UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

Javascript, JQuery y JSON avanzado.

www.sceu.frba.utn.edu.ar/e-learning

Módulo 2

JQUERY UI E

INTERACCIÓN CON EL SERVIDOR

“Interactions” de JQuery UI. Que es JSON. Interacción con el servidor.

Presentación de la Unidad:

En esta unidad continuaremos viendo las "interactions" de JQueryUI.

Veremos el formato JSON (Javascript Object Notation) para entender y procesar la respuesta recibida en este formato desde alguno de los componentes de JQueryUI ya sea un widget o una interaction.

Objetivos:

- Aprender a utilizar las interactions de JQueryUI.
- Aprender el formato JSON de comunicación entres las distintas plataformas.
- Procesar desde un widget o interaction información recibida en formato JSON.

Temario:

- DRAGGABLE
- DROPPABLE
- RESIZABLE
- SELECTABLE
- SORTABLE
- JSON
- AUTOCOMPLETE TOMANDO DATOS DE UN JSON

CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

En esta unidad comenzaremos viendo las interacciones que tenemos disponibles en JQueryUI. Luego veremos el formato JSON (Javascript Object Notation) para poder procesar a través de los widgets e interacciones información recibida en este formato. Veremos dos ejemplos de aplicación práctica utilizando el widget autocomplete.

DRAGGABLE

Mediante el método de interacción draggable() podemos dotar a un elemento de la capacidad de poder ser movido a otra parte de la página web por el método de arrastrar y soltar.

En el ejemplo draggable.html encontrarán el funcionamiento por defecto de esta interacción. No hay mucho que explicar con respecto a él simplemente al div que se quiere hacer desplazable se le asigna un id y luego se le aplica la interacción draggable().

En la documentación de la API encontrarán la lista de propiedades, métodos y eventos definidos para utilizar con esta interacción http://api.jqueryui.com/draggable/. No obstante los listamos a continuación:

Options

addClasses
appendTo
axis
cancel
connectToSortable
containment
cursor
cursorAt
delay
disabled
distance
grid
handle
helper
iframeFix
opacity
refreshPositions
revert
revertDuration
scope
scroll

scrollSensitivity
scrollSpeed
snap
snapMode
snapTolerance
stack
zIndex

Methods

destroy
disable
enable
option
widget

Events

create
drag
start
stop

Este ejemplo nos servirá para trabajar con la otra interacción explicada a continuación, droppable().

DROPPABLE

JQuery UI nos permite implementar de forma muy sencilla la funcionalidad de arrastrar y soltar un elemento (definido como draggable) dentro de otro que realiza la función de contenedor (definido como droppable).

De entre sus propiedades destacamos accept, mediante la cual configuraremos qué elementos draggable serán admitidos tras ser soltados sobre él, indicando su atributo 'id' o nombre de clase CSS.

Otra propiedad interesante es tolerance, que permite indicar si los elementos draggable serán admitidos cuando estén parcial o complementamente dentro del elemento droppable. Los valores que puede tomar son:

- fit: el elemento a mover debe ser soltado estando completamente dentro del contenedor.
- intersect: el elemento a mover debe ser soltado cuando al menos el 50% de su área (tanto a lo ancho como a lo alto) esté dentro del contenedor.
- pointer: el elemento será admitido en el contenedor cuando el botón del ratón sea levantado dentro de él.

- touch: el elemento a mover será admitido aunque tan sólo una pequeña parte de él se encuentre dentro del contenedor.

En el ejemplo droppable.html creamos dos elementos que podrán ser arrastrados hacia un contenedor, el cual admitirá sólo a uno en concreto de ellos cuando sea soltado estando completamente dentro de su área. El código del ejemplo se encuentra comentado con las correspondientes explicaciones para que sea entendido mejor.

Vamos a ver otro de los ejemplos que puede llegar a resultarnos de mucha utilidad y que se encuentra dentro de las alternativas enumeradas en la sección demos de jqueryui.com para la interacción droppable. El script correspondiente a este ejemplo es photo-manager.html

El tercer ejemplo que también resulta muy interesante es el incluido en shopping-cart.html. En el se define un acordeón para lo que se utiliza el widget accordion() en el cual se estructuran todos los productos que pueden ser seleccionados, a cada uno de éstos se los define como draggable y a la lista ordenada se la define como droppable y como sortable, para poder ordenar los productos.

