

HINTS

JQUERY

Es una biblioteca liviana de JavaScript, cuyo objetivo es “lograr más cosas escribiendo menos código”, facilitando el uso de este lenguaje en un sitio web. ¿Cómo lo hace? jQuery toma tareas comunes de JavaScript que requieren muchas líneas de código, y las envuelve en métodos que se pueden llamar con una sola línea para llevarlas a cabo.

Para utilizar esta herramienta, primero debemos incorporar la librería JQuery a un archivo HTML. Empezaremos creando un archivo HTML en nuestro IDE Visual Studio Code. De momento, nuestro HTML en blanco debería semejarse a lo siguiente:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-
6 scale=1.0">
7     <title>Document</title>
8   </head>
9   <body>
10  </body>
11 </html>
```

Teniendo nuestro HTML en blanco, podemos apreciar tres elementos fundamentales de cualquier DOM. En nuestra primera línea del código, está la definición de qué es un documento de tipo HTML. Además, cada elemento contiene etiquetas **head** y **body**. Adicionalmente, éstos tienen sus propios atributos, y aunque de momento no tenemos alguno, cada elemento también puede tener un método asociado.

Cada uno de estos atributos y elementos, nos permiten ver todas las ramas dentro del árbol que se genera en el modelo de objetos del documento, o DOM.

Para agregar jQuery a un documento HTML, hay más de una forma de hacerlo. Por un lado, se puede descargar la biblioteca desde jquery.com, o podemos incluirla desde CDN como Google.

Simplificando este proceso, incluiremos jQuery con el uso de un CDN. ¿Cómo lo hacemos? El único paso es incluir las siguientes líneas de código en las etiquetas head de nuestro HTML:

```
1 <head>
2   <meta charset="UTF-8">
3   <meta name="viewport" content="width=device-width, initial-scale=1.0">
4   <title>Ejercicio JQuery</title>
5   <script
6 src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></sc
7 ript>
8 </head>
```

Ahora que hemos incluido JQuery a nuestro HTML, estamos en condiciones de hablar sobre la sintaxis de esta librería, la cual está hecha para seleccionar elementos HTML desde el DOM, y realizar alguna acción sobre ellos.

Un ejemplo sencillo sobre esta sintaxis es:

```
1 $(selector).action()
```

Un signo **\$** se usa para definir que queremos acceder a jQuery. Este es seguido por un **selector**, que está puesto entre paréntesis, dentro del cual llamamos o seleccionamos el elemento del HTML que queremos usar. Por último, se establece una **acción** JQuery, que se realizará sobre él o los elementos. Tomando en cuenta que el método **hide()** se encarga de ocultarlos, considere los siguientes ejemplos:

```
1 $(this).hide() // oculta el elemento actual
2 $("h1").hide() // oculta todos los elementos <h1>
3 $(".hola").hide() // oculta todos los elementos class="hola".
4 $("#chao").hide() // oculta el elementos con el id="chao".
```

Antes de poner en práctica los ejemplos que acabamos de analizar de manera conceptual, primero debemos considerar el evento **Document Ready** (documento listo).

Una página no se puede manipular de forma segura hasta que el documento esté "listo" o **ready**. jQuery detecta este estado de preparación mediante el método **\$(document).ready()**, que solo se ejecutará una vez que el DOM de la página esté listo para que se ejecute el código JavaScript. Este método se puede llamar de dos formas:

```
1 $(document).ready(function () {
2 // Aquí van los métodos o acciones jQuery
3 });
4 $(function () {
5 // Aquí van los métodos o acciones jQuery
6 });
```

Debemos insertar nuestro código con jQuery dentro del método **Document Ready**, y como cualquier otro código JavaScript, se debe colocar dentro de las etiquetas script. Por ejemplo: supongamos que dentro del cuerpo de nuestro HTML tenemos un elemento **<h1>** con un texto. Dentro de nuestras etiquetas script, vamos a incluir el código con jQuery, llamando al método **Document Ready** dentro del cual establecemos que queremos esconder el elemento **<h1>**.

