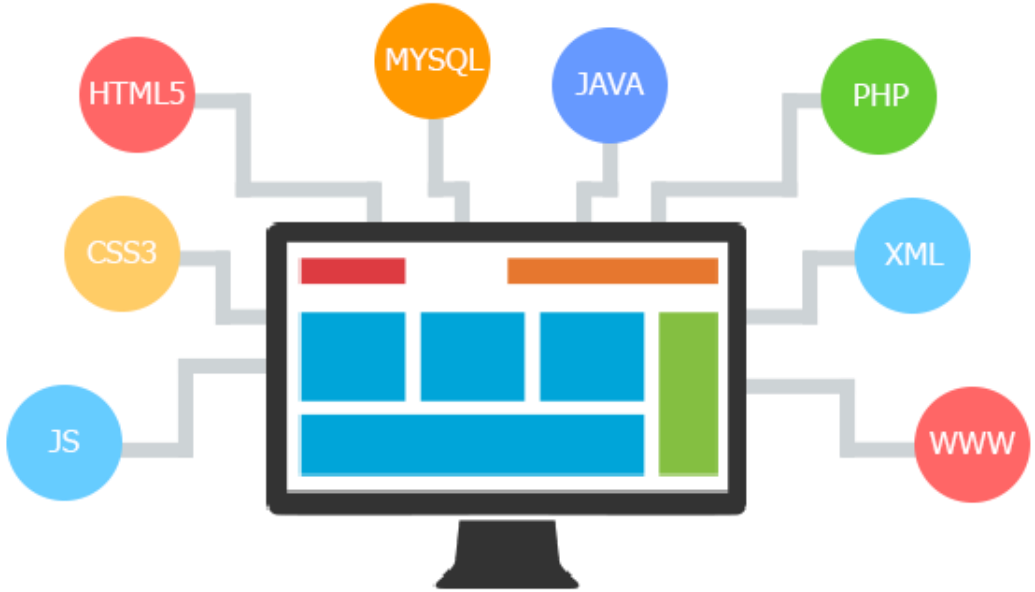


Desarrollo de Aplicaciones Web

Librería de JavaScript

Índice de contenidos

- JQuery es una librería que reduce considerablemente el código escrito y ofrece una gran cantidad de problemas resueltos para no tener que programarlas una y otra vez.

- Sobre esta librería existen extensiones (plugins) que se pueden incorporar (JQuery validation, JQueryUI, JQueryMobile, etc) para hacer más completas y mas fáciles de programar nuestras aplicaciones web.

- JQuery no es algo diferente a JavaScript ya que son un conjunto de definiciones nuevas para hacer lo mismo.

- En un principio JQuery esta ampliamente soportado y en teoría no debería quedarse anticuado, aunque los puristas de JavaScript no les agrada mucho.

- Es interesante no depender de JQuery y en primer lugar tener una buena base en JavaScript.

- Para usar JQuery hay varias opciones donde la primera decisión es si la descargamos o usamos un CDN (Content Delivery Network) para enlazarla desde Internet.

- A nivel de código no existe ninguna diferencia.

- Existen diversos servidores de otras empresas que mantienen versiones de la librería, además del oficial de JQuery, como Microsoft y Google.

- Si queremos descargarla debemos dirigirnos a https://jquery.com/download/ y descargamos la versión compressed (minificada) donde los espacios, saltos de línea y tabulaciones innecesarias están eliminadas y por tanto ocupa menos.

- Si optamos por usar un CDN, el oficial de JQuery es https://code.jquery.com/ donde obtendremos un código HTML que deberemos copiar en nuestra aplicación.

Inclusión de JQuery y comparación con JavaScript

```
<script type="text/javascript" src="jquery-3.7.1.min.js"></script>
```

.

```
//JavaScript Puro
document.addEventListener("load",
    ()=>{
        aler("Pagina cargada");
    });

document.addEventListener("click",
    ()=>{
        document.body.style.backgroundColor="cyan";
    });

let imagenes=document.querySelectorAll("img");
for (let i = 0; i < imagenes.length; i++) {
    imagenes[i].addEventListener("click",
        ()=>{
            alert("Hola");
        });
}
```

```
//JQuery
$(document).ready(
    ()=>{
        alert("Pagina cargada");
    });

$("body").click(
    ()=>{
        $("body").css("background-color","cyan");
    });

$("img").click(
    ()=>{
        alert("Hola");
    });
```

Más cosas interesante con JQuery

```
$("body").hide();
$("body").hide(3000);
//Encadenamiento de funciones o chaining
$("body").hide().show(3000);
$("body").hide().fadeIn(3000);
$("img").hide().width(300).height(500).css("border","3px solid red").slideDown(3000);
```

Todo esto no deja de ser JavaScript ¡¡¡¡

- No podemos olvidar que para hacer cualquier operación con el DOM los elementos de la pagina deben estar cargados o listos.

