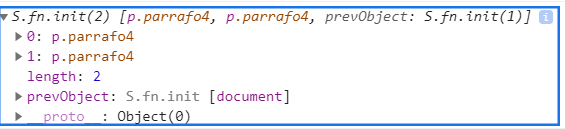


## 2.2 SELECTOR DE CLASE.



NOTA

Al usar el selector realmente se almacena como un array al cual podemos acceder:

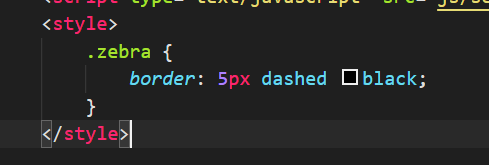


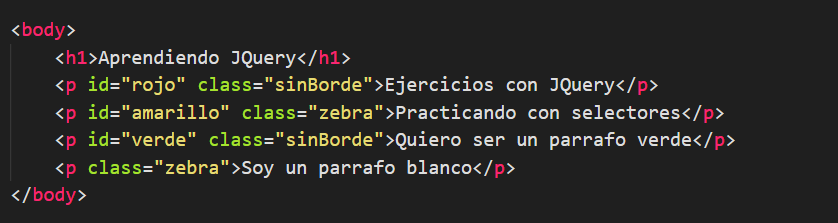
//\* Acceder al elemento del array del selector -> $(".parrafo4")

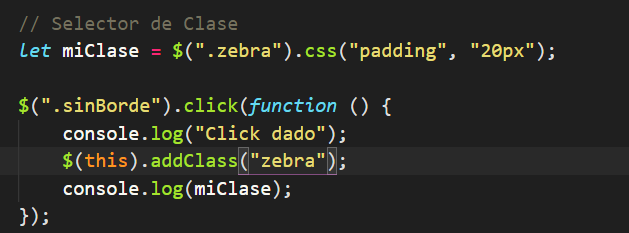
    console.log(parrafo4[0]);

    console.log(parrafo4.eq(0));

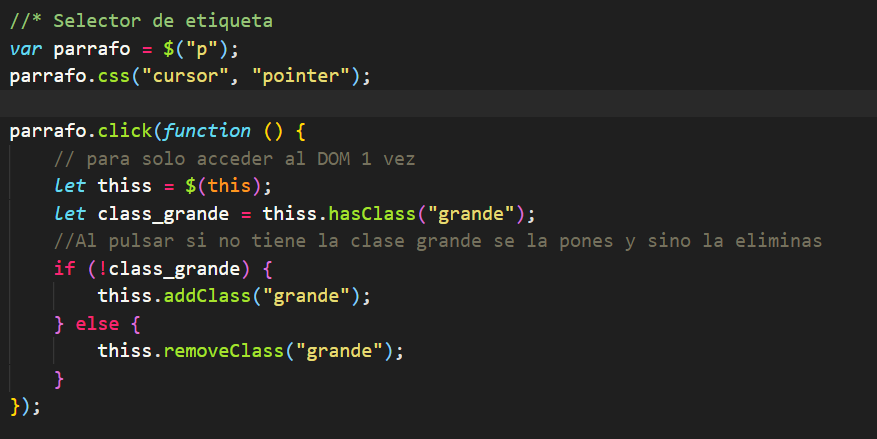
### CAMBIAR BORDE AL DARLE CLICK

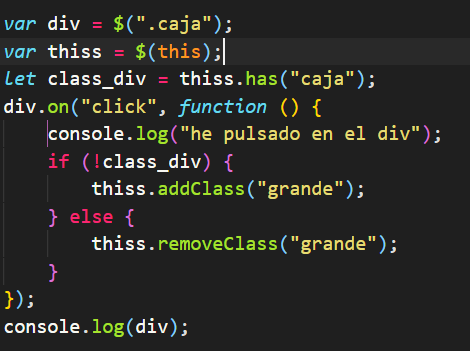






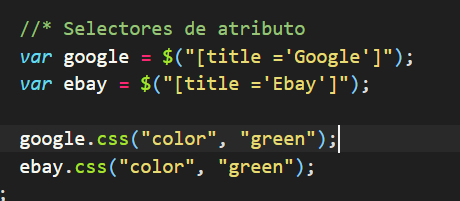
## 2.3 SELECCIÓN DE ETIQUETAS.





