Desarrollo Web Frontend

Edición 2021

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología – Sede Oro Verde

¿Qué es JQuery?

Instalación de JQuery

Primeros pasos

Eventos en JQuery

Funciones callback

¿Qué es JQuery?

Jquery es una bibilioteca liviana de JavaScript que tiene como propósito facilitar el uso de JavaScript en un sitio web → *Escribir menos, hacer más.*

- Toma tareas comunes que requieren muchas líneas de código en JavaScript y las envuelve en métodos que pueden ser utilizados en una sola línea de código.
- Simplifica las acciones de llamadas a AJAX y manipulación de elementos del DOM.
- Si bien se trata de una librería antigua (reemplazada en parte por tecnologías como Vue, Angular o React), aún existen muchos sitios web que actualmente la utilizan.

¿Qué es JQuery? (Continuación...)

La biblioteca JQuery cuenta con las siguientes características:

- Manipulación HTML/DOM
- Manipulación CSS
- Manejadores de eventos HTML
- Efectos y animaciones
- AJAX
- Utilidades

~~¿Qué es JQuery?~~

Instalación de JQuery

Primeros pasos

Eventos en JQuery

Funciones callback

Instalación de JQuery

Podemos añadir la librería de JQuery en nuestro sitio web de la siguiente manera:

- Descargando la librería JQuery desde el sitio oficial https://jquery.com/download/ e importar la librería dentro del **<head></head>** del documento web.

```
<head>
    <script src="jquery-3.5.1.min.js"></script>
</head>
```

Instalación de JQuery (Continuación...)

Podemos añadir la librería de JQuery en nuestro sitio web de la siguiente manera:

- Utilizando una red de entrega de contenidos (**CDN**) de cualquier proveedor. Ej: Google

```
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

~~¿Qué es JQuery?~~

~~Instalación de JQuery~~

Primeros pasos

Eventos en JQuery

Funciones callback

Primeros pasos con JQuery

El evento Document Ready

Este evento se encarga de ejecutar todo el código JQuery una vez que el documento se ha cargado completamente **(isReady).**

Esto permite que podamos incluir código JavaScript antes de las etiquetas **<body>**, en la sección **<head></head>** del documento web.

Por lo tanto es recomendado que nuestros scripts JavaScript comiencen de la siguiente manera:

```
js > JS index.js > ...
1
2   $(document).ready(function(){
3
4     //Metodos JQuery se ejecutan aquí ...
5
6   });
```

Primeros pasos con JQuery (Continuación...)

La sintaxis de JQuery utiliza **selectores** HTML para obtener y realizar ciertas **acciones** sobre dichos elementos**.**

$(selector).accion() ó JQuery(selector).accion()

Donde:

- **$ / JQuery** → referencia o definición a JQuery
- **(selector)** → elemento HTML a seleccionar / buscar.
- **.accion()** → una acción de JQuery para aplicar sobre el / los elementos seleccionados.

Ejemplo:

$('p').hide() → ocultar todos los elementos <p>

$('.test').hide() → ocultar todos los elementos con el atributo class="test"

$('#test').hide() → ocultar el elemento con el atributo id="test"

~~¿Qué es JQuery?~~

~~Instalación de JQuery~~

~~Primeros pasos~~

Eventos en JQuery

Funciones callback

Eventos en JQuery

Prácticamente todos los eventos del DOM tienen su método equivalente en JQuery.

A continuación veremos algunos de los eventos más comúnmente utilizados:

Eventos de mouse	Eventos de teclado	Eventos de formulario	Eventos document
click	keypress	submit	load
dbclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Eventos en JQuery (Continuación...)

Evento click

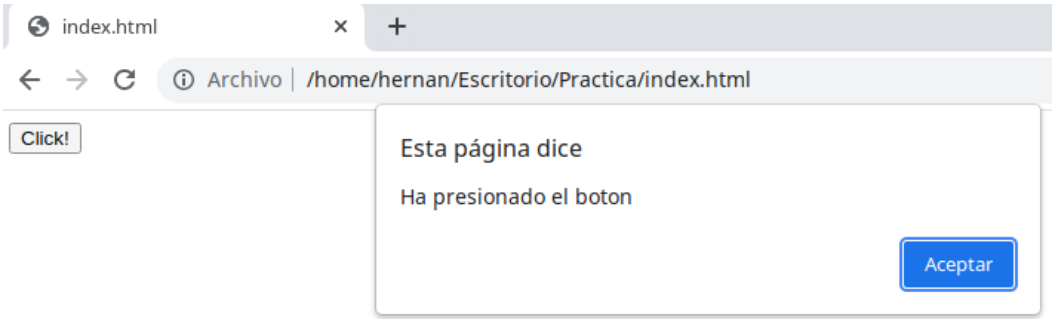
El método **click()** permite asignar un manejador de evento **click** sobre un elemento HTML. La función que se pasa como parámetro se ejecuta cuando se *dispara* dicho evento.

js > JS index.js > ...