- En otro caso podría resultar que un código correcto no funcionase porque el elemento no exista.

```
$(document).ready(
    ()=>{
        /*Todo el codigo necesario para
          nuestra pagina web*/
    });
```

Funciones DOM para acceder al HTML

```
let encabezado=document.getElementByTagName("h1");
let id_noticias=document.getElementById("noticias");
let articulos=document.getElementByClassName("articulo");
```

Es deducible que JQuery nos facilitara las cosas

```
let encabezado=$("h1");
let id_noticias=$("#noticias");
let articulos=$(".articulo");
```

Para saber el tipo de los elementos seleccionados

```
alert($("#principal").prop("tagName"));
```

Vemos que JQuery se basa en selectores CSS

```
//Podemos guardarlo en una variable
//si vamos a utilizarlo más veces
$(".clear");
$("#menu_principal");
$("div");
```

Podemos hacer las mismas combinaciones CSS

```
$("#principal a");
$("table .subrayado");
$('.iva.articulo');
$('p,h1,span');
$("input[type=submit]");
```

- Existen funciones para ver y modificar el contenido de los elementos de la pagina.

```
$("div#principal").text(); //visualiza el texto del elemento seleccionado con $
$("div#principal").text("Texto nuevo"); //modifica el texto del elemento seleccionado

$("div#principal").html(); //visualiza el HTML del elemento seleccionado con $
$("div#principal").html("<h1>HTML nuevo</h1>"); //modifica el HTML del elemento seleccionado
```

- También disponemos de las dimensiones y posición.

```
//Informacion dimensional del elemento boton
let caja=$("#boton");
alert("Ancho: "+caja.width()+" Alto: "+caja.height());
//Para modificar las dimensiones
caja.width(300);
caja.height(500);
```

Modificar propiedades CSS

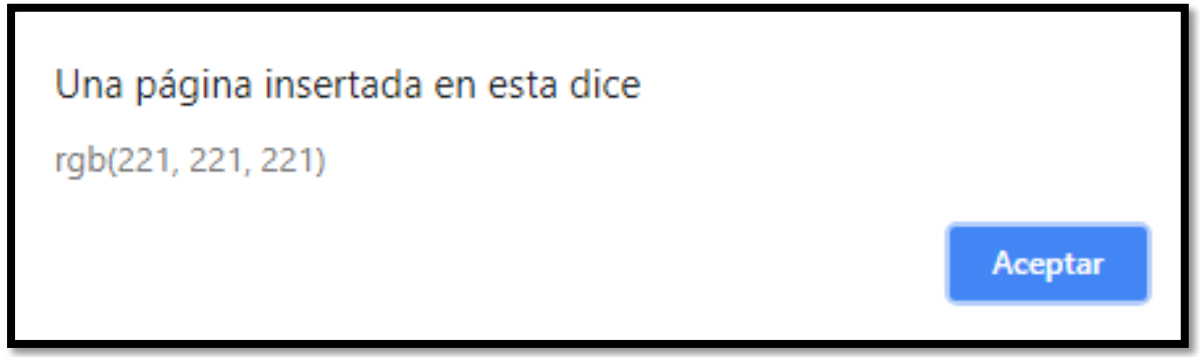
```
//Todos los parrafos a color rosa
$("p").css("background-color","pink");

//Todos los div con clase articulo a fuente 2em
$("div.articulo").css("font-size","2em");

//Varias propiedades en un sola linea
$("p").css({"background-color": "yellow", "font-size": "200%"});

//Si las propiedades son numericas
$("p").css( "border-width", "+=3" );
```

- La función CSS nos devuelve el valor actual de la propiedad.

```
$("#boton").css("font-family");
```

- Cuando la propiedad es un color devuelve una cadena con el RGB. Hay que fijarse en los espacios.

```
alert($("#boton").css("background-color"));
```



- Si la propiedad es numérica hay que aplicar parseInt si queremos realizar una comparación.

```
let texto=$("#secundario");
if(parseInt(texto.css("font-size"))>20)
{
    alert("El texto es demasiado grande");
}else{
    texto.css("font-size","+=1");
}
```

- Algunas veces es engorroso trabajar con varias propiedades por lo que se suele agrupar en clases CSS.