## 2.4 SELECTOR DE ATRIBUTOS.





## 2.5 SELECTOR DE MAS DE 1 ELEMENTO.

//\* Selector de mas de un elemento

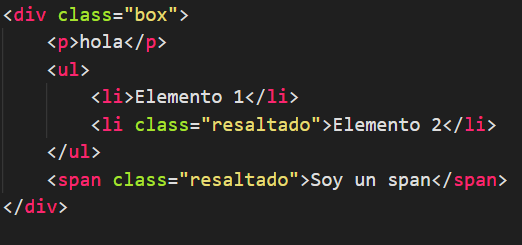
*var* seleccion = $("p, a");

    console.log(seleccion);

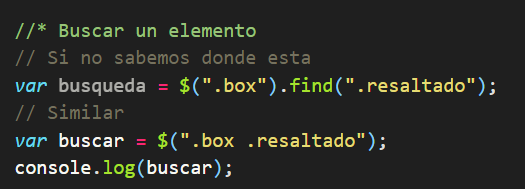
    seleccion.addClass("borde-superior");

## 2.6 FIND.

Buscar un elemento, en un contexto mas o menos complejo. Imaginemos que tenemos:



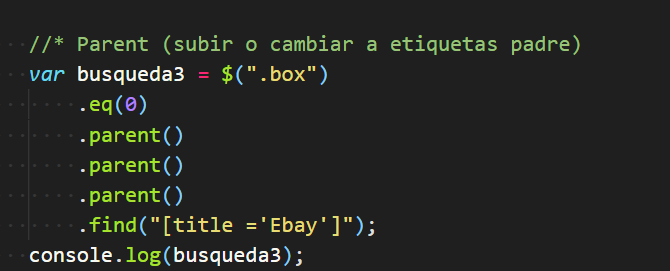
Sabemos que el elemento esta en el div con class box, pero no sabemos donde . Mediante el método find() podemos buscarlos:



## 2.7 PARENT.

Imaginemos que hemos seleccionado el div box y el span con class resaltado, y queremos salir a una etiqueta padre o a un párrafo de los que están fuera del div.

Si usamos 2 veces el método parent() pasaremos del span resaltado al div caja. Es decir por cada parent(), vamos subiendo de etiqueta.



# 3. EVENTOS.

## 3.1 MOUSEOVER Y MOUSEOUT.

//! .ready is deprecaded

/\* $(document).ready(function () {

    alert("eventos.js cargado");

}); \*/

// Similar a lo de arriba

$(*function* () {

    // Handler for .ready() called.

    //\* MouseOver y MouseOut

*var* caja = $("#caja");

    console.log(caja);

    //! .mouseover() is deprecaded usr .on("evento", function)

    /\*  caja.mouseover(function () {

        console.log("mouseOver");

        $(this).css("background", "red");

    }); \*/

    /\*  caja.mouseout(function () {

        console.log("mouseOut");

        $(this).css("background", "yellow");

    }); \*/

    caja.on("mouseover", *function* () {

        console.log("mouseOver");

        $(this).css("background", "red");

    });

    caja.on("mouseout", *function* () {

        console.log("mouseOut");

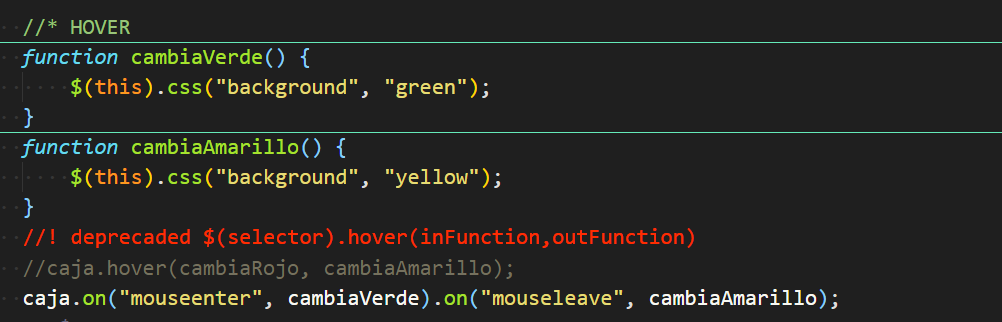
        $(this).css("background", "yellow");

