# Tutorial Ajax

Desarrollo de Software

AÑO 2023

# Introducción

Con el fin de reforzar los elementos vistos durante la sesión, planteamos un ejercicio guiado con que el se espera la resolución de dudas y una clara comprensión del tema, haciendo uso de conceptos y actividades previas se espera que al finalizar el documento el estudiante esté en condiciones de utilizar javascript para hacer la conexión entre un cliente y un servidor por medio de llamados asíncronos.

# Llamados AJAX

Con el fin de poder experimentar, haremos uso de los servicios creados en el documento de apoyo titulado **Tutorial Oracle Cloud Database Actions** puesto que este proceso arrojó como salida la posibilidad de crear un conjunto de servicios con los que interactuaremos.

Para iniciar la exploración con Ajax, crearemos una carpeta que será la raíz de nuestro proyecto. Durante este ejercicio solamente trabajaremos con archivos html y con archivos js (javascript) por lo que iniciaremos creando un archivo llamado index.html y junto a él una carpeta llamada js.

El archivo index.html al abrirlo con un editor de texto (o un IDE) debería verse de la siguiente manera:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

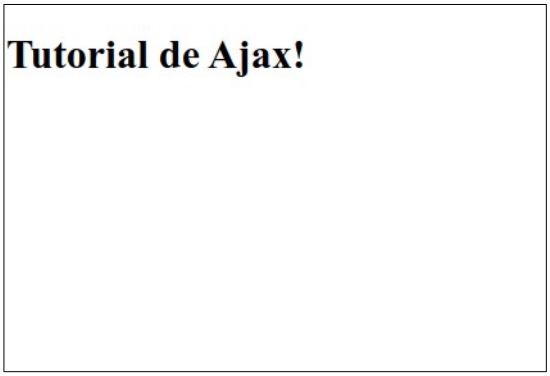
Ahora, agregaremos unos elementos que nos permitirán fácilmente visualizar este contenido de manera especial:

agregaremos un pequeño título, para eso utilizaremos la etiqueta <h1> y la escribiremos justo debajo de la línea que dice <body>.
Debería verse así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
</body>
</html>
```

Es posible que en este momento, el lenguaje html nos resulte nuevo, no debe alarmarnos, puesto que durante el ciclo, tendremos la oportunidad de revisarlo a fondo.

En este momento si guardarmos el archivo ahora lo abrimos con un navegador de internet (no hace falta cerrar el editor de texto) debemos ver lo siguiente:




Para hacer uso de los llamados Ajax, de manera fácil de leer y de escribir, haremos uso de la librería (framework) Jquery. Esta librería nos permitirá mediante una manera de escribir un poco más clara, realizar los llamados Ajax y comprender su funcionamiento.

Haremos en esta práctica llamados a los servicios que desarrollamos previamente, por tal razón, realizaremos 4 llamados. GET, POST, PUT, DELETE

A continuación, agregamos JQUERY. Para agregar la librería podemos hacerlo de varias maneras, dos de ellas son:

1) descargar la librería e incluirla dentro de nuestro proyecto
2)Direccionar nuestro navegador para que antes de cargar nuestro documento la descargue y pueda disponer de ella. Haremos esto entendiendo que siempre corremos el riesgo de no tenerla disponible, puesto que corresponde a un sitio de terceros. Para mitigar esto, haremos uso de google hosted libraries que es realmente estable.
Para visualizar todas las librerías que Google Hosted Libraries alberga, puedes visitar https://developers.google.com/speed/libraries

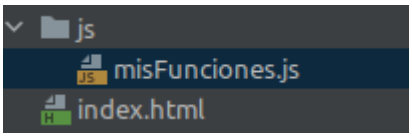
Para que dispongamos en nuestro proyecto de jQuery basta con agregar la siguiente línea desde la cabecera de nuestro documento, es decir dentro de la etiqueta head.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></
script>
```

de esta manera nuestro documento luce así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
</body>
</html>
```

En este momento nuestro navegador no presenta ningún cambio, puesto que no se ha modificado ninguna visualización.
Crearemos en la carpeta JS un archivo al que llamaremos misFunciones.js y luego lo agregaremos en nuestro html de una manera muy similar a como hemos agregado el jQuery.
Por ahora nuestro esquema de archivos debe verse así:




Y al agregar el archivo recién creado al index.html, se verá así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script type="text/javascript" src="js/misFunciones.js"></script>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
</body>
</html>
```

La línea que se ha agregado es <script type="text/javascript" src="js/misFunciones.js"></script>. Esta línea debe ir después del llamado de Jquery y te debes asegurar que la dirección corresponda a la carpeta y el archivo tal cual como los has llamado. Es mejor asegurarse que mayúsculas y minúsculas coincidan.