- Podemos manejar las clases CSS asociadas a un elemento de nuestra web.

```
let boton_ejemplo=$("#boton");
boton_ejemplo.addClass("boton_personalizado");
boton_ejemplo.removeClass("boton_personalizado");
//Efecto interruptor
if(boton_ejemplo.hasClass("boton_personalizado"))
{
   boton_ejemplo.removeClass("boton_personalizado");
}else{
   boton_ejemplo.addClass("boton_personalizado");
}
//JQuery sigue simplificandonos el trabajo
boton_ejemplo.toogleClass("boton_personalizado");
```

- Con la función attr podemos tomar el valor o modificar un atributo del elemento seleccionado.

```
//Ver un atributo
$("a").attr("href");
$("img").attr("src");
$("table").attr("border");

//Modificar un atributo
$("a").attr("href","ejemploDOM.html");
$("img").attr("src","encendida.gif");
$("table").attr("border","2");
```

- Para atributos fijos como disabled, required, etc se utiliza la función removeAttr

```
//Habiltamos un boton porque el formulario es correcto
//Le damos el foco para que se active al pulsar intro
$("#boton").removeAttr("disabled").focus();
```

- No siempre queremos aplicar una función a la selección completa. JQuery nos proporciona funciones para filtrar colecciones de objetos.

```
let primer=$("p").fist();
let ultimo=$("p").last();
let tercero=$("p").eq(2); //empieza en cero
```

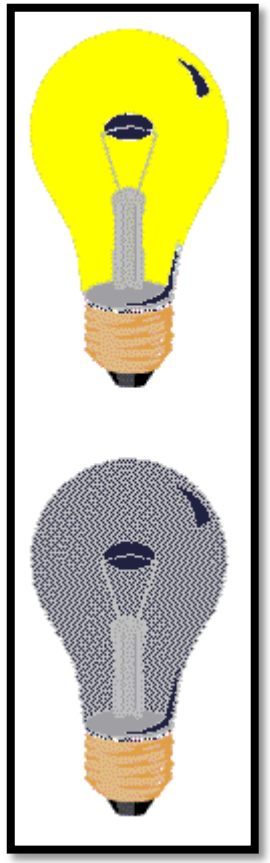
- Como podemos observar el operador $ no devuelve un array estilo corchetes [] por lo que hay que usar eq, last o first.
- Hasta ahora solo hemos visto el evento de clic y de carga(ready). Vamos a ver algunos más.

- Desde JQuery podemos gestionar eventos mediante funciones con su nombre.

```
$("p").click(...);
$("p").dblclick(...);
$("p").mouseenter(...);
$("p").mouseleave(...);
$("p").mousemove(...);
```

- Aunque no todos los eventos tienen una, por lo que existe la función genérica on PUNTOS SUSPENSIVOS

```
$("p").on("click",Pulsar);
```



```
let bombilla=$("img#bombilla").click(
    ()=>{
        if(bombilla.attr("src")=="apagada.gif"){
            bombilla.attr("src","encendida.gif");
        }else{
            bombilla.attr("src","apagada.gif");
        }
    });
```

- En este caso al contrario que con JavaScript puro la función attr solo devuelve el nombre del archivo y funcionaria.

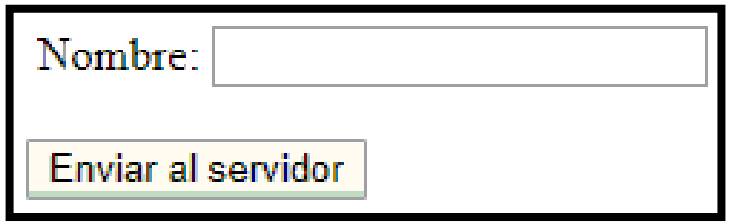
- El resto de eventos no vistos que tiene que ver con el teclado son más típicos en los formularios.

- Para acceder a la información que contiene un elemento de formulario HTML no se usa la función HTML si no la función val.

- Eventos muy relacionados a los formulario son change, blur, input, keypress , etc.

- Usados sobre todo para evitar valores inválidos en un formulario conforme lo escribimos.

- Los siguientes códigos controlan el contenido de un formulario como el siguiente:




```
let boton=$("input[type=submit]").click(
    (evento)=>{
        let nombre=$("#nombre").val();
        if(nombre==""){alert()
            alert("Nombre vacio");
            evento.preventDefault();
        }else if(nombre.length<10){
            alert("Nombre vacio");
            evento.preventDefault();
        }else if(nombre.length>20){
            alert("Nombre vacio");
            evento.preventDefault();
        }
        alert("Nombre correct,Adios¡¡¡");
    });
```

- Cuando el formulario contiene fallos no debe enviarse al servidor.
- Para ello necesitamos un objeto de tipo event que hasta ahora lo habíamos ignorado.
- Dicho objeto contiene información acerca del evento producido y lo controla.
- La función preventDefault evita el comportamiento por defecto, en este caso el envío (submit) del formulario.

- Los eventos en JQuery se pueden aplicar de manera masiva.
- Eso esta bien si queremos que la respuesta sea exactamente la misma.
- En la mayoría de los casos la respuesta a un evento es un poco distinta dependiendo del objeto de la pagina que recibe el evento.
- No hay más que mirar atrás y la mayoría de los ejemplos hacen referencia al propio objeto que recibe el evento.