    });

});

## 3.2 HOVER.

Podemos hacer los mismo que hicimos con el mouseover y mouseout con el evento hover: para ello vamos a definir 2 funciones :

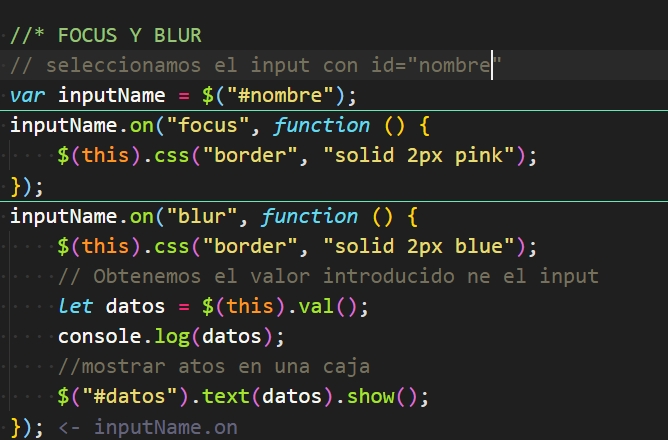


## 3.3 CLICK Y DOBLE CLICK.

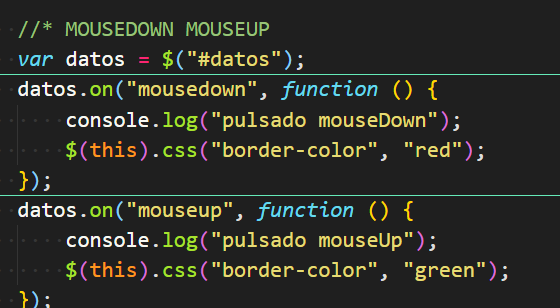


## 3.4 FOCUS Y BLUR.

Los eventos focus y blur, se ejecutan cuando por ejemplo en los inputs de un formulario.



## 3.5 MOUSEDOWN , MOUSEUP Y MOUSEMOVE



 // Mousemove

    $(document).on("mousemove", *function* () {

*var* pageCoords = "( " + event.pageX + ", " + event.pageY + " )";

*let* x = event.clientX;

*let* y = event.clientY;

        console.log(`Coord X: ${x}`);

        console.log(`Coord Y: ${y}`);

        console.log(pageCoords);

    });

### 3.5.1 EJEMPLO MOUSEMOVE(PUNTERO RATON CON UN DIV).

  $(document).on("mousemove", *function* () {

  $("body").css("cursor", "none");

*let* coord\_x = event.clientX;

*let* coord\_y = event.clientY;

*let* div\_sigueme = $("#sigueme");

        //  Mov Eje X - Eje Y

        div\_sigueme.css("left", coord\_x);

        div\_sigueme.css("top", coord\_y);

    });

# 4. EFECTOS Y ANIMACIONES.

El 4.1 todavia no trata los efectos pero es un ejercicio DOM

## 4.1 TRABAJO CON EL DOM Y LOS TEXTOS.

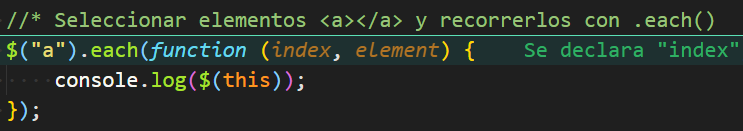
### 4.1.1 AÑADIR A UN ENLACE TEXTO DEL href.

Vamos a trabajar con los textos. Vamos implementar que en un menu lea los enlaces y ponga el texto del enlace en la etiqueta <a>, la cual esta vacia

Vamos a crear un menú que nos lleve a diferentes paginas, los cuales están vacios, y mediante js pondremos el texto de dichos enlaces:



Vamos a seleccionar todos los elementos <a> y recorrerlos mediante el metodo **each()**:



Vamos a seleccionar los enlaces y luego mediante el método **text()**, insertarlos en el enlace:

* Recorrer todos los elementos <a></a>
* Seleccionar attr de dichos elementos
* Añadirlos mediante el método **text()**

$(*function* () {

    //console.log($("a").length);

    //\* Seleccionar elementos <a></a> y recorrerlos con .each()