```
1 <body>
2   <h1>Esto es un elemento dentro de nuestro html</h1>
3 </body>
4 <script>
5   $(function () {
6     $("h1").hide();
7   });
8 </script>
9 </html>
```

Como recomendación, es importante que puedas ver en tu navegador el efecto de este método, y así practicar usando jQuery en tus desarrollos.

POLYFILLS

Ya que conocemos algunos aspectos de JavaScript, podemos considerar un tema interesante llamado polyfills. Éstos son fragmentos de código (generalmente JavaScript) que se utilizan para implementar una funcionalidad moderna en navegadores más antiguos que no lo admiten de forma nativa.

Por ejemplo, un polyfill podría usarse para imitar la funcionalidad de una sombra de texto en Internet Explorer 7, usando filtros de Internet Explorer patentados, o imitar unidades rem o consultas de medios usando JavaScript para ajustar dinámicamente el estilo según corresponda.

Actualmente existen API's o herramientas externas, las cuales permiten realizar las funcionalidades de los polyfills de una manera mucho más eficiente. Sin embargo, es importante conocer estos fragmentos en caso de tener que trabajar con ellos.

ATAJOS CON JQUERY

Con jQuery podemos aplicar las mismas reglas de estilo a múltiples elementos de nuestro HTML, utilizando el método **addClass()**. Para ello, primero debemos definir en nuestro selector los distintos elementos que deseamos manipular, separados por una coma.

Para ver este método en práctica, vamos a usar el siguiente código:

```
1 <body>
2   <h2>Primer elemento</h2>
3   <h5>Segundo elemento</h5>
4   <p>Tercer elemento</p>
5   <center><button>Agregar clase</button></center>
6 </body>
```

De estas líneas de código, aplicaremos los siguientes cambios de estilo a los elementos: `<h2>`, `<h3>` y `<p>`.

```
1 .clasePrueba {
2     font-weight: bold;
3     font-size: xx-large;
4     font-family: monospace;
5     color: green;
6     text-align: center;
7 }
```

Para hacer esto, vamos a llamar al método `addClass()`, pasando por parámetro el nombre de la clase que contiene todos nuestros cambios de estilos. De esta forma, cuando hacemos **CLIC** en el botón de nuestro HTML, le asignaremos a los elementos `<h2>`, `<h3>` y `<p>` la clase `clasePrueba`.

```
1 $(function () {
2     $("button").click(function () {
3         //Usamos el método addClass para asignarle una clase a una
4 serie de elementos.
5         $("h2, h5, p").addClass("clasePrueba");
6     });
7 });
```

En nuestro navegador, el código se ve de la siguiente forma. Cuando hacemos clic en el botón, veremos que le asignamos a los tres elementos de nuestro HTML la misma clase que contiene todos los cambios de estilo.

Primer elemento

Segundo elemento

Tercer elemento

Agregar clase

Los cambios efectuados mediante la clase `clasePrueba`, son los siguientes:

Primer elemento

Segundo elemento

Tercer elemento

Agregar clase

Hasta el momento, ya nos hemos familiarizado con cuatro métodos que podemos utilizar como manejadores de eventos, pero la librería jQuery nos permite usar aún más en distintos contextos. Ahora, analizaremos tres métodos más, y utilizaremos el siguiente código en nuestro HTML para ver como los manejadores de eventos afectan este código:

```
1 <center>
2   <div id="d1">Primer elemento de referencia.</div>
3   <div id="d2">Segundo elemento de referencia.</div>
4 </center>
```

DBLCLICK()

Este método se ejecuta cuando se hacen dos clics, o clic doble sobre un elemento HTML. Podemos probarlo con el siguiente código:

```
1 $("#d1").dblclick(function() {
2     $(this).css({
3         "fontSize": "larger",
4         "color": "orange"
5     });
6 });
```

```

6     })
7     $("#d2").dblclick(function() {
8         $(this).css("background-color", "yellow");
9     })
  
```

El resultado de este código en nuestro navegador, es el siguiente:

Primer elemento de referencia.
 Segundo elemento de referencia.



Primer elemento de referencia.
 Segundo elemento de referencia.

MOUSEDOWN() & MOUSEUP()

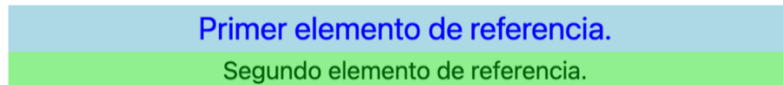
El primer método se ejecuta cuando se presiona el botón izquierdo, derecho o del medio del mouse, mientras el cursor esté sobre el elemento HTML. En cambio, el segundo método se ejecuta cuando ***se deja de presionar*** el botón izquierdo, derecho o del medio del mouse, mientras el cursor esté sobre el elemento HTML. Veamos esto con el siguiente código:

```

1  $("#d1").dblclick(function() {
2      $(this).css({
3          "fontSize": "larger",
4          "color": "orange"
5      });
6  })
7  $("#d2").dblclick(function() {
8      $(this).css("background-color", "yellow");
9  })
  
```

En nuestro navegador podemos ver el efecto de este manejador al cambiar nuestra vista, de la siguiente forma:

Primer elemento de referencia.
Segundo elemento de referencia.



FOCUS()

JavaScript dispone de mecanismos para posicionarnos en un elemento de la página. Esto, por ejemplo, nos puede ser útil para posicionarnos en un campo concreto de un formulario, o por validaciones que nos hagan ir a otro campo.

Pero no solo nos sirve para hacer foco en campos de formulario. También podemos hacerlo en otros elementos seleccionables de una página, tales como un enlace.

El método que nos sirve para hacer foco es: **.focus()**.

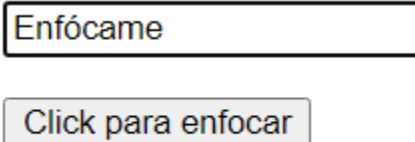
En nuestro JavaScript realizaremos la siguiente función.

```
1 function enfocar() {  
2   document.getElementById("ElementoEnfocado").focus();  
3 }
```

En el HTML se asigna el método **pruebaFocus()** al evento **onclick**

```
1 <input type="text" id="ElementoEnfocado" value="Enfócame">  
2 <p></p>  
3 <button type="button" onclick="enfocar()">Click para enfocar</button>
```

Finalmente, el resultado al presionar el botón será:



LOAD() Y SCROLL()

El método `.load()` se envía a un elemento cuando éste, y todos los subelementos, se han cargado por completo. Este método, junto con `.unload()` y `.error()`, están obsoletos desde jQuery 1.8. Esto significa que no se usan, pero deben reemplazarse por otros como el método `.on()`, pasándole la palabra `"load"` como su primer argumento, dado que primero recibe el tipo de manejador de evento que queremos usar, y luego una función por ejecutar.

```
1 $("img").on("load", function () {  
2     alert("hola.");  
3 });
```

Esto hace que aparezca un mensaje de alerta una vez que se carga la imagen de un perrito, la cual vamos a usar como referencia:

```
1 
```

En nuestro código JavaScript primero veremos un alert:

127.0.0.1:5500 says
hola.

OK

Y luego vemos la imagen de nuestro perrito:



Otro detector de eventos realmente útil es `.scroll()`. Para poner este selector en práctica, vamos a sacar la imagen de nuestro HTML, y colocaremos el siguiente div.

```
1  <div>
2      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus
3  ullamcorper erat diam, vel pretium ex porta
4      id. Integer et placerat velit, sed tincidunt est. Nulla molestie,
5  arcu sit amet mattis accumsan, nibh sem
6      hendrerit dui, non blandit massa erat sed risus. In vel libero et
7  justo aliquet ultricies. Proin aliquam erat at
8      enim ullamcorper consequat. Sed at pharetra ante. Vestibulum id
9  neque ligula. Donec quis vestibulum turpis, vel
10     auctor mi. Sed tincidunt arcu ante, quis ornare tellus imperdiet
11  egestas. Vestibulum in ipsum purus. Praesent
12     vestibulum rhoncus faucibus. Donec convallis ornare nisl vel
13  interdum. Maecenas ac dolor a magna consectetur
14     pellentesque et sed libero. Donec dignissim ligula ut est euismod
15  pretium. Pellentesque laoreet a mi id euismod.
16     In libero urna, placerat at consequat ut, vehicula at libero.
17
18     Curabitur commodo et lacus sagittis vulputate. Duis interdum
19  egestas augue, eu pharetra orci feugiat a.
20     Curabitur volutpat commodo mauris, sed pharetra dolor sagittis
21  sed. Ut at pulvinar lorem, vel porta elit.
22     Curabitur faucibus gravida nunc vel congue. Nulla vel urna dui.
23  Curabitur sollicitudin dui ac ante imperdiet,
24     nec pharetra magna bibendum. Nullam auctor erat mauris, congue
25  aliquet velit venenatis ut. Maecenas interdum
26     urna ut aliquam blandit. Nunc sit amet orci in orci tincidunt
27  placerat quis molestie purus. Vestibulum eleifend
28     eleifend condimentum. Ut eget congue eros.
29
30     Class aptent taciti sociosqu ad litora torquent per conubia
31  nostra, per inceptos himenaeos. Aenean a libero eu
32     tortor luctus ullamcorper a non sem. Praesent gravida diam nec
33  pharetra facilisis. Praesent posuere rhoncus
34     mauris, nec semper metus aliquet vestibulum. Aenean auctor purus
35  mauris, non bibendum neque convallis vitae.
36     Interdum et malesuada fames ac ante ipsum primis in faucibus.
37  Aliquam at tortor ut libero sodales mollis.
38     Curabitur non condimentum ante. Etiam id pulvinar nisl.
39
40     Donec id commodo neque. Nam nibh velit, ullamcorper eu magna eu,
41  ornare dignissim tortor. Vestibulum aliquam
42     diam nibh, mattis sodales orci semper et. Nullam porta metus id
43  ligula tempus eleifend. In eget ullamcorper leo,
```