- Si queremos que cada imagen nos diga su atributo src podríamos pensar varias opciones. (No todas validas)

```
            //PRIMERA OPCION
let imagenes=$("img");
    imagenes.click(
    ()=>{
        //¿Como las diferenciamos? NO VALE
        alert(imagenes.eq(??).attr("src"));
    });
```

- Vamos a poner en contraposición cuando lo hacíamos con JavaScript puro

```
let imagenes=document.querySelectorAll("img");
for (let i = 0; i < imagenes.length; i++) {
    imagenes[i].addEventListener("click",
        ()=>{
            alert(imagen[i].src);
        });
}
```

```
            //SEGUNDA OPCION
let imagenes=$("img");
    for (let i = 0; i < imagenes.length; i++) {
     imagenes.eq(i).click(
        ()=>{
            alert(imagenes.eq(i).attr("src"));
        });
    }
    //SERIA LO MISMO QUE NO USAR JQUERY
```

- Como no tenemos acceso a cada variable individual necesitamos el objeto event que nos proporciona target que es el elemento concreto que recibe el evento.

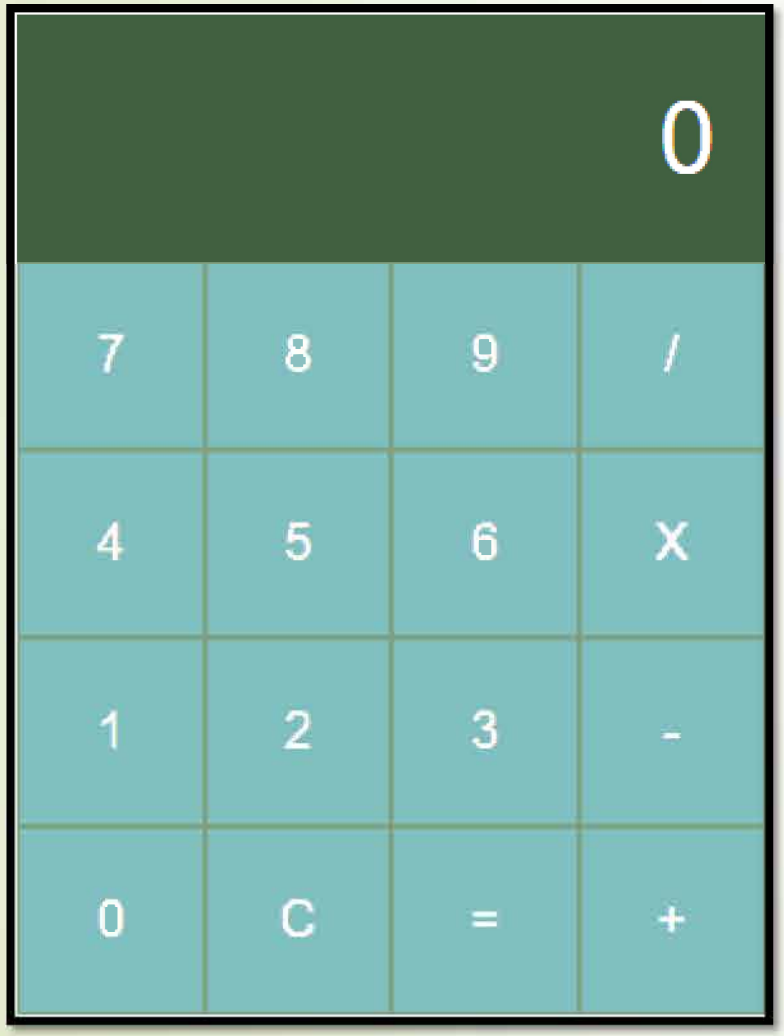
```
                //TERCERA OPCION
let imagenes=$("img");
    imagenes.click(
    (evento)=>{
        alert($(evento.target).attr("src"));
    });
```