En la documentación de la API encontrarán la lista de propiedades, métodos y eventos definidos para utilizar con esta interacción http://api.jqueryui.com/droppable/. No obstante los listamos a continuación:

Options

accept
activeClass
addClasses
disabled
greedy
hoverClass
scope
tolerance

Methods

destroy
disable
enable
option
widget

Events

activate
create
deactivate
drop
out
over

RESIZABLE

Permite redimensionar cualquier elemento del **DOM**.

En el ejemplo resizable.html, tras pulsar en el botón aparecerá un pequeño triángulo en la parte inferior derecha del 'DIV': haz click en él y arrastra para redimensionarlo a lo alto y ancho.

En la documentación de la API encontrarán la lista de propiedades, métodos y eventos definidos para utilizar con esta interacción http://api.jqueryui.com/resizable/. No obstante los listamos a continuación:

Options

alsoResize
animate
animateDuration
animateEasing
aspectRatio
autoHide
cancel
containment
delay
disabled
distance
ghost
grid
handles
helper
maxHeight
maxWidth
minHeight
minWidth

Methods

destroy
disable
enable
option
widget

Events

create
resize
start
stop

SELECTABLE

Permite realizar selecciones en un grupo de elementos, con la posibilidad de realizar selecciones múltiples manteniendo pulsada la tecla Control mientras hacemos click con el botón izquierdo.

Podemos observar su funcionamiento en el script selectable.html.

En la documentación de la API encontrarán la lista de propiedades, métodos y eventos definidos para utilizar con esta interacción http://api.jqueryui.com/selectable/. No obstante los listamos a continuación:

Options

appendTo
autoRefresh
cancel
delay
disabled
distance
filter
tolerance

Methods

destroy
disable
enable
option
refresh
widget

Events

create
selected
selecting
start
stop
unselected
unselecting

SORTABLE

Permite ordenar elementos arrastrando y soltando.

Su funcionamiento por defecto es el de tener una lista de elementos a los cuales le podemos cambiar el orden de ellos. Para ello creamos una lista <ul>, permitiendo cambiar los elementos de posición. Este código lo pueden encontrar en el archivo default.html dentro de la carpeta jquery – sortable.

A partir de este ejemplo vamos a crear una alternativa un poco mas elaborada que nos pueda servir para mostrar datos con un ordenamiento determinado, permitiéndome cambiar ese ordenamiento, guardarlo y la próxima vez que consulte los datos verlos con el último ordenamiento establecido.

Para ello tenemos una tabla con nombres de personas y tenemos una columna ordenamiento dentro de esa misma tabla en la cual se guarda el orden en el que se deben mostrar. A través de el componente sortable() voy a poder reordenar esos nombres de la lista y almacenarlos en la tabla nuevamente para poder ser mostrados con éste último la próxima vez.

Esto podría aplicarse por ejemplo a una tabla de ofertas_destacadas, en la tabla yo tendría las ofertas destacadas y puedo a través del backend ir modificando el orden en el que quiero que sean mostradas esas ofertas en el frontend. O podría hacerlo con una tabla de imágenes de una galería, en fin las utilidades que se les ocurra. La idea es mostrar como podemos hacer interactuar esta interacción de JQuery con un script del lado del servidor.

Entonces nuestro primer paso es a partir del código que tenemos en default.html le agragamos a la lista de elementos un formulario que contenga campos de input hidden que serán los enviados al script php que actualice los datos en la tabla. Tendría que quedarnos algo de este estilo, que lo encontramos en el script sortable.html:

```
<form action="ordenar.php" method="post">
  <ul id="sortable">
    <li class="ui-state-default"><input type="hidden" value="1" name="1" />Item 1</li>
    <li class="ui-state-default"><input type="hidden" value="2" name="2" />Item 2</li>
    <li class="ui-state-default"><input type="hidden" value="3" name="3" />Item 3</li>
    <li class="ui-state-default"><input type="hidden" value="4" name="4" />Item 4</li>
    <li class="ui-state-default"><input type="hidden" value="5" name="5" />Item 5</li>
    <li class="ui-state-default"><input type="hidden" value="6" name="6" />Item 6</li>
    <li class="ui-state-default"><input type="hidden" value="7" name="7" />Item 7</li>
  </ul>
  <input type="submit" value="Ordenar" name="ordenar" />
</form>
```