    $("a").each(*function* (*index*, *element*) {

        //console.log($(this));

*let* elemento = $(this);

//seleccionamos attr href;

*let* enlace = elemento.attr("href");

        // Insertamos el texto en el enlace

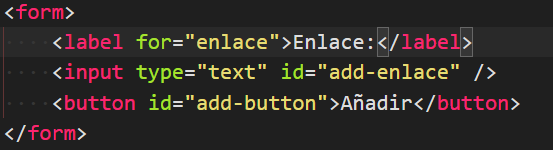
        elemento.text(enlace);

    });

});

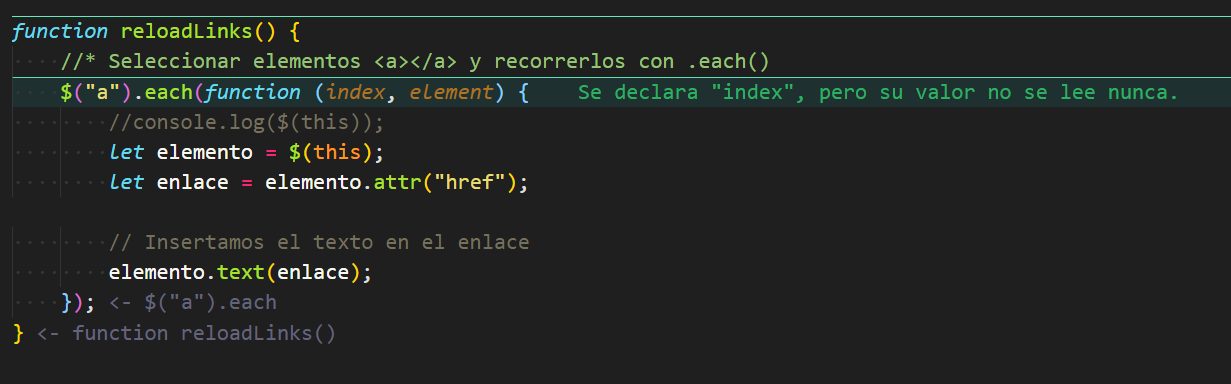
### 4.1.2 MEJORA EJERCICIO.

Vamos a añadir un campo de un form para ir añadiendo enlaces a nuestro menú, y que Js los ponga automáticamente en la etiqueta <a></a>, como hicimos anteriormente.



* Capturar valor del input al darle click al botón
* Seleccionar la lista con id=”menu”, y con el método **html()**, añadimos el valor NOS MACHACA LOS OTROS ENLACES
* USAR append()
* Insertar código que recorria los elementos <a></a> en una función **reloadLink()**, esta función la llamaremos nada mas cargar la pagina y cuando añadimos el valor a los enlaces

FUNCION





Si usamos **prepend()**, lo añadimos al principio de la lista.

Ademas tenemos un par de métodos interesantes:

* before() -> Lo posiciona antes del elemento que estamos seleccionando
* after() -> Lo posiciona despues del elemento que estamos seleccionando

Veamos como quedaría:

**before()**

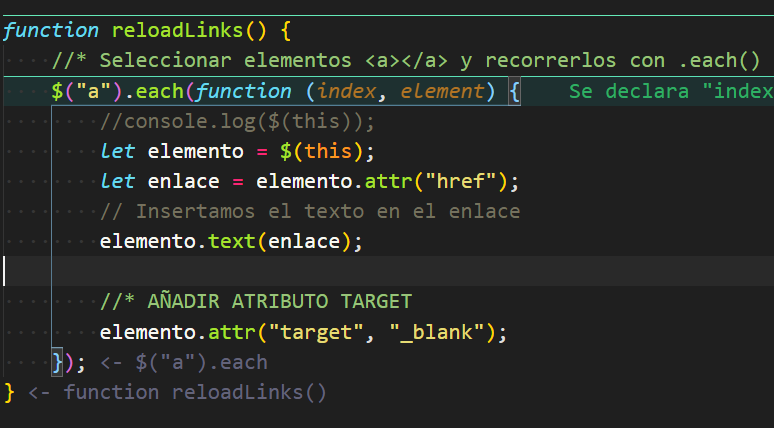


**after()**



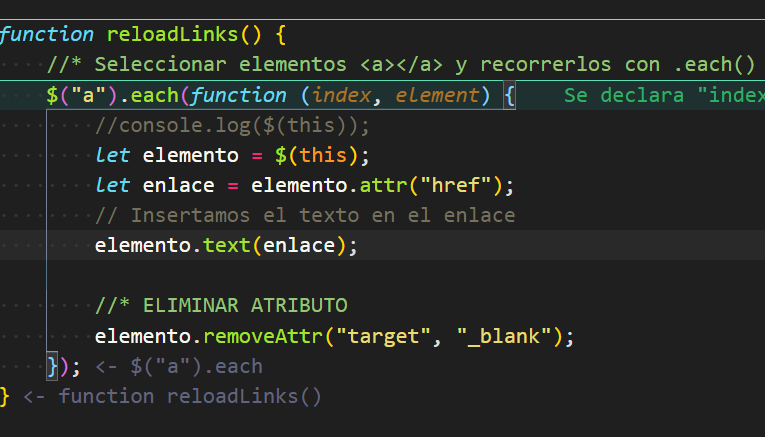
### 4.1.3 AÑADIR ATRIBUTOS.

Vamos a añadir a los elementos del menú el atributo target=”\_blank”, que nos abre el enlace en una nueva pestaña para ello en la función que selecciona los enlaces del menú usamos el método **attr()**



### 4.1.4 ELIMINAR ATRIBUTO.

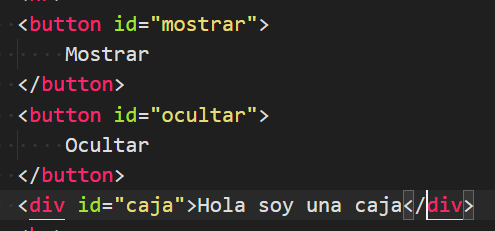
Utilizaremos el método **removeAttr()**



## 4.2 EFECTOS.

### 4.2.1 OCULTAR Y MOSTRAR.hide() y show().

Vamos a implementar 2 botones que muestren y oculten un div



$(*function* () {

*let* mostrar = $("#mostrar");

*let* ocultar = $("#ocultar");

*let* div = $("#caja");

    mostrar.hide();

    //\* Ocultar div

    ocultar.on("click", *function* () {

        console.log("pulsado ocultar");

        div.hide();

        mostrar.show();

        ocultar.hide();

    });

    //\* Mostrar div

    mostrar.on("click", *function* () {

        console.log("pulsado mostrar");

        div.show();

        ocultar.show();

        mostrar.hide();

    });

});

#### 4.2.1.1 TRANSICIONES AL OCULTAR/MOSTRAR.

Podemos modificar la velocidad de la transición al ocultarse y al mostrarse

div.hide("slow");

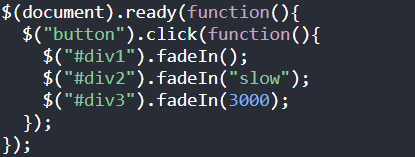
div.show("slow");

div.hide("fast");

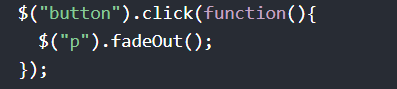
div.show("fast");

### 4.2.2 OCULTAR Y MOSTRAR.fadein() Y fadeout(), fadeTo().

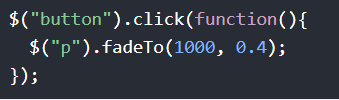
Con fadeIn() mostramos elemento



The fadeOut() method gradually changes the opacity, for selected elements, from visible to hidden (fading effect).

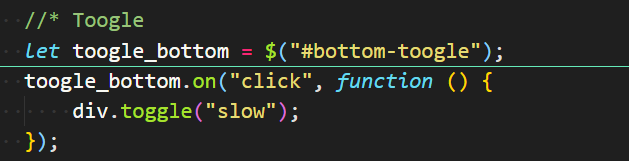


Cambia la opacidad de un párrafo en transicion



### 4.2.3 toogle() .

Mediante un toogle vamos a poder mostrar y ocultar elementos:



### 4.2.4 slideUp() y slideDown().

Son parecidos a hide y show pero los oculta/muestra en forma de persiana

## 4.3. ANIMACIONES.

### 4.3.1 METODO animate() .

$(*function* () {

    console.log("animaciones.js cargado!!");