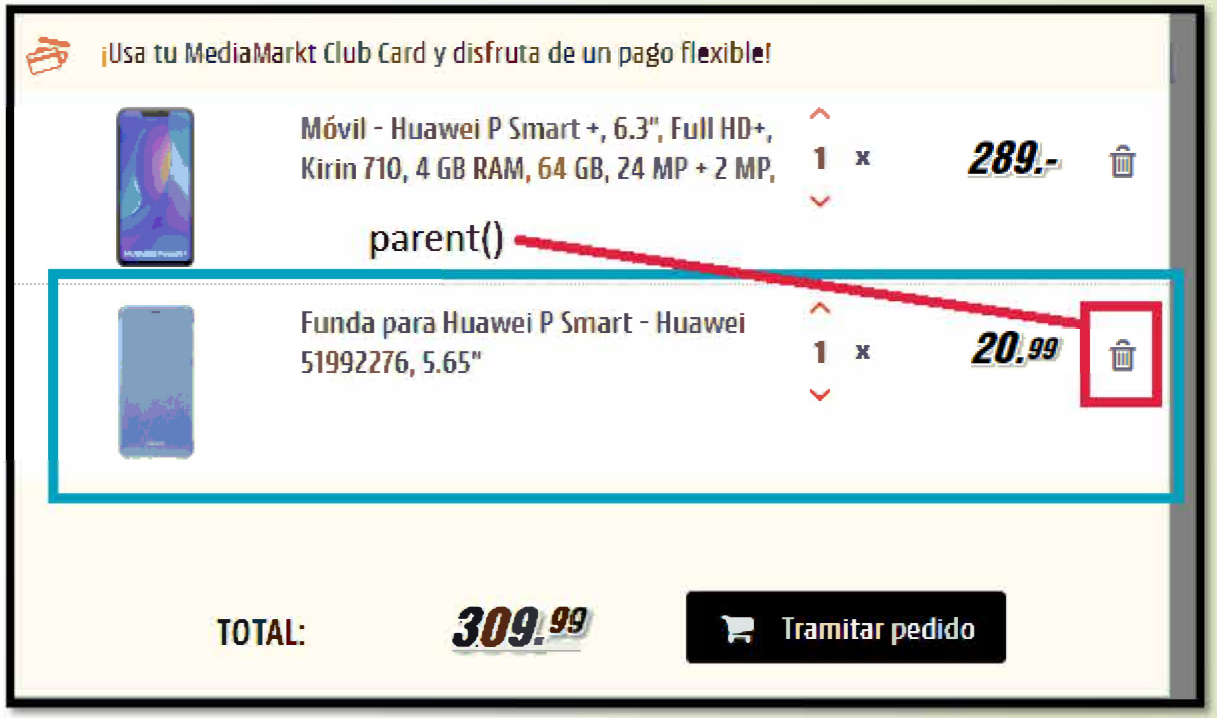
- Da igual que sean 1 o 1000 cada una se refiere a si misma (target) sin necesidad de repetir código.

```html
<a href="destino1.html">Enlace a destino1</a>
<a href="destino2.html">Enlace a destino2</a>
<a href="destino3.html">Enlace a destino3</a>
<a href="destino4.html">Enlace a destino4</a>
```

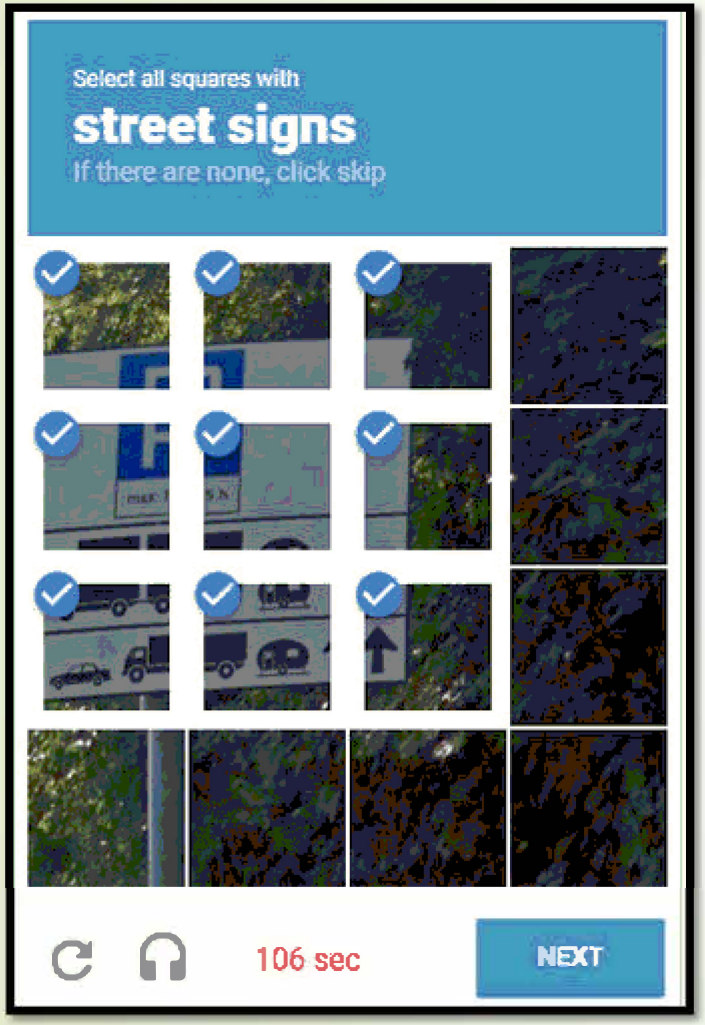
```javascript
let enlaces=$("a");
enlaces.click(
    (evento)=>{
        alert($(evento.target).attr(href));
    });
```

¿En que resultaría útil el event.target en estas apps?