Pero lo que tenemos que lograr es armar esa lista dinámicamente con los datos obtenidos de la tabla de personas tal como se muestra en el script sortable.php:

```
$result = mysql_query("select * from personas order by orden");

if (mysql_num_rows($result) != 0){
        echo '<form action="ordenar.php" method="post">';
        echo '<ul id="sortable">';
        while ($row = mysql_fetch_assoc($result)){
                echo '<li class="ui-state-default"><input type="hidden"
value="'.$row['id_persona'].'" name="idpersona[]" />'.$row['nombre'].'</li>';
        }
        echo '</ul>';
        echo '<input type="submit" value="Ordenar" name="ordenar" />';
        echo '</form>';
}
else{
        echo 'Aun no hay datos';
}
```

De esta forma creamos un formulario que tiene un campo de input hidden name=idpersona[], es decir un array indexado de inputs entonces será el subíndice de ese array el que determine el nuevo ordenamiento de los nombres en la tabla.

El script ordenar.php actualiza los datos de la tabla recorriendo el array de POST asignandóle al campo ordenamiento de la tabla la posición + 1 del array.

```
for ($i=0;$i<count($_POST['idpersona']);$i++){
//      echo $_POST['orden'][$i].'<br />';
        $j=$i+1;
        $result = mysql_query("update personas set orden = ".$j." where id_persona =
".$_POST['idpersona'][$i]);
```

Como dijimos anteriormente este ejemplo es adaptable a cualquier caso para el cual quiera establecer un orden y quiera permitir modificarlo.

En la documentación de la API encontrarán la lista de propiedades, métodos y eventos definidos para utilizar con esta interacción http://api.jqueryui.com/sortable/. No obstante los listamos a continuación:

Options

appendTo
axis
cancel
connectWith
containment

cursor
cursorAt
delay
disabled
distance
dropOnEmpty
forceHelperSize
forcePlaceholderSize
grid
handle
helper
items
opacity
placeholder
revert
scroll
scrollSensitivity
scrollSpeed
tolerance
zIndex

Methods

cancel
destroy
disable
enable
option
refresh
refreshPositions
serialize
toArray
widget

Events

activate
beforeStop
change
create
deactivate
out
over
receive
remove
sort
start
stop
update

JSON

JSON es una notación Javascript para escribir objetos que se ha hecho bastante popular en el mundo del desarrollo de webs y que se utiliza en diversos lenguajes de programación, componentes (habitualmente Ajax), etc. Su éxito se debe a que es una excelente forma para almacenar información que deseamos compartir entre distintos componentes o lenguajes de las aplicaciones web. Nos daremos cuenta que si trabajamos con Ajax y JQuery o alguno de los otros frameworks Javascript existentes, JSON nos será de gran utilidad.

JSON, cuyas siglas significan JavaScript Object Notation (en español Notación de Objetos de JavaScript), es un formato ligero, fácil de escribir o codificar, así como también es fácil de leer por los seres humanos. Desde Javascript podemos procesar directamente cualquier objeto JSON y existen librerías para la mayoría de los lenguajes de programación que tienen funciones para interpretar este formato. Por ello se ha adoptado universalmente. Para más información podemos visitar el sitio web de JSON en http://www.json.org/.

Esto quiere decir que con JSON podemos comunicar datos fácilmente entre scripts Javascript y scripts PHP. Por ejemplo, pensemos en una validación de formulario que se desea hacer con Ajax. Los datos del formulario se pueden enviar a PHP por medio de POST y luego podríamos desde PHP enviar a Javascript el resultado de validar esos datos en el servidor. Como la validación puede ser positiva o negativa, así como puede tener más o menos códigos de error y acciones a realizar dependiendo de la información procesada, el script PHP tiene que mandar una respuesta más o menos elaborada al script Javascript y una posibilidad es enviar esos datos desde PHP utilizando la notación JSON.