```
1
2   $(document).ready(function(){
3
4     $("button").click(function(){
5       alert("Ha presionado el boton");
6     });
7
8   });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <button>Click!</button>
10  </body>
11  </html>
```



Eventos en JQuery (Continuación...)

Evento dblclick

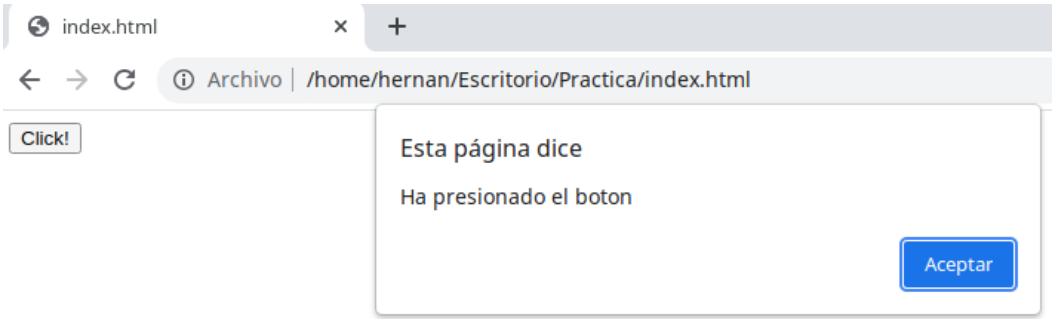
El método **dblclick()** permite asignar un manejador de evento **double click** sobre un elemento HTML. La función que se pasa como parámetro se ejecuta cuando se *dispara* dicho evento.

js > JS index.js > ...

```
1
2   $(document).ready(function(){
3
4     $("button").dblclick(function(){
5       alert("Ha presionado el boton");
6     });
7
8   });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <button>Click!</button>
10  </body>
11  </html>
```



Eventos en JQuery (Continuación...)

Evento mouseenter

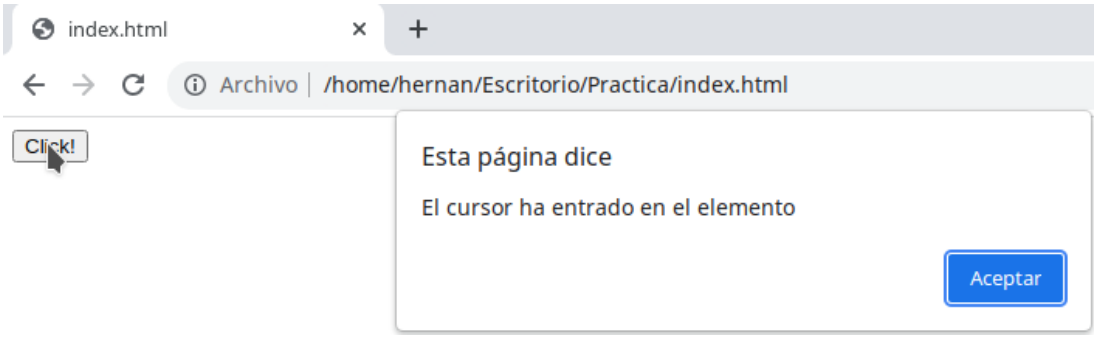
El método **mouseenter()** permite asignar un manejador de evento cuando el cursor del mouse **entra** en un elemento HTML. La función que se pasa como parámetro se ejecuta cuando se *dispara* dicho evento.

js > JS index.js > ...