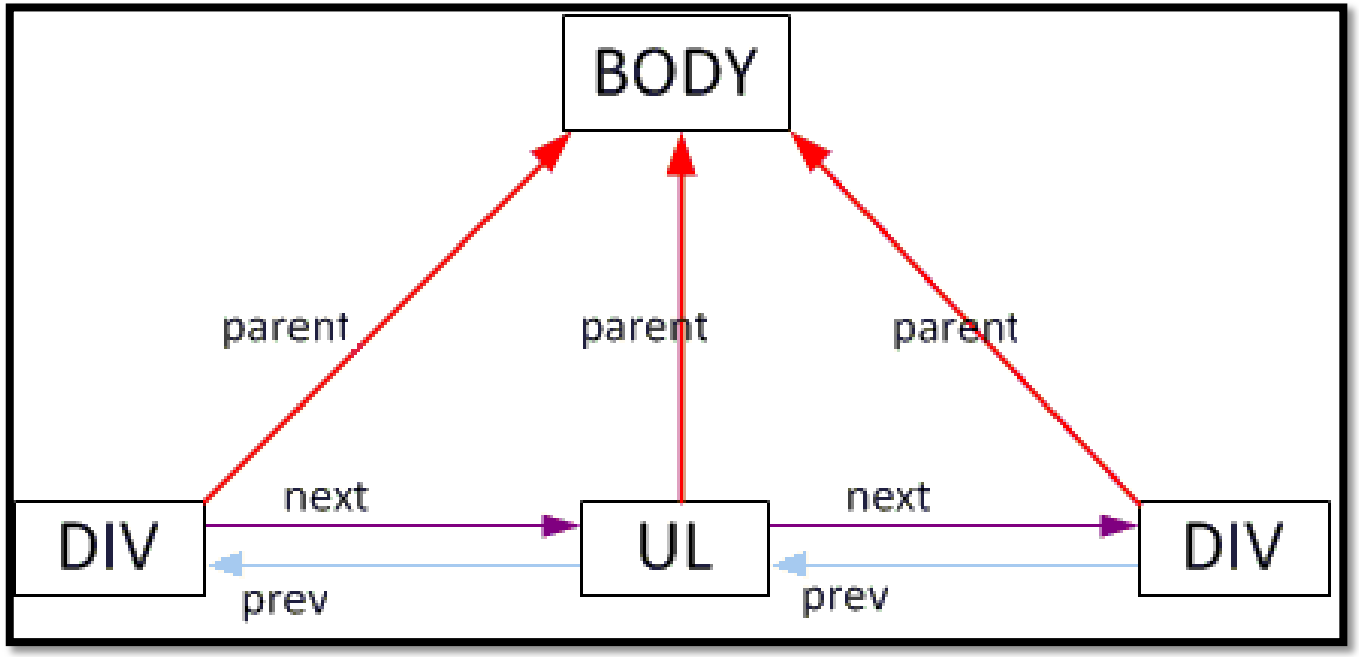




Otras situaciones donde el event.target salva la vida

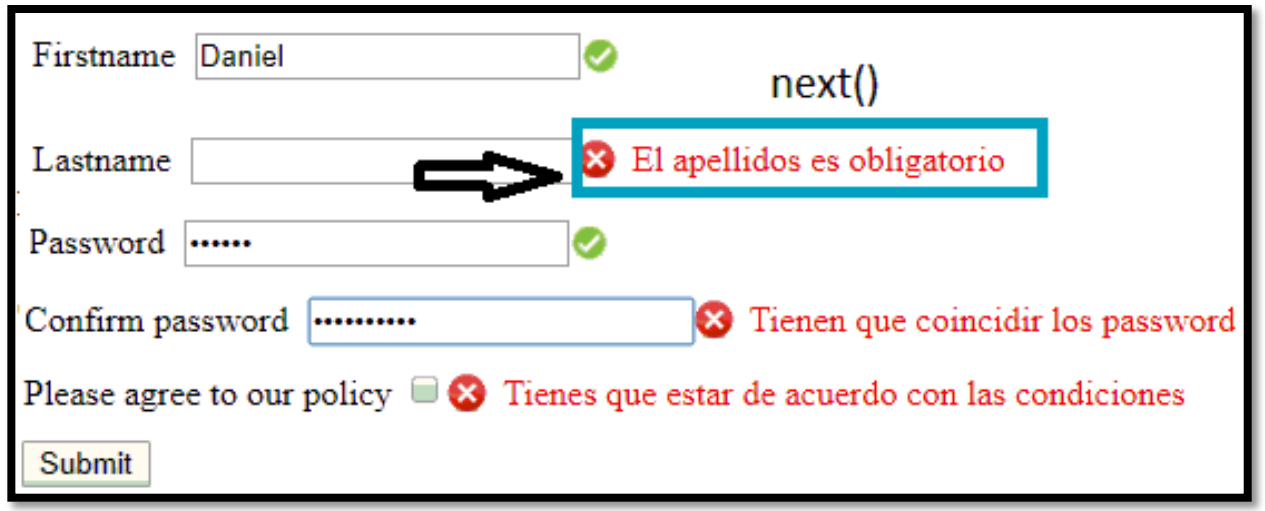


- JQuery nos permite hacer traversing en el DOM o lo que es lo mismo tener acceso directo a los nodos que rodean a uno en concreto.