Como dijimos JSON es un formato de intercambio de datos ligero que se utiliza para comunicarse entre distintas plataformas y lenguajes de programación.
Es comparable con el lenguaje de etiquetas XML ya que cumple el mismo propósito. A continuación vemos un ejemplo de cada uno de ellos y luego hacemos una pequeña comparación.
El ejemplo siguiente define un objeto JSON de empleados, con una serie de 3 registros de empleados:

```
{"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]}
```

El ejemplo siguiente de XML también define un objeto empleados con 3 registros de empleados:

```
<employees>
```

```
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

¿QUÉ ES JSON?

- JSON significa Java Script Object Notation
- JSON es un formato de intercambio de datos ligero
- JSON es independiente del lenguaje *
- JSON es "auto-descriptivo" y fácil de entender

* JSON utiliza la sintaxis de JavaScript, pero el formato JSON es sólo texto, al igual que XML.
El texto puede ser leído y utilizado como un formato de datos por cualquier lenguaje de programación.

El formato JSON es sintácticamente idéntico al código para la creación de objetos de JavaScript.

Debido a esta similitud, en lugar de utilizar un programa de análisis (como hace XML), un programa de JavaScript puede utilizar las funciones estándar de JavaScript para convertir datos JSON en objetos nativos de JavaScript.

Código del ejemplo json1.html

```
<!DOCTYPE html>
<html>
<body>

<h2>JSON Object Creation in JavaScript</h2>

<p id="demo"></p>

<script>
var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}';

var obj = JSON.parse(text);
```

```
document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.street + "<br>" +
obj.phone;
</script>

</body>
</html>
```

Parecido a XML porque:

- Tanto JSON y XML es perfectamente legible
- Tanto JSON y XML es hierarchichal (valores dentro de los valores)
- Tanto JSON y XML se pueden analizar y ser utilizados por una gran cantidad de lenguajes de programación
- Tanto JSON y XML se pueden recuperar con un XMLHttpRequest

Diferente a XML porque:

- JSON no utiliza etiqueta final
- JSON es más corto
- JSON es más rápido de leer y escribir
- JSON puede utilizar matrices

La mayor diferencia es la siguiente:

Para analizar un XML hay que utilizar un analizador XML, JSON puede ser analizado por una función estándar de JavaScript.

¿POR QUÉ JSON?

Para aplicaciones AJAX, JSON es más rápido y más fácil que XML ya que XML requiere:

- Recuperar un documento XML
- Utilizar el DOM XML para recorrer el documento
- Extraer y almacenar los valores en las variables

SINTAXIS DE JSON

La sintaxis JSON es un subconjunto de la sintaxis de JavaScript.

Estas son sus características principales:

- Los datos son pares nombre / valor
- Los datos son separados por comas
- Las llaves tienen objetos
- Los corchetes tienen matrices

Los datos JSON se escriben como pares nombre / valor.

Un par nombre / valor consiste en un nombre de campo (entre comillas), seguido de dos puntos, seguido de un valor:

Ejemplo:

"firstName":"John"

 Los nombres en JSON requieren comillas dobles. Los nombres de JavaScript no.

Los valores de JSON pueden ser:

- Un número (entero o punto flotante)
- Una cadena (entre comillas dobles)
- Booleano (verdadero o falso)
- Una matriz (entre corchetes)
- Un objeto (entre llaves)
- nulo

Objetos JSON

Los objetos JSON se escriben dentro de llaves.

Al igual que en JavaScript, los objetos JSON pueden contener múltiples pares nombre / valor:

Ejemplo:

{"firstName":"John", "lastName":"Doe"}

JSON Arrays

Los arrays en JSON se escriben entre corchetes.

Al igual que JavaScript, una matriz JSON puede contener varios objetos:

Ejemplo:

```
"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter","lastName":"Jones"}
]
```

En el ejemplo anterior, el objeto "empleados" es una matriz que contiene tres objetos. Cada objeto es un registro de una persona (con un nombre y un apellido).

JSON USO DE SINTAXIS JAVASCRIPT

Debido a que la sintaxis JSON se deriva de la notación de objetos de JavaScript, no se necesita software adicional para trabajar con JSON en JavaScript.

Con JavaScript se puede crear una matriz de objetos y asignar los datos a ella, así:

Ejemplo:

```
var employees = [
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter","lastName": "Jones"}
];
```

La primera entrada de la tabla de objeto JavaScript se puede acceder de esta manera:

```
// returns John Doe
employees[0].firstName + " " + employees[0].lastName;
```

También se puede acceder de esta manera:

```
// returns John Doe
employees[0]["firstName"] + " " + employees[0]["lastName"];
```

Los datos pueden ser modificados de esta manera:

```
employees[0].firstName = "Gilbert";
```

También se puede modificar asi:

```
employees[0]["firstName"] = "Gilbert";
```

ARCHIVOS JSON

- El tipo de archivo para los archivos JSON es ".json"
- El tipo MIME para el texto JSON es "application / json"

COMO UTILIZAR JSON

Ahora vamos a ver cómo convertir un texto JSON a un objeto JavaScript.

Un uso común de JSON es leer datos de un servidor web, y mostrar los datos en una página web.

Por simplicidad, este se puede demostrar mediante el uso de una cadena como de entrada (en lugar de un archivo).

Creamos un string en Javascript con sintaxis JSON:

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

La sintaxis de JSON es un subconjunto de la sintaxis de JavaScript.

La función de JavaScript JSON.parse (text) se puede utilizar para convertir un texto JSON en un objeto de JavaScript:

```
var obj = JSON.parse(text);
```

Ahora podemos usar en nuestra página el nuevo objeto Javascript de esta forma:

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Pueden ver este ejemplo completo en json2.html

USAR EVAL()

Los navegadores más antiguos, sin el apoyo de la función JavaScript JSON.parse () pueden utilizar la función eval () para convertir un texto JSON en un objeto de JavaScript:

```
var obj = eval ("(" + text + ")");
```

La función eval() puede compilar y ejecutar cualquier JavaScript. Esto representa un potencial problema de seguridad. **Tenemos que tratar de evitarla.**

Es más seguro utilizar un analizador JSON para convertir un texto JSON a un objeto JavaScript.

Un analizador JSON reconocerá únicamente el texto JSON y no compilará scripts.

En los navegadores que proporcionan apoyo JSON nativo, los analizadores JSON son también más rápidos.

Apoyo JSON nativo se incluye en todos los principales navegadores y en la última de ECMAScript (JavaScript) estándar:

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

Para navegadores mas viejos hay disponible una librería en https://github.com/douglascrockford/JSON-js.

JSON HTTP REQUEST

Como dijimos anteriormente un uso común de JSON es leer datos de un servidor web, y mostrar los datos en una página web.

Veremos, en 4 sencillos pasos, cómo leer datos JSON, utilizando XMLHttp.

EJEMPLO JSON3.HTML

Este ejemplo lee un menú desde myTutorials.txt, y muestra el menú en una página web:

```
<div id="id01"></div>

<script>
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

xmlhttp.onreadystatechange = function() {
   if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      var myArr = JSON.parse(xmlhttp.responseText);
      myFunction(myArr);
   }
}
```

```
xmlhttp.open("GET", url, true);
xmlhttp.send();

function myFunction(arr) {
   var out = "";
   var i;
   for(i = 0; i < arr.length; i++) {
      out += '<a href="' + arr[i].url + '">' +
      arr[i].display + '</a><br>';
   }
   document.getElementById("id01").innerHTML = out;
}
</script>
```

Explicación del ejemplo

1 – Declaramos un array de objetos. Le damos a cada objeto un dos propiedades: display y url. Llamamos a este array myArray

```
var myArray = [
{
"display": "JavaScript Tutorial",
"url": "http://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "http://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "http://www.w3schools.com/css/default.asp"
}
]
```

2- Creamos una función Javascript que recorre el array de objetos y muestra el contenido como vínculos HTML.