    //\* Animaciones

*let* bottom\_animation = $("#bottom-animation");

*let* caja = $("#caja");

    bottom\_animation.on("click", *function* () {

        caja.animate(

            {

                marginLeft: "500px",

                with: "600px",

                height: "100px",

                fontSize: "40px",

                opacity: "0.5",

            },

            "slow"

        );

    });

});

### 4.3.2 SECUENCIA DE ANIMACIONES.

Podemos implementar un secuencia de animaciones encadenando el método **anímate()** :

$(*function* () {

    console.log("animaciones.js cargado!!");

    //\* Animaciones

*let* bottom\_animation = $("#bottom-animation");

*let* caja = $("#caja");

    bottom\_animation.on("click", *function* () {

        caja.animate(

            {

                marginLeft: "500px",

                with: "600px",

                height: "100px",

                fontSize: "40px",

                opacity: "0.5",

            },"slow")

            .animate(

                {

                    borderRadius: "999px",

                    marginTop: "100px",

                },

                "slow")

            .animate(

                {

                    borderRadius: "0px",

                    marginTop: "0px",

                },

                "slow")

            .animate({

                borderRadius: "0px",

                marginLeft: "0px",

            },"slow");

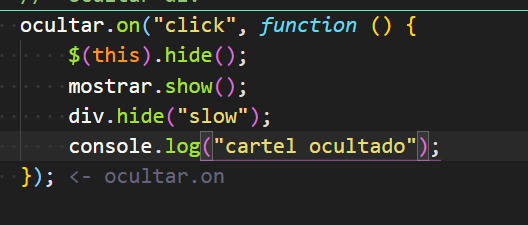
    });

});

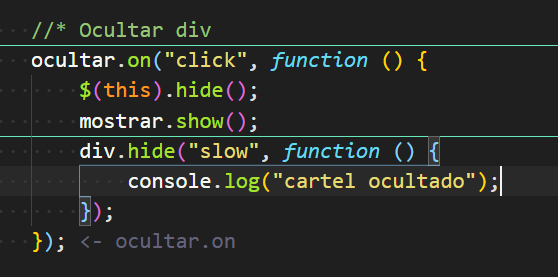
### 4.3.3 CALLBACKS EN ANIMACIONES.

Supongamos el ejemplo donde queremos ocultar un caja como hicimos antes y al ocultarse obtener un mensaje por consola como que la caja se ha ocultado para saber cuando ha finalizado la animación.

Inicialmente podríamos pensar en meter simplemente un console.log()



Asi vamos a tener un problema, que se muestra antes de que finalice la animación, para solucionar esto podemos implementar **una función de callback como 2º parametro** y asi se mostrara cuando acabe la animación.



# 5. PETICIONES AJAX.

Ajax nos permite realizar peticiones asíncronas a un servidor y recoger los resultados sin necesidad de recargar la pagina.

Las peticiones AJAX se pueden utilizar de varias maneras:

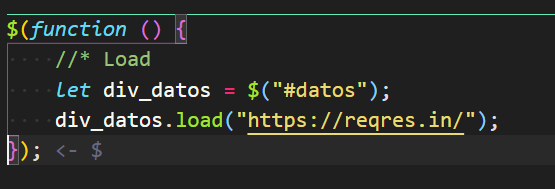
* Obtener código HTML e Incrustarlo en cualquier elemento de la pagina.
* Obtener de la petición un JSON y trabajar con el JSON de la manera que necesitemos.

En aplicaciones monolíticas donde tenemos un backend y un frontend mezclado(App creadas completamente con laravel o symfoby u otro framework), van a tener peticiones AJAX a una URL y esta petición nos devuelve código HTML.