```
1
2   $(document).ready(function(){
3
4     $("button").mouseenter(function(){
5       alert("El cursor ha entrado en el elemento");
6     });
7
8   });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <button>Click!</button>
10  </body>
11  </html>
```



Eventos en JQuery (Continuación...)

Evento mouseleave

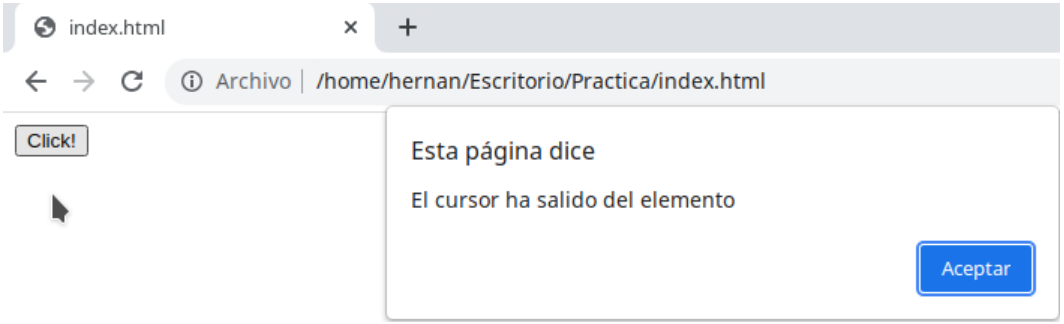
El método **mouseleave()** permite asignar un manejador de evento cuando el cursor del mouse **sale** del elemento HTML. La función que se pasa como parámetro se ejecuta cuando se *dispara* dicho evento.

js > index.js > ...

```
1
2   $(document).ready(function(){
3
4     $("button").mouseleave(function(){
5       alert("El cursor ha salido del elemento");
6     });
7
8   });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <button>Click!</button>
10  </body>
11  </html>
```



Eventos en JQuery (Continuación...)

Evento hover

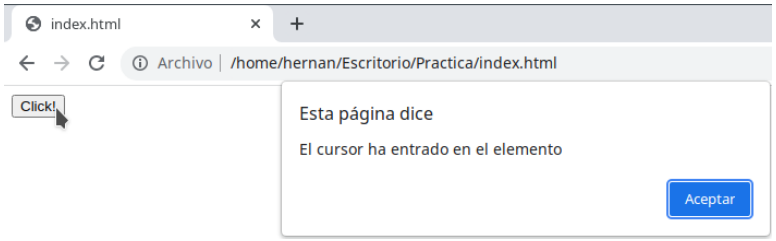
El método **hover()** es una combinación de los métodos **mouseenter()** y **mouseleave()**. Recibe dos funciones por parámetro. La primera se ejecutará cuando el cursor del mouse ingrese en el elemento HTML, y la segunda función cuando el cursos del mouse salga del elemento HTML.

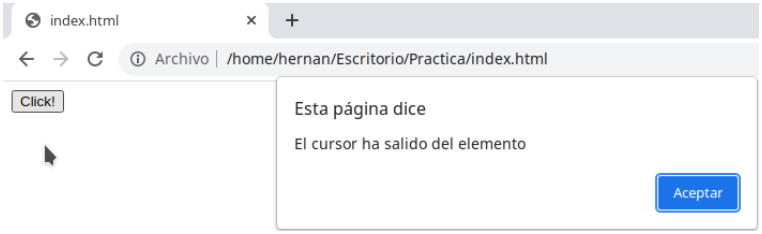
js > index.js > ...

```
2   $(document).ready(function(){
3
4     $("button").hover(
5       function(){
6         alert("El cursor ha entrado en el elemento");
7       },
8       function(){
9         alert("El cursor ha salido del elemento");
10      }
11    );
12
13  });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <button>Click!</button>
10  </body>
11  </html>
```





Eventos en JQuery (Continuación...)

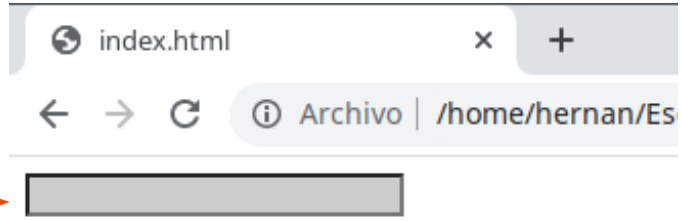
Evento focus

El método **focus()** permite asignar un manejador de evento cuando el elemento HTML entra en foco (seleccionado). La función que se pasa como parámetro se ejecuta cuando se *dispara* dicho evento.