```
44     sit amet blandit ante. Nulla cursus urna porta consectetur
45 fermentum. Maecenas eu nisi a sapien suscipit dictum.
46     Aenean fermentum tincidunt ligula, eget ultrices leo tincidunt
47 nec. In mollis, metus eget pulvinar congue, orci
48     sapien sagittis nisi, quis posuere magna libero sollicitudin
49 velit. Etiam accumsan tortor eu massa commodo
50     aliquet. Vestibulum posuere gravida arcu, quis vestibulum diam
51 mollis at. Phasellus sed mauris diam. Vivamus
52     vitae urna ornare, luctus odio non, auctor ligula. Nunc fermentum
53 fringilla nunc, nec commodo arcu molestie non.
54
55     Nulla pulvinar nibh justo, ac sollicitudin turpis eleifend et.
56 Mauris non orci at purus euismod tempor. Mauris
57     congue rutrum dolor, vel blandit arcu dictum vel. Nullam ac rutrum
58 nibh. Vestibulum mattis nibh nisl, vel
59     convallis enim pulvinar ac. Nulla pharetra fermentum lacus, eget
60 iaculis odio luctus et. Sed a diam eget felis
61     interdum mollis. Fusce sit amet semper est, ac venenatis mi. Proin
62 feugiat blandit neque.
63     </div>
```

Luego, le aplicaremos algunas sentencias de estilo para demostrar el desplazamiento.

```
1     <style>
2     div {
3         border: 1px solid black;
4         width: 500px;
5         height: 100px;
6         overflow: scroll;
7     }
8     </style>
```

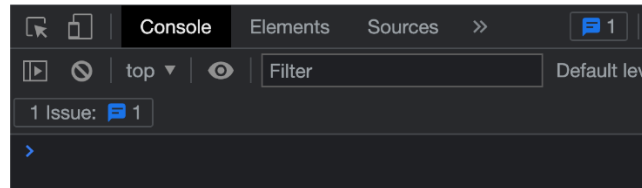
Ahora, vamos a usar el manejador de eventos `.scroll()` para mostrar por la consola las veces que el código detecta que nos hemos desplazado por el contenido de la página.

```
1 $(document).ready(function () {
2     $("div").scroll(function () {
3         console.log("se desplazó");
4     });
5 });
```

Así se debería ver nuestra página:

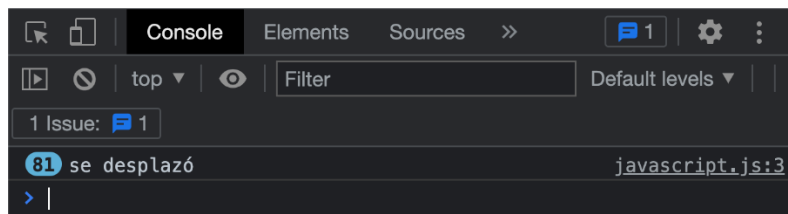


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus ullamcorper erat diam, vel pretium ex porta id. Integer et placerat velit, sed tincidunt est. Nulla molestie, arcu sit amet mattis accumsan, nibh sem hendrerit dui, non blandit massa erat sed risus.



Si ahora desplazamos el cursor en el cuadrado con nuestro texto, veremos en la consola todas las veces que el sitio detectó que lo hicimos.

In vel libero et justo aliquet ultricies. Proin aliquam erat at enim ullamcorper consequat. Sed at pharetra ante. Vestibulum id neque ligula. Donec quis vestibulum turpis, vel auctor mi. Sed tincidunt arcu ante, quis ornare tellus imperdiet egestas. Vestibulum in ipsum



Al saber utilizar estos nuevos manejadores de eventos, podremos incorporar nuevas formas de interactuar con nuestras páginas web utilizando la librería jQuery.