**EL PRIMER LLAMADO AJAX**

Lo que haremos a continuación es traer la información que el servidor entrega en formato JSON. Para ello, agregaremos un elemento div al body, el cual estará vacío. Lo identificaremos con un id. Agregarmos un botón que será el encargado de "traer" la información. Posteriormente haremos nuestro primer llamado Ajax. El resultado lo imprimiremos por consola para analizar el contenido y elegir lo que nos convenga. Posteriormente mostraremos el contenido en el div que creamos. ¡Manos a la obra!

Creamos el div con un identificador para referirnos a él, agregamos bajo el título, el texto
`<div id="resultado"></div>`

Se vería así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script type="text/javascript" src="js/misFunciones.js"></script>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
<div id="resultado"></div>
</body>
</html>
```

Ya que un elemento div es un contenedor y en este momento está vacío, no se debería modificar en nada lo que visualizamos en el navegador web.

Ahora agregamos el botón. El botón tendrá una acción predeterminada que llamaremos "traerInformacion" (sin tilde). Este es el nombre que pronto le daremos a nuestra función de javascript, así al presionar el botón, se ejecutará el código que ella contenga. Bajo el div agregaremos ahora el botón con:

<button onclick="traerInformacion()">Consultar</button>

Ahora nuestro navegador luego de presionar el botón de actualizar nos debe mostrar lo siguiente:

En este momento empezaremos a editar nuestro archivo JavaScript que se llama misFunciones.js y que está la carpeta js.

Agregaremos un método que se llama traerInformacion. Para agregar un método en javascript basta con escribir:

```
function traerInformacion(){

}
```

Dentro de las llaves (también llamados corchetes) escribiremos el código que traerá la información. Todas las líneas del método anterior se ejecutarán al presionar el botón debido a que su propiedad onclick tiene como valor el **mismo nombre** que tiene la función.

Deliberadamente se ha omitido la tilde de la palabra información. No es recomendable utilizar tildes o la letra ñ en código javascript.

En jQuery un llamado ajax simple se vería más o menos así:

```
$.ajax({
      url:"url del servidor",
      type:"GET",
      datatype:"JSON",
      success:function(respuesta){
            //Acá se puede validar la respuesta.
      }
});
```

La estructura de esta instrucción tiene: URL, ahí escribiremos la dirección del nuestro servidor, en nuestro caso Oracle Cloud nos lo proporciona. En type se escribirá el tipo de petición: GET, POST, PUT, DELETE.

Success corresponde a la función que ejecutará en caso de éxito. También tenemos la posiblidad de agregar funciones según sea el caso, por ejemplo en caso de error o antes de enviar la petición al servidor. También cuando todo está completo. Un campo adicional pero que no se requiere en este ejemplo es data, en el que establecemos qué información enviaremos al servidor. Lo revisaremos en el caso POST. Por su parte, datatype corresponde al tipo de dato que esperamos que el sevidor nos entregue, en nuestro caso es JSON y esto permite que al recibirlo esta información sea formateada y JavaScript lo asuma como un objeto en lugar de una cadena de texto.

Dentro de la función de success se puede ver un parámetro que arbitrariamente hemos llamado respuesta. Pues ahí, sin importar el nombre de la variable, se encontrará la respuesta del servidor en caso de ser exitosa la petición. Para validar, escribiremos como primera instrucción

console.log(respuesta);

De esa manera podemos ver en **la consola del navegador** la información contenida en respuesta.

Para acceder a la consola, presionamos la tecla F12 en la mayoría de navegadores, también se puede hacer clic derecho en cualquier lugar de la página y luego hacer clic en inspeccionar elementos. En el panel que se despliega se puede encontrar la pestaña console.

El código de nuestro javascript se deberá ver así:

```
function traerInformacion(){
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"GET",
        datatype:"JSON",
        success:function(respuesta){
            console.log(respuesta);
        }
    });
}
```

Al refrescar nuestro navegador y presionar el botón se debería poder visualizar el siguiente elemento.