```
js > JS index.js > ...
2   $(document).ready(function(){
3
4     $("input").focus(function(){
5       $(this).css("background-color", "#cccccc");
6     });
7
8   });
```

```
index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <input type="text" name="nombre">
10  </body>
11  </html>
```

El uso de **this** representa el elemento HTML seleccionado



Input en foco cambia de color

Eventos en JQuery (Continuación...)

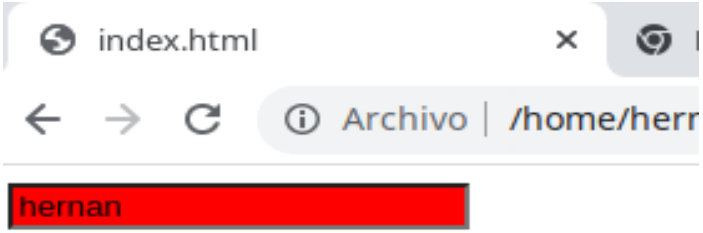
El método on()

Este método permite asignar uno o más manejadores de eventos sobre un elemento HTML. La función que se pasa como parámetro se ejecuta cuando se *dispara* ***alguno*** de los eventos definidos.

```
js > JS index.js > ...
2   $(document).ready(function(){
3
4     $("input").on('keypress paste',function(){
5       $(this).css("background-color", "red");
6     });
7
8   });
```

```
index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <input type="text" name="nombre">
10  </body>
11  </html>
```

Si se escribe sobre el elemento INPUT o se “pega” contenido el mismo cambiará de color



Eventos en JQuery (Continuación...)

Otro ejemplo sobre el método **on()**.

```
js > JS index.js > ...
2   $(document).ready(function(){
3
4     $("p").on(
5       {
6         mouseenter: function(){
7           //acciones cuando el cursor entra en el elemento
8         },
9         mouseleave: function(){
10          //acciones cuando el cursor sale del elemento
11        },
12        click: function(){
13          //acciones cuando se hace click sobre el elemento
14        }
15      }
16    );
17  });
```

```
index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <p>Texto del párrafo</p>
10  </body>
11  </html>
```

A diferencia del ejemplo anterior, en este caso podemos definir diferentes acciones según el tipo de evento que ocurra.

Eventos en JQuery (Continuación...)

Métodos hide() y show()

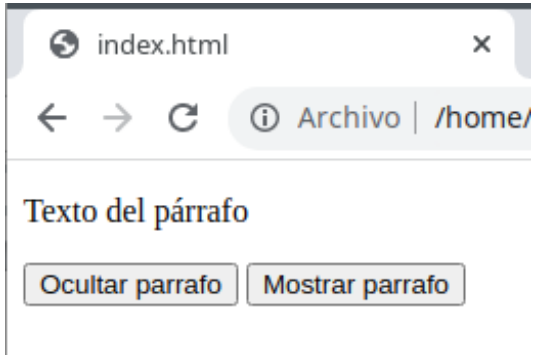
Estos métodos permiten mostrar u ocultar elementos HTML de forma directa.

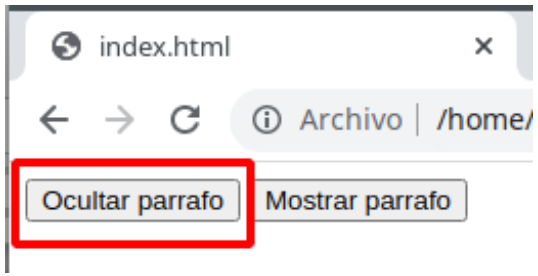
js > JS index.js > ...

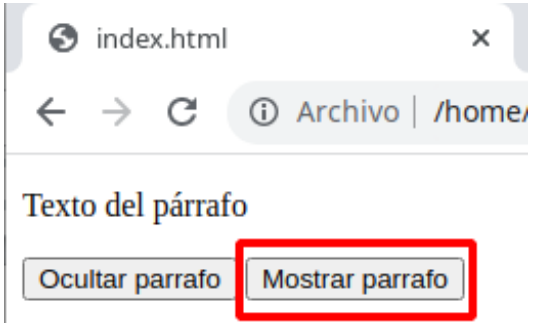
```
1
2   $(document).ready(function(){
3
4     $("#ocultar").click(function(){
5       $('p').hide();
6     });
7
8     $("#mostrar").click(function(){
9       $('p').show();
10    });
11
12  });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <p>Texto del párrafo</p>
10    <button id="ocultar">Ocultar parrafo</button>
11    <button id="mostrar">Mostrar parrafo</button>
12  </body>
13  </html>
```







Eventos en JQuery (Continuación...)

Métodos hide() y show()

Además ambos métodos admiten parámetros de manera opcional. En este caso la definición de los métodos **hide** y **show** quedaría de la siguiente manera:

$(selector).hide(velocidad,callback) – $(selector).show(velocidad,callback)

Donde:

- **Velocidad** → especifica la velocidad con la que se muestra / oculta el elemento. Admite las siguientes opciones: *slow, fast o un número* equivalente a milisegundos.
- **Callback** → define la función a ser ejecutada cuando finalice el método **hide() / show()**.