```
function myFunction(arr) {
   var out = "";
   var i;
   for(i = 0; i < arr.length; i++) {
      out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
   }
   document.getElementById("id01").innerHTML = out;
}
```

Llamamos a la función pasándole como parámetro el array de objetos.

```
myFunction(myArray);
```

Ponemos el array en un archivo de texto myTutorials.txt

```
[
{
"display": "JavaScript Tutorial",
"url": "http://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "http://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "http://www.w3schools.com/css/default.asp"
}
]
```

4- Leemos el archivo de texto con un XMLHttpRequest y usamos myFunction(myArray) para mostar el array

```
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
   var myArr = JSON.parse(xmlhttp.responseText);
   myFunction(myArr);
   }
}

xmlhttp.open("GET", url, true);
xmlhttp.send();
```

JSON FUNCTION FILES

Ahora veremos el mismo ejemplo pero utilizando un archivo de funciones.

EJEMPLO JSON4.HTML

```
<div id="id01"></div>

<script>
```

```
function myFunction(arr) {
   var out = "";
   var i;
   for(i = 0; i<arr.length; i++) {
      out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
   }
   document.getElementById("id01").innerHTML = out;
}
</script>

<script src="myTutorials.js"></script>
```

Explicación del ejemplo

1 – El paso 1 es el mismo, creamos array de objetos. Le damos a cada objeto un dos propiedades: display y url. Llamamos a este array myArray

```
var myArray = [
{
"display": "JavaScript Tutorial",
"url": "http://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "http://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "http://www.w3schools.com/css/default.asp"
}
]
```

2- Creamos una función Javascript que recorre el array de objetos y muestra el contenido como vínculos HTML.

```
function myFunction(arr) {
   var out = "";
   var i;
   for(i = 0; i < arr.length; i++) {
      out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
   }
   document.getElementById("id01").innerHTML = out;
}
```

Llamamos a la función pasándole como parámetro el array de objetos, en este caso pasamos la variable en forma literal.

```
myFunction([
{
"display": "JavaScript Tutorial",
"url": "http://www.w3schools.com/js/default.asp"
```

```
},
{
"display": "HTML Tutorial",
"url": "http://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "http://www.w3schools.com/css/default.asp"
}
]);
```

El llamado a la función lo ponemos en un script externo llamado myTutorial.js e incluimos en nuestro html ese js externo de la siguiente manera:

```
<script src="myTutorials.js"></script>
```

EJEMPLO JSON5.HTML

El último ejemplo toma los datos de un script PHP que recupera la información en una tabla sql y genera un JSON. En el archivo Customers_MYSQLI.php es el script del lado del servidor que lee los datos de la tabla sql y arma un JSON. Los datos devueltos son procesados en json5.html

CREAR UN JSON DESDE PHP

Para crear una cadena para expresar un objeto u otro tipo de variable con JSON en PHP se dispone de una función llamada json_encode(), que recibe lo que deseamos convertir en notación JSON y devuelve una cadena de texto con el JSON producido.

Podemos convertir en JSON cualquier cosa que necesitemos, como una cadena, una variable numérica, un array -normal o asociativo- y objetos con todo tipo de datos dentro. Veremos varios ejemplos que ilustrarán este proceso.

```
$mivariable = "hola";
print_r(json_encode($mivariable));
```

Esto devuelve:

```
"hola"
```

Ahora convertimos un array normal:

```
$miArray = array(1,4,6,8,3,34.8,9,43);
```

```
print_r(json_encode($miArray));
```

Que devuelve:

```
[1,4,6,8,3,34.8,9,43]
```

Estos ejemplos anteriores son sin duda quizás demasiado simples, pero la cosa empieza a tener sentido cuando se convierte un array asociativo en JSON, como se puede ver a continuación.