```
▼{items: Array(9), hasMore: false, limit: 25, offset: 0, count: 9, …}
   count: 9
   hasMore: false
  ▶items: (9) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
   limit: 25
  ▶links: (4) [{…}, {…}, {…}, {…}]
   offset: 0
  ▶[[Prototype]]: Object
```

Como podemos ver, hay un arreglo llamado items, el cual tiene la información que nos interesa mostrar en nuestro contenedor div.

A continuación lo que haremos es recorrer el arreglo items y agregar dinámicamente el contenido.

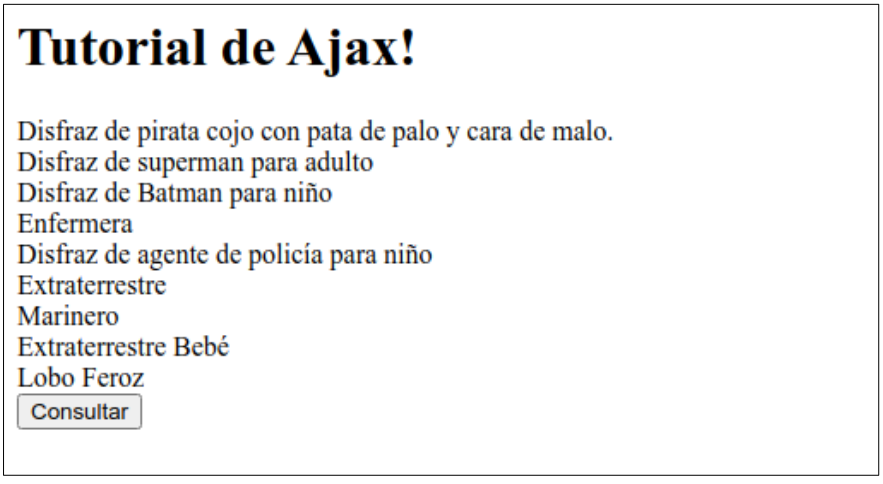
Lo haremos así:

Primero un ciclo que recorra el arreglo items,  el cual es un atributo de respuesta. Al recorrerlo agregaremos al div llamado resultado el contenido de la propiedad name de cada elemento del

arreglo items. Al finalizar agregaremos la etiqueta <br> para realizar un salto de línea. Nuestro código se verá así:

```
function traerInformacion(){
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"GET",
        datatype:"JSON",
        success:function(respuesta){
            console.log(respuesta);
            for(i=0;i<respuesta.items.length;i++){
                $("#resultado").append(respuesta.items[i].name+"<br>");
            }
        }
    });
}
```

Si vamos al navegador y actualizamos, se verá así:




Como podemos ver, nos está mostrando los elementos que teníamos almacenados en la base de datos.

Ahora, se puede experimentar concatenando cualquier otra propiedad. También es posible realizar una tabla para mostrar la información. Para evitar que el llamado ajax quede difícil de leer,

podemos enviar la información a otro método y que sea este método el encargado de agregar la información a la página.

El método lo llamaremos pintarRespuesta y le entregaremos directamente el arreglo ITEMS que está dentro de respuesta. Este método creará una tabla en una variable e irá escribiendo las etiquetas necesarias para visualizar la información.

Por último con la tabla lista, la agregará.

Veamos:

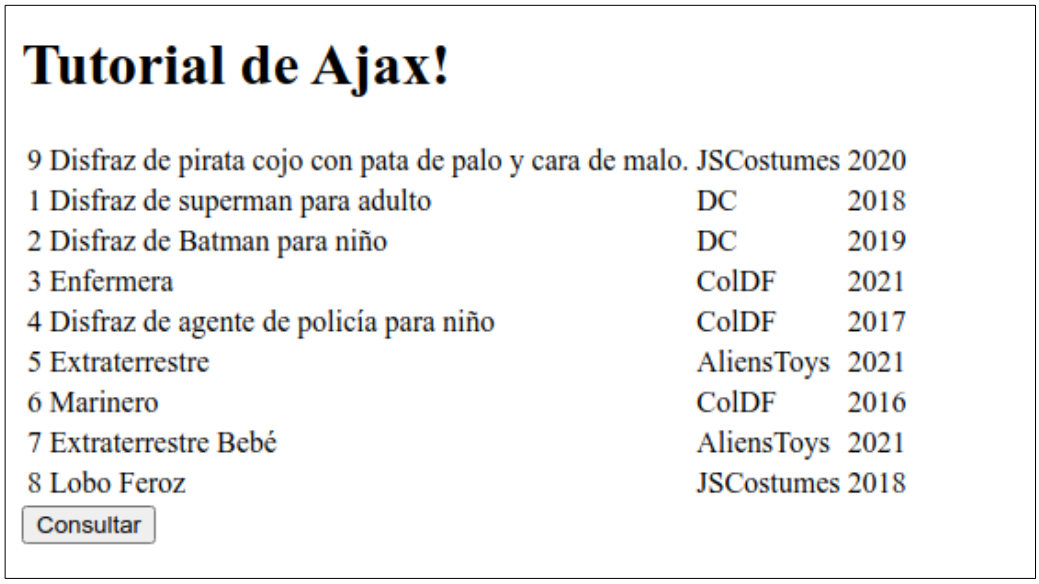
```
function traerInformacion(){
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"GET",
        datatype:"JSON",
        success:function(respuesta){
            console.log(respuesta);
            pintarRespuesta(respuesta.items);
        }
    });
}

function pintarRespuesta(items){

    let myTable="<table>";
    for(i=0;i<items.length;i++){
        myTable+="<tr>";
        myTable+="<td>"+items[i].id+"</td>";
        myTable+="<td>"+items[i].name+"</td>";
        myTable+="<td>"+items[i].brand+"</td>";
        myTable+="<td>"+items[i].model+"</td>";
        myTable+="</tr>";
    }
    myTable+="</table>";
    $("#resultado").append(myTable);

}
```

Al refrescar y presionar el botón, nuestra página lucirá así:




En este momento hemos hecho nuestro primer llamado AJAX, traemos del servidor información y la pintamos en una tabla. Ahora enviaremos información al servidor.

**ENVÍO DE INFORMACIÓN MEDIANTE POST.**

A continuación crearemos cuatro campos de ingreso de información. Será lo más cercano a un formulario, sin embargo **NO HARÁ FALTA** utilizar la etiqueta FORM.

Para ello, crearemos un div bajo el  botón, en este div agregaremos los campos de entrada de información y a cada uno de ellos los identificaremos con un nombre relacionado con la información que contiene.

Este identificador nos permitirá después llamarlo para extraer su información. Esta información será empaquetada y enviada en un llamado AJAX tipo POST.

El div en nuestro index tendrá el siguiente código:

```
<div>
    <input type="number" id="id" placeholder="id">
    <input type="text" id="name" placeholder="name">
    <input type="text" id="brand" placeholder="brand">
    <input type="number" id="model" placeholder="model">
    <input type="number" id="category" placeholder="category_id">
</div>
```

los elementos input tienen tres atributos:

*type: corresponde al tipo de información que recibe, en este caso o texto o números.

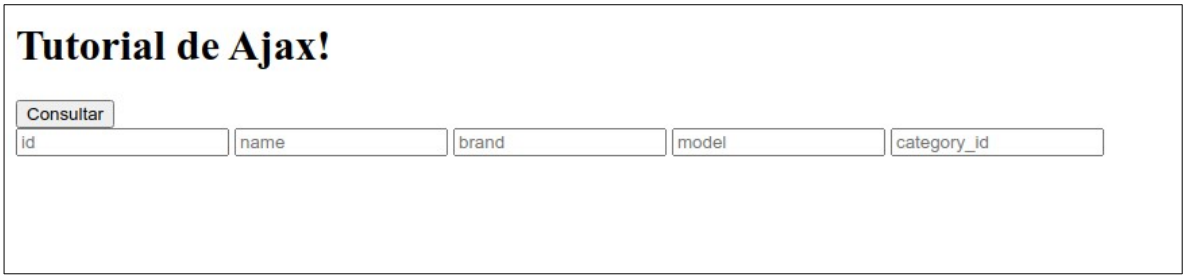
*id: El identificador por medio del cual nos referiremos a él para extraer su información.

*placeholder: Corresponde al texto de ayuda que muestra la caja de texto cuando está vacía. Indica la información que debe escribir el usuario.

En este momento nuestro archivo index.html se debe ver así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script type="text/javascript" src="js/misFunciones.js"></script>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
<div id="resultado"></div>
<button onclick="traerInformacion()">Consultar</button>
<div>
    <input type="number" id="id" placeholder="id">
    <input type="text" id="name" placeholder="name">
    <input type="text" id="brand" placeholder="brand">
    <input type="number" id="model" placeholder="model">
    <input type="number" id="category" placeholder="category_id">
</div>
</body>
</html>
```

