ARQUITECTURA DE INFORMACIÓN EN WEB Y LABORATORIO.

PRÁCTICA II

GARAY HERNANDEZ MAURICIO DE

PRIMAVERA 2021

ÍNDICE

Introducción

Reto JQuery y JQuery UI

¿Qué es?

Según la propia página de JQuery, esta es una herramienta (biblioteca de Javascript) rápida, ligera, y llena de características. Se caracteriza por hacer la navegación de elementos HTML de manera muy sencilla, permitiéndonos reaccionar a eventos de estos, modificar y acceder valores, animarlos y más mucho más fácil. Encima de esto, tiene un manejo de Ajax muy sencillo y fácil de usar en comparación con Javascript puro (JQuery, 2021).

Contenido

Objetivo de la práctica

El objetivo de este reto es poder demostrar la utilidad, casos de uso y funcionamiento de JQuery. Se explicaran paso a paso algunas herramientas con las que cuenta esta biblioteca, así como el código para su implementación y ejemplos de uso.

Ventajas y Desventajas

Vanilla JS tiene muchas ventajas, pero la verdad es que es demasiado costoso en tiempo, en especial cuando consideras todas las herramientas que existen en el mercado para facilitar el desarrollo de una aplicación web. Sin embargo, sigue existiendo el hecho de que aprender un framework entero como ReactJS puede llegar a consumir demasiado tiempo. Es por eso que solamente utilizar la biblioteca de JQuery puede ser una gran solución para desarrollar el frontend de muchas páginas. La gran ventaja es que, con base en un estudio de mercado realizado por W3techs, 77.7% de las plataformas web utilizan la biblioteca de JQuery (W3techs, 2021). Si alguna herramienta tiene una gran adaptabilidad y es conocida, es esta biblioteca. Otras ventajas que tiene es la gran facilidad que te da en comparación con JS puro para manejo de eventos (como clics de botones, *blur*, pasos del cursor, entre otros) y de elementos de la página web. Creemos que utilizar esta herramienta ahorraría mucho tiempo en comparación a vanilla JS, porque no es difícil de utilizar, simplifica muchas funciones y tiene gran adaptación.

Las desventajas es que no presenta tanta ayuda y abstracción como ReactJS u otro framework, no optimiza tanto como otros frameworks (como el mencionado al inicio del párrafo), no es tan eficiente en manejo de muchas solicitudes como estos, entre otros. En general, creo que las ventajas que tiene JQuery pesan más que sus defectos; y por esto siento que es una grandiosa herramienta.

CDN

Un CDN o Cloud Delivery Network es un grupo de servidores que trabajan en conjunto para poder entregar servicios a varias aplicaciones de internet (Cloudflare, s.f). Cuando nosotros importamos algún *script* o *stylesheet* de internet (ya sea bootstrap o JQuery), el link que pegamos, si es que no descargamos todos los archivos, es comúnmente conocido como CDN. Para utilizar JQuery, se necesita incluir en el *head* del documento HTML el CDN visualizado en la siguiente figura 1:

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js" ></script>
```

Para JQuery UI, de igual forma en la misma posición que JQuery, pero debajo del CDN anterior, se incluye de esta manera:

```
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
```

Algunas funciones explicadas

JQuery:

$(document).ready(function(){})

Esta función nos permite ejecutar código en el instante que termina de cargar/prepararse un documento. Puede ser muy útil si te acabas de mover a un nuevo HTML que tiene su propio script de JS y deseas inmediatamente cargar cierto contenido en este documento utilizando código. Ejemplo: yo deseo abrir https://mirevista/articulo?id=3, y quiero que utilizando JS, inmediatamente se cargue con los datos del artículo que tiene id=3. Utilizaría esta función, y con otras de JQuery cargaría valores del artículo en campos del HTML.

Cómo utilizarlo:

1. Simplemente escribe la sentencia $(document).ready(function(){
   myCallback(); //puedes igual desarrollar aquí tu código.
   });
2. Desarrolla lo que deseas dentro del callback.

Código:

Utilizando el ejemplo del artículo, primero obtendríamos los datos de éste.