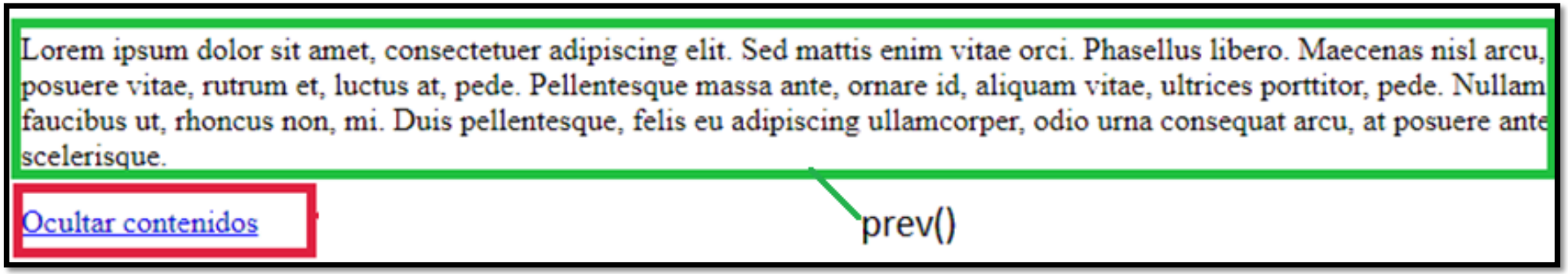
- Dichas funciones son útiles es situaciones comunes.

- En los formularios a la derecha de cada input puede haber un span sin contenido.



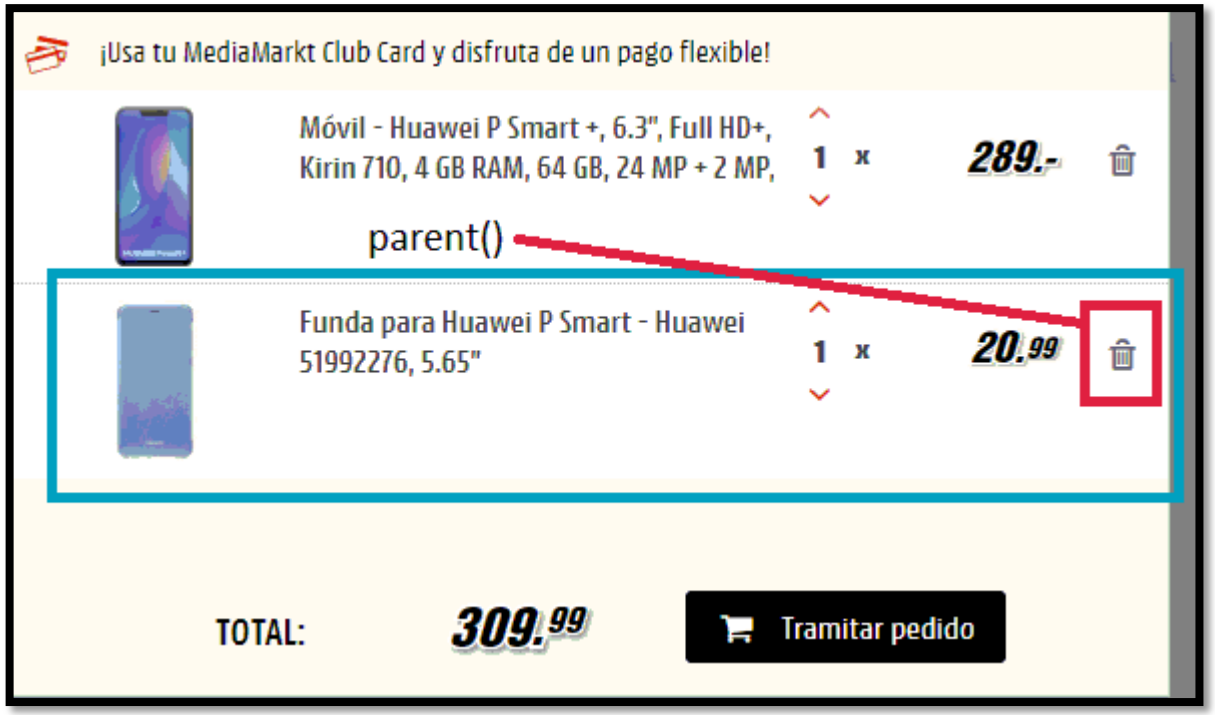

```
let apellidos=$("#apellidos");
apellidos.blur(
    (evento)=>{
        let contenido=$(evento.target);
        let aviso=$(evento.target).next();
        if(contenido.val()==""){
            aviso.text("Apellido obligatorio")
        }
    });
```