Al refrescar, se verá así:




A continuación crearemos un botón guardar y este botón invocará una función que se llamará guardarInformacion. Esta función capturará la información de cada caja de texto, creará una cadena JSON y la enviará al servidor.

```
<button onclick="guardarInformacion()">Guardar!</button>
```

Capturar la información se verá así:

```
function guardarInformacion(){
    let myData={
      id:$("#id").val(),
      name:$("#name").val(),
      brand:$("#brand").val(),
      model:$("#model").val(),
      category_id:$("#category_id").val(),
    };

}
```

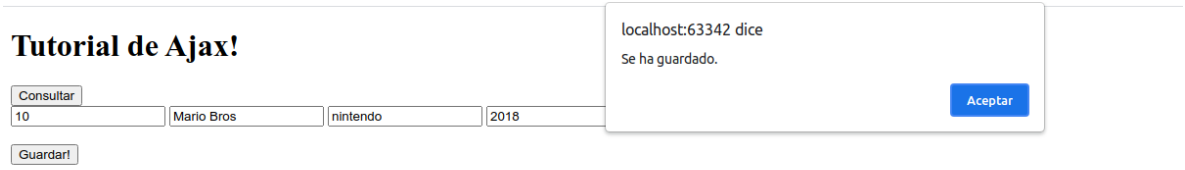
Hemos creado un objeto en javascript en el que sus atributos deben llamarse **exactamente igual** que los parámetros que recibe el método POST en el servidor. En este caso, son id, name, brand, model y category_id. Si escribimos distinto el nombre de uno de estos campos (lado izquierdo) el servidor será incapaz de saber dónde debe ubicar ese dato. A menudo si enviamos un dato y el resultado guardado es nulo, puede que haya algún problema en la manera como lo nombramos.

Ya que tenemos la información capturada (Jquery nos brinda acceso a la información de los input mediante el id y el método val) continuamos con el llamado ajax. Este llamado AJAX será de tipo POST y enviará el objeto myData, pero para enviarlo lo convertirá en formato JSON. El método success mostrará una alerta que permita saber que todo está bien.