```
$("button").click(function(){
  $("p").hide(1000);
});
```

El efecto de ocultar el elemento **p** demora 1000 milisegundos.

~~¿Qué es JQuery?~~

~~Instalación de JQuery~~

~~Primeros pasos~~

~~Eventos en JQuery~~

Funciones callback

Funciones callback

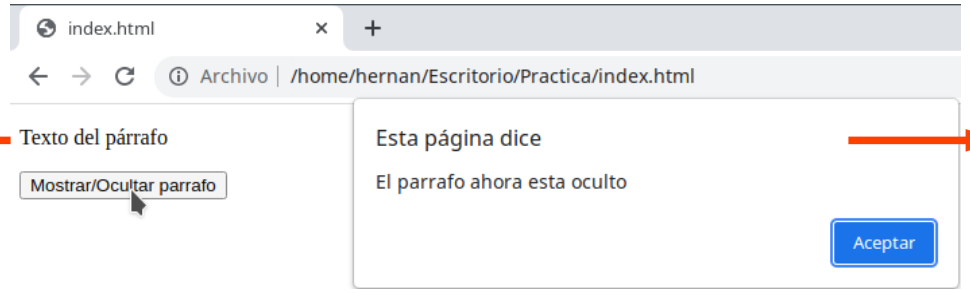
Las instrucciones en JavaScript se ejecutan línea a línea, sin embargo, puede ocurrir que la próxima instrucción se ejecute mientras aún otro método o efecto no haya finalizado su ejecución.

Este tipo de situaciones pueden solucionarse utilizando funciones callback. Veamos un ejemplo **sin utilizar** funciones callback:

```
js > JS index.js > ...
1
2   $(document).ready(function(){
3
4     $("#btn").click(function(){
5       $("p").hide(1000);
6       alert("El parrafo ahora esta oculto");
7     });
8   });
```

```
index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <p>Texto del párrafo</p>
10    <button id="btn">Mostrar/Ocultar parrafo</button>
11  </body>
12  </html>
```

Método **hide** aún en ejecución



El mensaje se superpone a la ejecución del método **hide**

Funciones callback (Continuación...)

Veamos el ejemplo anterior, pero en este caso utilizamos funciones callback para la ejecución del método **alert()** después que finalice la ejecución del método **hide()**.

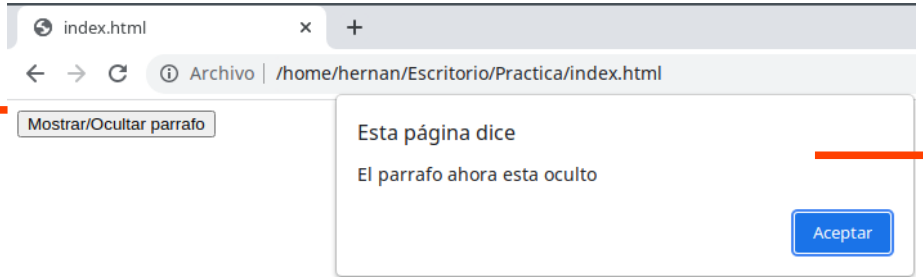
js > index.js > ...

```
1
2   $(document).ready(function(){
3
4     $("#btn").click(function(){
5       $("p").hide(1000, function(){
6         alert("El parrafo ahora esta oculto");
7       });
8     });
9   });
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
6       <script src="js/index.js?"> </script>
7   </head>
8   <body>
9     <p>Texto del párrafo</p>
10    <button id="btn">Mostrar/Ocultar parrafo</button>
11  </body>
12  </html>
```

Método **hide** termina de ejecutarse



El mensaje se muestra al final de la ejecución del método **hide**

¿Consultas?