- Al pulsar el icono del cubo de basura hay que borrar es que esta su izquierda (prev).




```
let enlaces=$("a");
enlaces.click(
    (evento)=>{
        let parrafo=$(evento.target).prev();
        parrafo.hide();
    });
```

- Al pulsar el icono del cubo de basura hay que borrar al a su contenedor que es su elemento padre.




```
let botones=$("#rubbish-icon");
enlaces.click(
    (evento)=>{
        let producto=$(evento.target).parent();
        producto.remove();
    });
```

- Otras funciones interesantes para moverse en el árbol DOM y que incluyen **JQuery** son **children** y **find**.
- **children** devuelve todos los hijos directos.
- **find** devuelve todos los hijos directos o indirectos filtrados por selector.
- **children** también admite filtros.

```
let elementos=$("div").children();
let elementos=$("div").children("p.first");
//Todos los descendientes span
let elementos=$("div").find("span");
//Todos los descendientes
let elementos=$("div").find("*");
```

- Otro evento especial es el scroll, tanto en la forma de asignar una función al evento como la lectura del estado en que se encuentra el scroll.

- Es necesario para conseguir los típicos menús fijos (sticky) y el archiconocido efecto parallax.

```
$(document).scroll(
    ()=>{
        if($(document).scrollTop()>50){
            $("body").css("background-color","red");
        }else{
            $("body").css("background-color","inherit");
        }
    });
```

- Para borrar elementos con Jquery usamos la función remove también.
- Se aplica sobre una selección de elementos con $ directamente o sobre una variable.
- Tenemos que tener en cuenta que no solo desaparece de la página como hace hide.
- Elimina completamente el objeto JavaScript con sus datos, estilos asignados y eventos asignados.

```
$("p").remove();
//equivale tambien a
let parrafos=$("p");
parrafos.remove();
```

```
$("input[type=radio]").remove();
$("#nombre").remove();
$("table.especial").remove();
```

```
$("p").last().remove();
//equivalente a
let ultimo_p=$("p").last();
ultimo_p.remove();
```

- Para crear elementos nuevos en JQuery es muy sencillo con el operador $.

```
let nuevo_parrafo=$("<p>Nuevo parrafo con contenido</p>");
let nuevo_enlace=$("<a>Enlace creado con jquery</a>");
let nueva_imagen=$("<img></img>");
nueva_imagen.attr("src","encendida.gif");
let nuevo_div=$("<div>Nuevo div articulo</div>");
nuevo_div.addClass("articulo");
```

- Los elementos nuevo no aparecerán hasta que se añadan a la página, por ejemplo directamente al body con el método append.

```
$("body").append(nuevo_parrafo);
...
$("body").append(nuevo_div);
```

- Podemos añadir no solo al body, también a otros elementos.

```
let opcion=$("<option>Nueva opcion</option>");
opcion.attr("value","Nuevo");
$("select").append(opcion);
```

- También a elementos que acabamos de crear

```
let lista_nueva=$("<ul><li>Punto 1</li></ul>");
let nuevo1=$("<li>Punto al principio</li>");
let nuevo2=$("<li>Punto al final</li>");
lista_nueva.prepend(nuevo1);
lista_nueva.append(nuevo2);
```

- Observamos que prepend añade al principio del elemento y append añade también pero al final

- Podemos crear elementos que dependan de otras fuentes introducidos a través de variables.

```
let enlace=$("<a></a>");
enlace.addClass("boton");
enlace.attr("href",url);
enlace.text(texto);
```

- O directamente con el operador $

```
let enlace=$("<a class='boton' href='"+url+"'>"+texto+"</a>");
```

- Si queremos que los elementos que se creen dinámicamente reaccionen a distintos eventos tenemos que añadir el manejador en el mismo momento de la creación del elemento.
- Posteriormente añadirlos al árbol de la página.

- Para crear nuevo elementos con sus manejadores antes de añadirlos a body.

```
let nuevo_li=$("<li>Elemento con evento</li>";
nuevo_li.click(
    (evento)=>{
        let objetivo=$(evento.target);
        alert("Has pulsado el elemento"+objetivo.text());
        objetivo.remove();
    });
nuevo_li.dblclick(
    (evento)=>{
        let objetivo=$(evento.target);
        objetivo.css("background-color","red");
    });
```

- Podemos generar códigos HTML con gran cantidad de características usando las funciones de JQuery y bucles

```
let select_nuevo=$("<select></select>");
let opcion;
for(let i=1;i<=100;i++){
    opcion=$("<option>Numero "+i+"</option>");
    opcion.attr("value",i);
    opcion.css("font-size",25);
    opcion.css("background-color","green");
    select_nuevo.append(opcion);
}
$("body").append(select_nuevo);
```

- Podemos insertar un hermano delante o detrás de un elemento simplemente con las funciones after y before.

```
$("img").after("Texto despues de la imagen");
$("img").before("Texto antes de la imagen");
```