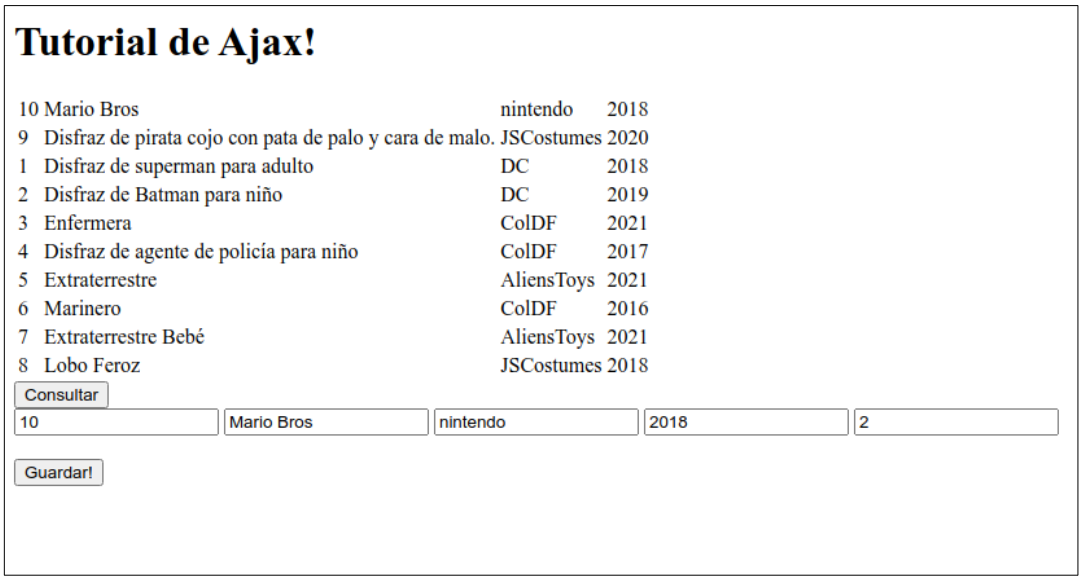
El método ajax se debe hacer a la misma url. El código del método quedaría así:

```
function guardarInformacion(){
    let myData={
      id:$("#id").val(),
      name:$("#name").val(),
      brand:$("#brand").val(),
      model:$("#model").val(),
      category_id:$("#category_id").val(),
    };
   let dataToSend=JSON.stringify(myData);
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"POST",
        data:myData,
        datatype:"JSON",
        success:function(respuesta){
            alert("Se ha guardado.")
        }
    });
}
```

Al probar, lucirá así:




Al presionar el botón Consultar, sin necesidad de recargar la página se verá así:



Ahora si queremos mejorar el ejercicio, podemos limpiar la tabla, la información de las cajas de texto y volver a llamar el método traerInformacion, de esa manera tendremos una experiencia un poco más clara. Todo se ajustará en el *success:*

```
success:function(respuesta){
    $("#resultado").empty();
    $("#id").val("");
    $("#name").val("");
    $("#brand").val("");
    $("#model").val("");
    $("#category_id").val("");
    traerInformacion();
    alert("Se ha guardado.")
}
```

La tabla se borra con el método empty() del div, y en cada campo se pasa por parámetro un par de comillas vacías en el método val. Por último se llama traerInformacion().

**EDITAR INFORMACIÓN**

Para editar los campos simplemente haremos uso del mismo método que hicimos previamente, y cambiaremos la palabra POST por PUT. La información se traerá de las cajas de texto y se enviará. En este caso, se modificará aquel elemento que coincida con el identificador.

El método editar información estará atado a un botón que dirá actualizar. El código de nuestro index y de nuestro javascript se verá así:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Ajax Tutorial</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script type="text/javascript" src="js/misFunciones.js"></script>
</head>
<body>
<h1>Tutorial de Ajax!</h1>
<div id="resultado"></div>
<button onclick="traerInformacion()">Consultar</button>

<div>
    <input type="number" id="id" placeholder="id">
    <input type="text" id="name" placeholder="name">
    <input type="text" id="brand" placeholder="brand">
    <input type="number" id="model" placeholder="model">
    <input type="number" id="category_id" placeholder="category_id">
</div>
<br>
<button onclick="guardarInformacion()">Guardar!</button>
<button onclick="editarInformacion()">Actualizar!</button>