```
$(document).ready(function()
{
    var id=getUrlVars()["id"];
    var articulo=articulo1;
    arregloArticulos.forEach(function(articuloActual)
    {
        if(articuloActual.id==id)
        {
            articulo=articuloActual;
        }
    })
    var Autor=articulo.Autor;
    var Titulo=articulo.Titulo;
    var Categoria=articulo.Categoria;
    var Texto=articulo.Texto;
    var Dia=articulo.Dia;
    var Mes=articulo.Mes;
    var Anio=articulo.Anio;
    var datosA=""+Autor+Titulo+Categoria+Texto+Dia+Mes+Anio;
    console.log(datosA);
```

Resultados:
Lo primero que sucede al entrar a articulo.html:

```
Kirya HernandezOzimandias: El F.C Barcelona se convierte articulos.js:125
en rey de reyesInternacionalEn la tremenda noche del dÃa 17 de abril del
2021, el Futbol Club <br> Barcelona dio un golpe pesado en contra del
Athletic Club de Bilbao para <br> llevarse asi y despues de tres anos de
sequia, la ansiada Copa del Rey. <br> Esta edicion copera nos ha dejado
muchos matices y sabores de boca. Desde <br> un milagroso Alcoyano, hasta
la decepcion del Atletico de Madrid. Pero sin <br> duda alguna, las
miticas hazanas que creo el Futbol Club Barcelona para <br> llevarselas
dejan mucho que hablar. <br> Tras dos remontadas (Granada y Sevilla, goles
```

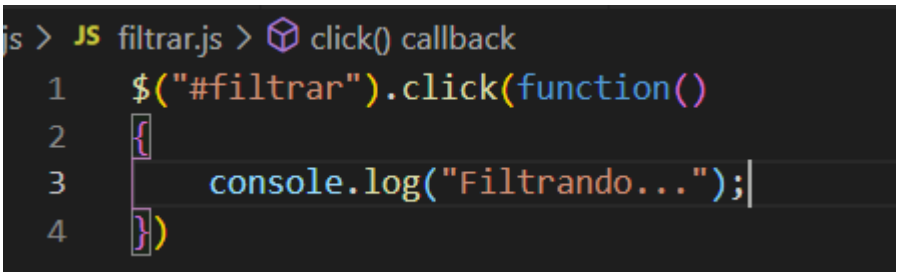
$.getScript(scriptRoute)
Esta función te permite inicializar un archivo JS desde otro de la misma extensión. Por ejemplo, si tienes una *Single Page Application* y deseas moverte de filtrar a modificar, main.js podría hacer $.getScript(modificar.js). Esto permitiría que utilicemos todas las funciones del segundo script.
Cómo utilizarlo:

1. Es imperativo tener un servidor HTTPS dado de alta. Si no, no funcionará esta función (ni Ajax) porque utilizan este protocolo para mandar a llamar otras funciones.
2. Desde el script origen, cuando se mande la señal para que nosotros podamos utilizar el segundo script, mandar a llamar $.getScript(segundoarchivo.js).

3. ¡Listo! Ahora podremos utilizar todas las funciones del script invocado.

Código:

Supongamos que queremos utilizar las funciones de filtrar.js si el usuario presiona un botón con id=abrirModalFiltro. Posteriormente, si el usuario presiona uno con id=filtrar, ya se hace la búsqueda dentro del script filtrar.js.



Resultados:

```
Filtrando...
```

$(element).click(function(){});

Nos permite reaccionar, a través de un callback, a cuando un elemento especificado (por clase o id) sufre un evento de click.

Cómo utilizarlo:

1. Especificar el elemento entre comillas. Utilizar # para indicar id y . para indicar clase. Ejemplo: $(“.miClase”).click(function(){console.log(“hola”)}); o $(“#miId”).click(function(){console.log(“hola”)});. Si no se utiliza ningún prefijo, se está hablando de una etiqueta, de esta forma: $(“div”).click(function(){console.log(“Por qué harías esto?”)});
2. Simplemente escribe la sentencia $(element).click(function(){ myCallback(); //puedes igual desarrollar aquí tu código. });
3. Desarrolla lo que deseas dentro del callback.

Código:

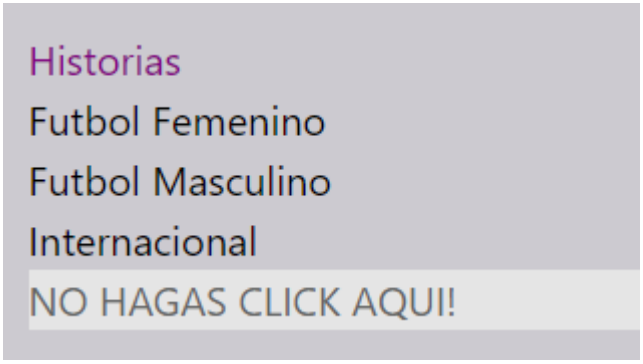
Quiero cambiar el tema de mi página al hacer click en un botón.

```
$("#botonTema").click(function()
{
    if(temaOscuro)
    {
    $(".letrasTema").animate({
        color:"#fff"
    },1000)
    $(".background").animate({
        backgroundColor:"#000"
    },1000)
    temaOscuro=false;
    }
    else
    {
    $(".letrasTema").animate({
        color:"#000"
    },1000)
    $(".background").animate({
        backgroundColor:"#CCCAD0"
    },1000)
    temaOscuro=true;
    }
});
```

Resultados:

Antes:

Durante:



Después:

$(element).val();

Esta función nos permite obtener el valor de algún elemento mencionado. Por ejemplo, lo podemos utilizar para validar el valor de algún input.

Cómo utilizarlo:

1. Aquí no hay callback, la función simplemente devuelve el valor del elemento.
2. Recibe el valor que devuelve y dale el tratado necesario.

Código:

Vamos a ver el valor que hay en un campo de email cuando se presiona el botón de enviar.

```
$("#subirMail").click(function()
{
  var valorMail=$("#Email1").val();
  console.log(valorMail);
})
```

Resultados:

$(element).blur(function(){});

Esta función nos permite reaccionar al evento cuando, después de haber presionado un elemento (por ejemplo, un *input type text*) nos salimos de este, ya sea por presionar fuera o utilizar el tabulador.

Cómo utilizarlo:

1. Simplemente escribe la sentencia $(element).blur(function(){
   myCallback(); //puedes igual desarrollar aquí tu código.
   });
2. Desarrolla lo que deseas dentro del callback.

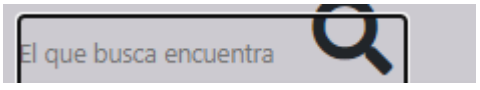
Código:

En la página que he mostrado de ejemplo, aún no se desarrolla todo el módulo de búsqueda. Por eso, cuando hacen blur, reemplazamos el texto de la barra de búsqueda con un "Disponible próximamente:)". Utilizo la palabra reservada this para que solamente se cambie el valor de esa barra de búsqueda y no de todas en el documento.

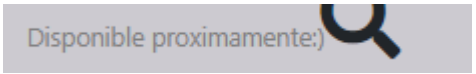
```
$(".searchBar").blur(function()
{
  $(this).val("Disponible proximamente:)");
  $(this).attr("style","color:gray");
});
```

Resultados:

Antes del blur:



Después del blur:



$(element).mouseover(function(){});

Esta función nos permite reaccionar al evento cuando el cursor pasa por encima de un elemento especificado.

Cómo utilizarlo:

1. Asegurarse que el elemento no esté debajo de otros o con un índice z que haga imposible la captura del evento.
2. Simplemente escribe la sentencia $(element).mouseover(function(){
   myCallback(); //puedes igual desarrollar aquí tu código.
   });
3. Desarrolla lo que deseas dentro del callback.

Código:

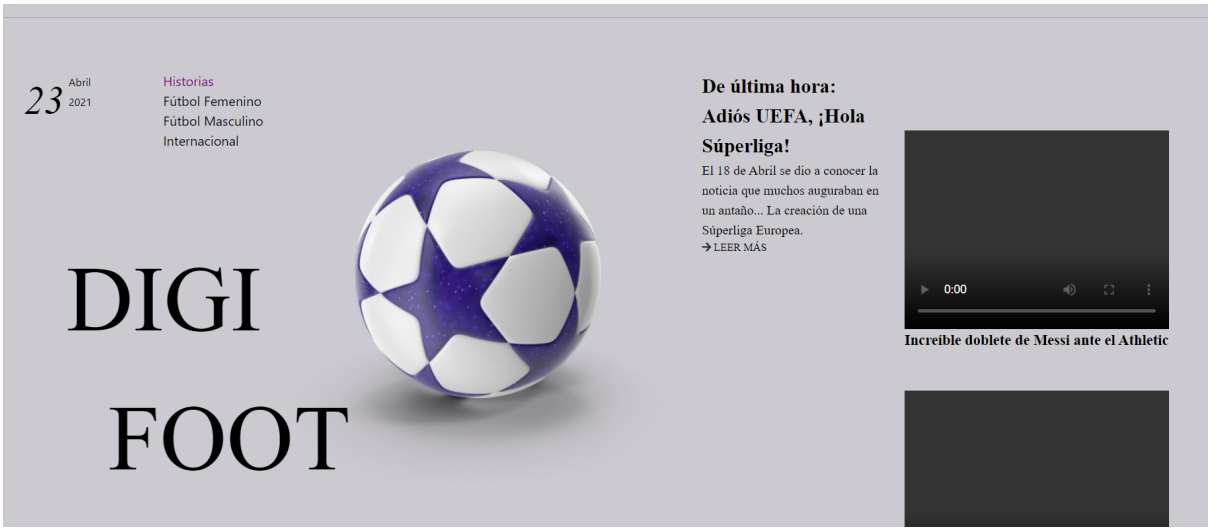
Tengo un balón en la página principal, y quiero que se mueva de un lado a otro cada que paso el cursor por encima. La imagen como tal es un *background-image*, por eso necesito reaccionar a cuando el mouse pasa por encima del div que lo contiene.