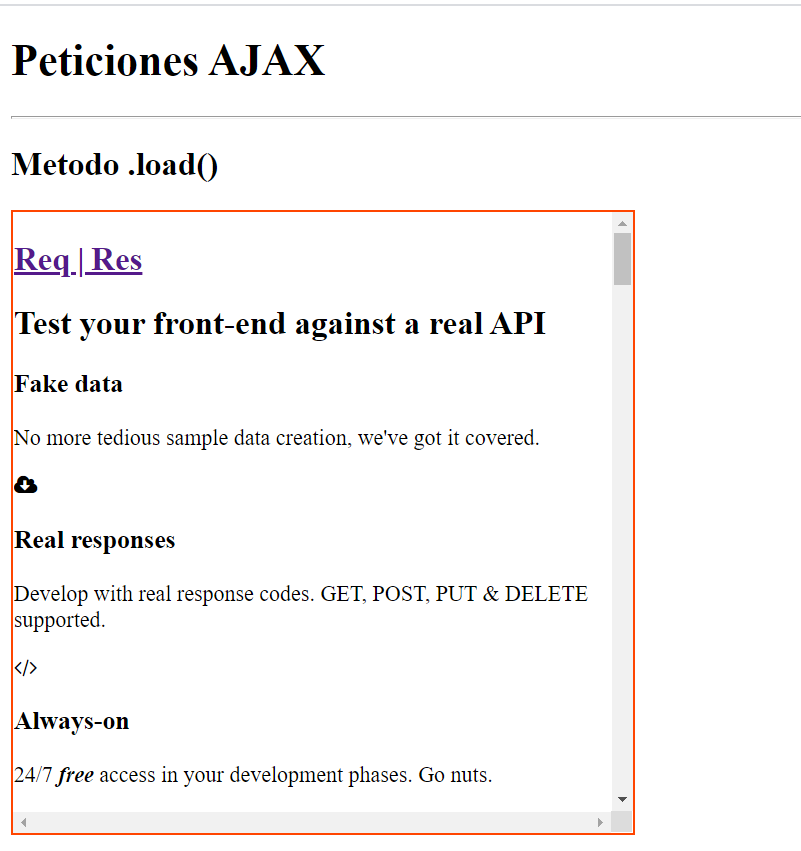
Vamos a usar una FAKE API para ver como funciona: -> <https://reqres.in/>

## 5.1 LOAD.

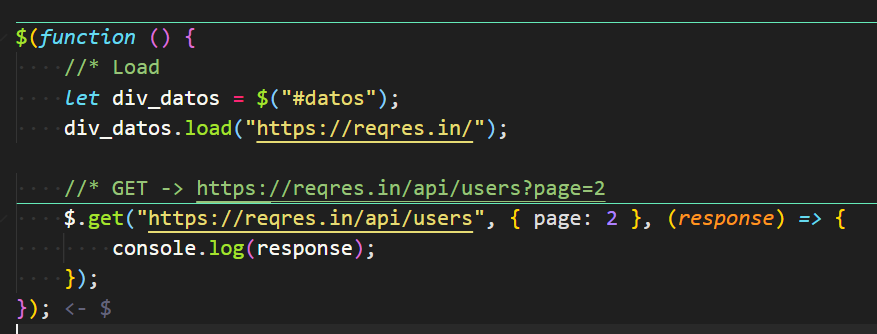
Nos permite realizar una petición AJAX por GET a una URL e incrustar el resultado en un div o cualquier elemento de la pagina. Suele devolver código html.



Si mostramos el div

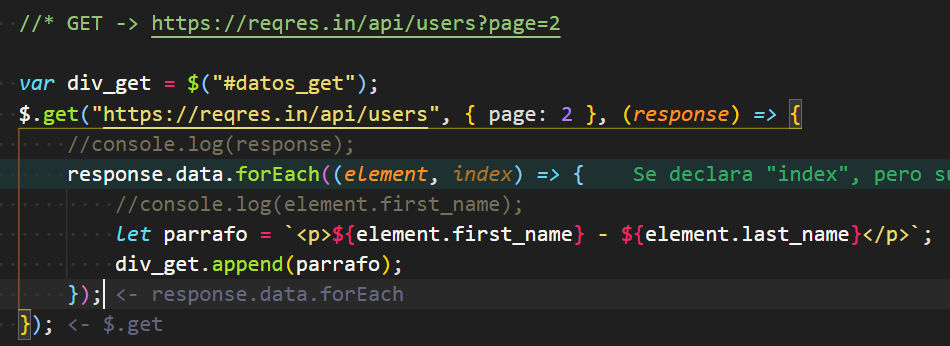


## 5.2 METODO GET.





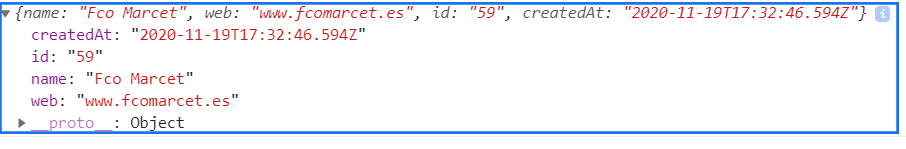
Ahora podemos recorrer el array, accediendo a su índice data que es donde esta lo que nos interesa y recorriendolo