```
$miArray = array("manzana"=>"verde", "uva"=>"Morada", "fresa"=>"roja");
print_r(json_encode($miArray));
```

```
{"manzana":"verde","uva":"Morada","fresa":"roja"}
```

Este resultado ya tiene una forma parecida a lo que estamos acostumbrados a ver en JSON. Podríamos fijarnos que la cadena que devuelve no tiene saltos de línea e identación de los elementos, con lo que la lectura se dificulta un poco. No obstante, podríamos utilizar esa cadena para crear una variable Javascript y por supuesto podremos acceder a cualquier valor del JSON, de una manera parecida a esta:

```
<script>

var obj = '<?php echo json_encode($miArray);?>';

JSONFrutas = JSON.parse(obj);

alert(JSONFrutas.manzana);

</script>
```

Luego de utilizar JSON.parse podemos acceder a los componentes del JSON como lo hacemos con cualquier objeto, variable, operador punto y luego el nombre de la propiedad a acceder.

El código antes descripto se encuentra en el script json6.php dentro de la carpeta JSON.

CONSEJOS PARA CREAR UN JSON CORRECTAMENTE

La función json_decode() de PHP requiere unas normas un poco más estrictas a la hora de construir un JSON. Si no atendemos esas reglas no obtendremos un mensaje de error, sino que la función devolverá un valor nulo: null. Por tanto, algo que Javascript puede interpretar correctamente puede que no sea así por PHP, así que hay que prestar especial atención sobre una serie de puntos.

1) En la cadena JSON, tanto el nombre de una propiedad como el valor deben estar entre comillas, menos los valores numéricos, o palabras como null, que pueden estar sin comillas.

Incorrecto:
```
$json = json_decode('{nombre: "cualquier valor"}');
```

Correcto:
```
$json = json_decode('{"nombre": "cualquier valor"}');
```

2) Se tienen que utilizar comillas dobles. No se permiten comillas simples para los nombres de variables y sus valores.

Incorrecto:
```
$json = json_decode("{'frutas': ['pera', 'uva']}");
```

Correcto:
```
$json = json_decode('{"frutas": ["pera", "uva"]}');
```

3) No podemos dejarnos una coma que después no tenga nada, así como un corchete o llave de más o de menos.

Incorrecto:
```
$json = json_decode('{"nombre": "valor", }');
```

Correcto:
```
$json = json_decode('{"nombre": "valor"}');
```

4) Tenemos que enviar los datos a json_decode() en UTF-8. Si estamos trabajando en iso-8859-1, habría que codificar la cadena en UTF-8 previamente:

Incorrecto:
```
$json = json_decode('{"mañana": "miércoles"}');
```

Correcto:
```
$json = json_decode(utf8_encode('{"mañana": "miércoles"}'));
```

CREAR OBJETOS PHP A PARTIR DE UNA CADENA JSON

A continuación veremos cómo crear variables u objetos PHP a partir del contenido de cadenas JSON.

PHP puede interpretar datos provenientes de una cadena con notación JSON, de modo que se puedan guardar en variables y luego utilizarlas en los scripts del lado del servidor. En este articulo veremos cómo hacerlo y además destacaremos una serie de consejos para que los JSON estén bien formados y consigamos una correcta

interpretación.

Primero vamos a mostrar un objeto definido con JSON mediante Javascript, para hacer unas comprobaciones y saber si está bien construido.

```
<script>
var objJson = {
  nombre: "Juan",
  idCliente: 22,
  observaciones: null,
  producto: {
    id: 5,
    nombre: "Smartphone LG"
  }
}
alert(objJson.nombre)
alert(objJson.producto.nombre)
</script>
```

Este script Javascript, ejecutado en un navegador, nos mostraría dos mensajes de alerta con valores que se han colocado en el objeto definido con notación JSON. Esto no tiene nada que ver con lo que vamos a ver sobre PHP, pero al menos tenemos claro que nuestro JSON es correcto.

Ahora vamos a crear una variable PHP con la cadena para hacer este objeto y lo cargaremos en una variable PHP interpretando el JSON. Para ello PHP dispone de una función llamada json_decode() que recibe la cadena con notación JSON y devuelve un objeto, o cualquier otro tipo de variable, que estuviera representada en el JSON.

```
<?php
$str_obj_json = '{
  "nombre": "Juan",
  "idCliente": 22,
  "observaciones": null,
  "producto": {
    "id": 5,
    "nombre": "smatphone LG"
  }
}';
$obj_php = json_decode($str_obj_json);
?>
```

Ahora podríamos mostrar el contenido de esa variable con la función de PHP print_r(), que muestra el contenido de variables de manera que se pueda entender por un humano.