```
$('.columnaBalon').mouseover(function(){

    console.log("izq arriba");
    $(".imagenBalon").animate({
      left: '+='+balonMovimiento+"rem",
      top: '+='+balonMovimiento+"rem"
    },500);
    balonMovimiento*=-1;

});
```

Resultados:

El movimiento es muy pequeño, pero se puede ver cómo está más alejado de “DIGIFOOT” en una imagen a comparación de la otra.

Antes:



Después:

JQuery UI

JQuery UI es una herramienta que nos permite incluir una gran cantidad de *widgets*, efectos y temas basados en JQuery. Al utilizarlo, nos permite hacer aplicaciones más interactivas y con más estética de manera sencilla (JQuery UI, 2021).

JQuery UI Tooltip

Esta herramienta sumamente sencilla nos permite ver las descripciones de un elemento de una manera más elegante.

Cómo utilizarlo:

1. En el HTML, simplemente agrega una propiedad *title* que tenga de valor la descripción que seas del tooltip.
2. En el JS, indica $(elemento).tooltip(). Esto permitirá que se vea elegante el tooltip de ese elemento.

Código:

Solamente haremos el tooltip para la barra de búsqueda, pero para el botón de menú no, así podremos apreciar la diferencia.

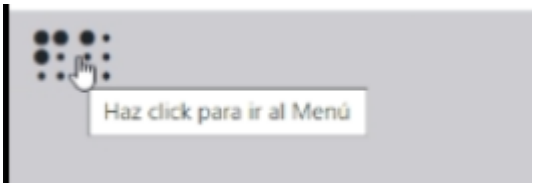
HTML:

```
<input type="text" name="search" class="form-control-plaintext input-lg letrasTema searchBar" title="Ingresa el título del artículo que quieres leer" placeholder="El que busca encuentra">
```

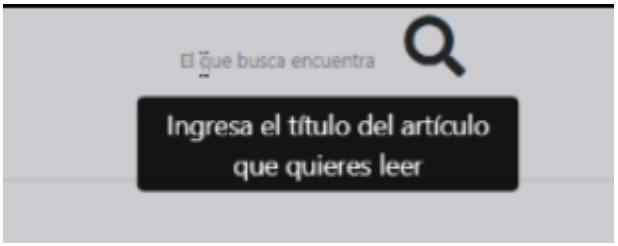
```
$(document).ready(function()
{
  main();
  $(".searchBar").tooltip();
```

Resultados:

Sin tooltip:



Con tooltip:



JQuery UI DatePicker

Esta herramienta es súmamente útil. No solamente porque permite tener una interfaz personalizable para escoger fechas (donde puedes nombrar, a través de notación JSON, los meses y días como quieras); si no que también porque permite mostrarle un formato distinto al usuario que el que vas a recibir. Ejemplo: mySQL trata las fechas como AAAA-MM-DD, pero la mayoría de los ciudadanos mexicanos las escriben como DD/MM/AAAA. Datepicker nos permite mostrarle al usuario un formato, pero interpretarlo en otro.

Cómo utilizarlo:

1. Crear dos inputs: uno visible y uno hidden. El valor del visible tendrá el formato que le parezca más cómodo al usuario. El hidden el que el programador prefiera.
2. Agregar un id distinto a cada uno de estos inputs.
3. Inicializar estos elementos como datepicker dentro de JS.
4. Aquí, podemos configurar el formato de cada uno, así como los nombres de cada mes y día.

Código:

En este HTML, #fecha_i tendrá el formato del usuario, mientras que fecha_inicial el que yo deseo para la base de datos.

```
<div>
    <input type="text" id="fecha_i" name="fecha_i" class="form-control" style="width:5%" placeholder="dd/mm/aaaa">
    <input type="hidden" id="fecha_inicial" name="fecha_inicial" class="form-control"  value="">
</div>
```

Aquí, le doy formato a los meses y días. Además, le doy formato alterno a #fecha_i y le indico en qué campo aplicará este.

```
$('#fecha_i').datepicker({
    dateFormat: "dd/mm/yy",
    altField: "#fecha_inicial",
    altFormat:"yy-mm-dd",
    monthNames: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'],
    dayNamesShort: ['Dom', 'Lun', 'Mar', 'Mié', 'Juv', 'Vie', 'Sáb'],
    dayNamesMin: ['Do', 'Lu', 'Ma', 'Mi', 'Ju', 'Vi', 'Sá'],
});
```

Resultados:

JQuery UI Animate

Esta herramienta nos permite animar alguna transición de un elemento, ya sea al transformar posición, dimensiones, color, fondo. Además, nos permite especificar, en milisegundos, cuánto tiempo queremos que dure esta animación.

Cómo utilizarlo:

1. Reacciona al evento que quieres que desate la animación, por ejemplo: el click de un botón.
2. Llama $(elemento).animate(function(){myCallback()},milisegundos);
3. Desarrolla el callback.

Código:

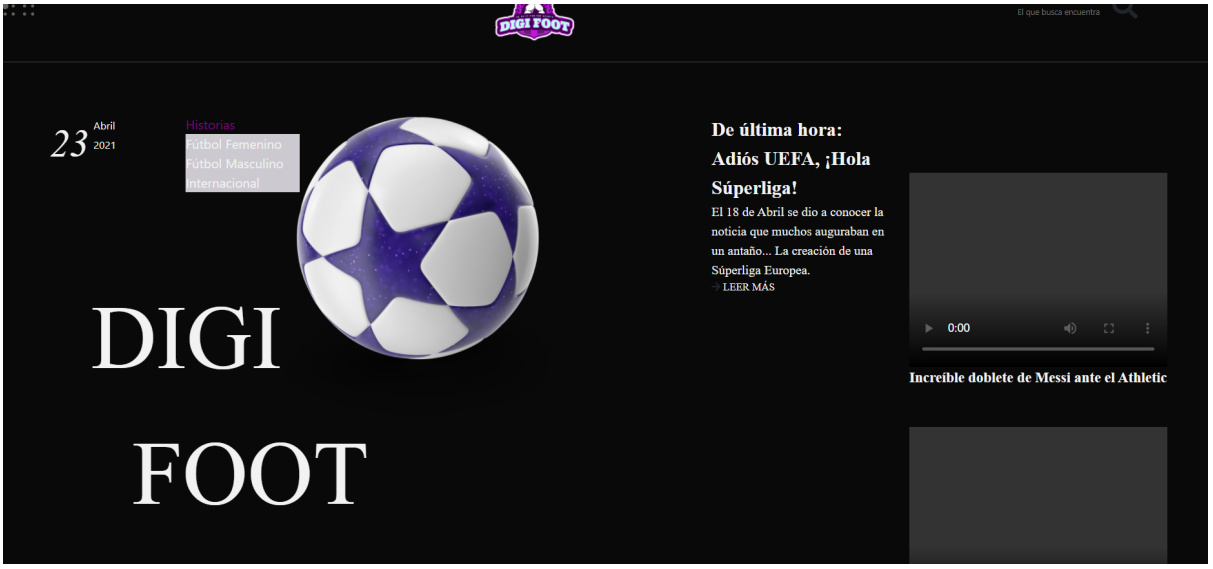
En este ejemplo, quiero que cuando el usuario presione un botón, inicie una transición de un tema claro a uno oscuro para toda la página. Además, quiero que, alternándose, el balón se mueva una distancia aleatoria de izquierda a derecha para simular un partido.