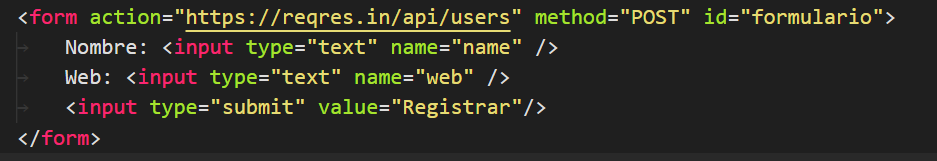
## 5.3 METODO POST

Vamos a hacer una petición al servidor para registrar un usuario.





Vamos a implementar un formulario para verlo mejor



//\* POST

*var* formulario = $("#formulario");

  //! NO USAR ARROW FUNCTIONS EN JQUERY

  formulario.on("submit", *function* (*e*) {

    // al enviar los datos que no nos redirija a

    //  la url que envia los datos el form

    e.preventDefault();

    //obtener campos del form

*let* nombre = $("input[name='nombre']").val();

//let nombre =$("#formulario").name

*let* pagina\_web = $("input[name='web']").val();

//let pagina\_web = $("#formulario").web;

*var* usuario = {

          name: nombre,

          web: pagina\_web,

      };

      // Obtener url de la action del for

*let* url\_action\_form = $(this).attr("action");

      //\* peticion POST

      $.post(url\_action\_form, usuario, *function* (*response*) {

         // console.log(response.name);

*let* div\_post = $("#datos\_post");

*let* parrafo =`<p>${response.name}-${response.web}</p>`;

          div\_post.append(parrafo);

      }).done(*function* () {

          alert("Usuario añadido correctamente");

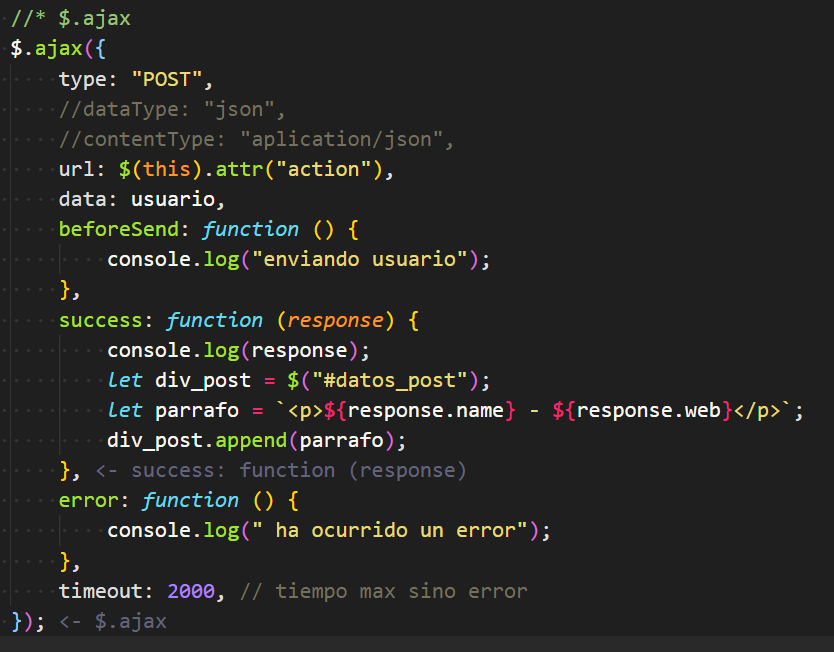
      });

      return false; // para que no no redirecione al action

  });

## 5.4 METODO $.AJAX .

Este método nos permite configurar mucho mas la petición AJAX:



# 6. JQUERY UI.