```
stdClass Object
(
```

```
  [nombre] => Juan
  [idCliente] => 22
  [observaciones] =>
  [producto] => stdClass Object
  (
    [id] => 5
    [nombre] => Smartphone LG
  )
)
```

Por último podemos intentar acceder a una de las propiedades de este objeto, de esta manera:

```
echo "Una propiedad cualquiera:" . $obj_php->nombre;
```

El código de este ejemplo se encuentra en json7.php dentro de la carpeta JSON.

AUTOCOMPLETE TOMANDO DATOS DE UN JSON

El código del siguiente ejemplo se encuentra en la siguiente ruta JSON/json6/

Para ejecutarlo crear la base de datos cartilla-farma e importar la tabla de calles.

En el script autocompletar.php tenemos el código correspondiente al widget accordion() de JQuery.

Para ello como vimos en la anterior unidad tenemos un <input id="calles" /> dentro de nuestro <body> al cual le asiganamos el widget autocomplete() de la siguiente manera:

```
$( "#calles" ).autocomplete({
        source: "buscar-autocompletar.php",
        minLength: 3,
        select: function( event, ui ) {
                log( ui.item ?
                        "Selected: " + ui.item.value + " aka " + ui.item.label :
                        "Nothing selected, input was " + this.value );
        }
});
```

A la propiedad source le asignamos el nombre del script que buscará en la tabla de calles las calles que coincidan con los caracteres ingresados a partir del tercero (minLength: 3).

El script buscar-autocompletar.php accede a la tabla de calles y devuelve un array en

formato JSON que será interpretado por el autocomplete que mostrará las sugerencias.

Si ejecutamos en el navegador el script http://localhost/JSON/json6/buscar-autocompletar.php?term=bar nos arrojará el JSON generado:

[{"value":"31","label":"ALBARELLOS AVDA"},{"value":"32","label":"ALBARIO"},{"value":"33","label":"ALBARRACIN"},{"value":"34","label":"ALBARRACIN DE SARMIENTO"},{"value":"74","label":"AMENABAR"},{"value":"178","label":"BARADERO"},{"value":"179","label":"BARAGAA"},{"value":"180","label":"BARCA CABO DE HORNOS"},{"value":"181","label":"BARCALA"},{"value":"182","label":"BARCELONA"},{"value":"183","label":"BARILARI ATILIO S ALTE"},{"value":"185","label":"BARILOCHE"},{"value":"186","label":"BAROLO PASAJE"},{"value":"187","label":"BARRACAS"},{"value":"188","label":"BARRAGAN"},{"value":"189","label":"BARRIENTOS"},{"value":"190","label":"BARROS PAZOS JOSE"},{"value":"191","label":"BARZANA"},{"value":"333","label":"CALDERON DE LA BARCA PEDRO"},{"value":"408","label":"CASTRO BARROS"},{"value":"593","label":"DEL BARCO CENTENERA"},{"value":"738","label":"ESCOBAR"},{"value":"832","label":"BARANDA ANDRES"},{"value":"964","label":"HIDALGO BARTOLOME"},{"value":"987","label":"IBARROLA"},{"value":"988","label":"IBARROLA RODRIGO DE"},{"value":"1077","label":"LAMBARE"},{"value":"1304","label":"MITRE BARTOLOME"},{"value":"1394","label":"OBARRIO MANUEL"},{"value":"1740","label":"SANLUCAR DE BARRAMEDA"},{"value":"1830","label":"TABARE"},{"value":"2049","label":"BARRIO SAN MARTIN"},{"value":"2125","label":"ALBARELLOS"},{"value":"2217","label":"BARRIO AUTOPISTA"},{"value":"2363","label":"BARCELO ALBERTO"},{"value":"2399","label":"BARBIERI VICENTE"},{"value":"2400","label":"BARCENA"},{"value":"2401","label":"BARROS ALVARO"},{"value":"2402","label":"CRUZ BARTOLOME"}]