</body>
</html>
```

```
function editarInformacion(){
    let myData={
        id:$("#id").val(),
        name:$("#name").val(),
        brand:$("#brand").val(),
        model:$("#model").val(),
        category_id:$("#category_id").val(),
    };
    let dataToSend=JSON.stringify(myData);
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"PUT",
        data:dataToSend,
        contentType:"application/JSON",
        datatype:"JSON",
        success:function(respuesta){
            $("#resultado").empty();
            $("#id").val("");
            $("#name").val("");
            $("#brand").val("");
            $("#model").val("");
            $("#category_id").val("");
            traerInformacion();
            alert("Se ha Actualizado.")
        }
    });
}
```

En este caso, hemos incluído una línea que dice contentType y hemos hecho que data sea igual al objeto en formato JSON. Con cotentType lo que hacemos es decirle al servidor en qué formato le estoy enviando la información. Se puede actualizar de la misma manera el método POST.

**BORRAR ELEMENTOS**

Para borrar elementos, incluiremos en la tabla de cada elemento un botón que lo borrará. Este botón llamará el método borrar y recibirá por parámetro el ID del elemento. El botón se creará de manera dinámica y para ello modificaremos el método que pinta la tabla.

El método que borra armará la información con el parámetro de la función y posteriormente hará un llamado AJAX al método borrar. Luego limpiará la tabla y llamará de nuevo al método que consulta la información.

Veamos la modificación de la tabla:

La línea que se ha agregado es

myTable+="<td> <button onclick='borrarElemento("+items[i].id+")'>Borrar</button>";

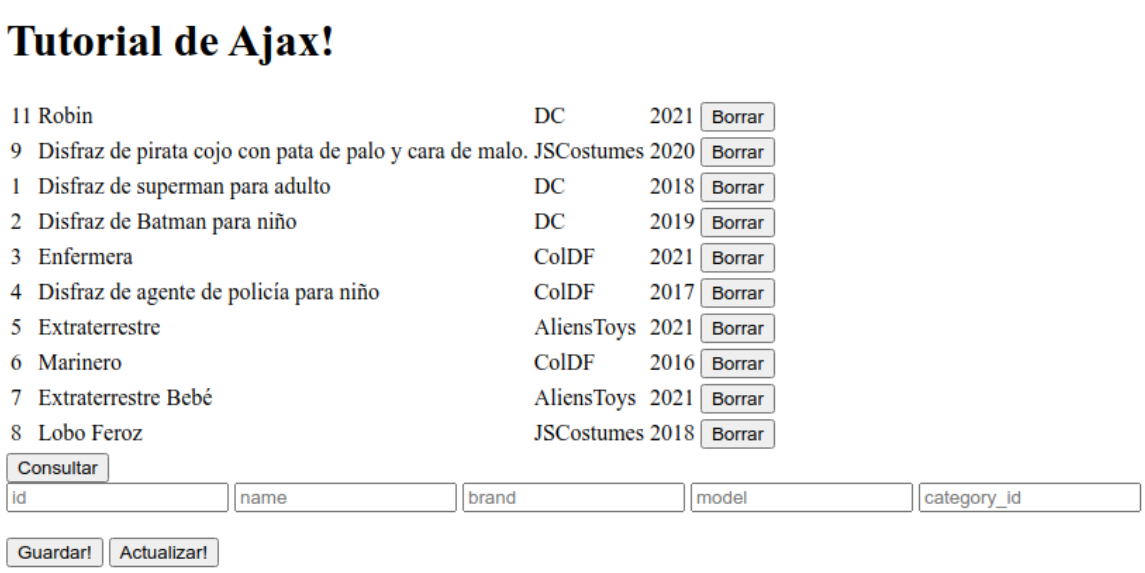
Vale la pena notar que dentro de onclick se ha utilizado una comilla sencilla, de esta manera Javascript entenderá que estamos agregando una comilla en la cadena de texto. Si quiero agregar una comilla doble en la cadena, siempre puedo utilizar el carácter de escape \.

```
function pintarRespuesta(items){

    let myTable="<table>";
    for(i=0;i<items.length;i++){
        myTable+="<tr>";
        myTable+="<td>"+items[i].id+"</td>";
        myTable+="<td>"+items[i].name+"</td>";
        myTable+="<td>"+items[i].brand+"</td>";
        myTable+="<td>"+items[i].model+"</td>";
        myTable+="<td> <button onclick='borrarElemento("+items[i].id+")'>Borrar</button>";
        myTable+="</tr>";
    }
    myTable+="</table>";
    $("#resultado").append(myTable);

}
```

Ahora la tabla luce así:




Nuestro método entonces borrarElemento será así:

```
function borrarElemento(idElemento){
   let myData={
        id:idElemento
    };
   let dataToSend=JSON.stringify(myData);
    $.ajax({
        url:"https://g533ded53dca5a3-db202107191810.adb.sa-saopaulo-1.oraclecloudapps.com/ords/admin/costume/costume",
        type:"DELETE",
        data:dataToSend,
        contentType:"application/JSON",
        datatype:"JSON",
        success:function(respuesta){
            $("#resultado").empty();
            traerInformacion();
            alert("Se ha Eliminado.")
        }
    });
}
```