```
$("#botonTema").click(function()
{
  var factor1=getRandomInt(8);
  factor1++;
  var factor2=getRandomInt(8);
  factor2++;
  if(temaOscuro)
  {
    $(".imagenBalon").animate({
      top: '-=' + $(".imagenBalon").height() / factor1,
      left: '-=' + $(".imagenBalon").width() / factor1,
    },2000);
    $(".letrasTema").animate({
      color:"#fff"
    },1000)
    $(".background").animate({
      backgroundColor:"#000"
    },1000)
    temaOscuro=false;
  }
  else
  {
    $(".imagenBalon").animate({
      top: '+=' + $(".imagenBalon").height() / factor2,
      left: '+=' + $(".imagenBalon").width() / factor2,
    },2000);
    $(".letrasTema").animate({
      color:"#000"
    },1000)
    $(".background").animate({
      backgroundColor:"#CCCAD0"
    },1000)
    temaOscuro=true;
  }
  console.log(temaOscuro);
});
```

Resultados:
A pesar de que no se puede ver el video en este documento, adjunto imágenes.
Antes:

EQUIPO 2

Durante:



Después:

Dónde encontrar más sobre JQuery

Además de la enorme cantidad de recursos en línea que hay por la gran adaptación de estas herramientas, las documentaciones oficiales están muy detalladas y completas. Estas se encuentran en las referencias 1 para JQuery y 3 para JQuery UI, que están al final del documento actual. La documentación oficial de JQuery UI tiene una gran cantidad de ejemplos interactivos para comprender más fácilmente su funcionamiento.

Conclusiones

En conclusión, hemos demostrado la utilidad que tienen varias herramientas de JQuery y JQuery UI, así como su simplicidad e implementación. En este reto, se han podido ver ejemplos de JQuery, y se ha logrado demostrar que aprender esta biblioteca es una gran opción, por la gran cantidad de versatilidad que le agrega a nuestra aplicación web y su gran adaptación en el ambiente de desarrollo web.

Referencias

1.- *What is JQuery?* (2021.). JQuery. Recuperado el 22 de Abril de 2021, de https://jquery.com/

2.- *Usage statistics of Javascript libraries for websites*. (2021). Recupero el 22 de Abril de 2021, de https://w3techs.com/technologies/overview/javascript_library

3.- *JQuery UI*. (2021). Recuperado el 22 de Abril de 2021, de https://JQuery UI.com/

4.- *What is a CDN? How do CDNs work?* (s.f). Recuperado el 22 de Abril de 2021, de https://www.cloudflare.com/learning/cdn/what-is-a-cdn/