|  |
| --- |
| **JSON** |

**JSON** significa **JavaScript Object Notation**. Es otro de los formatos más habituales para enviar datos desde el servidor. Se trata de un formato de datos que se estructura de igual manera que un objeto de javaScript. Es similar a XML, dado que también es un método de intercambio de datos.

La cadena de datos de JSON tiene un formato similar al de un objeto de javaScript, por lo que los navegadores pueden convertir estos objetos en javaScript con facilidad. Algo que no ocurre con XML, que exige el uso de una serie de nodos necesarios para que javaScript traduzca los datos que se encuentran en el documento XML, en un proceso que suele ser más lento y que exige más programación. Gracias a ello, JSON suele ser la opción preferida para intercambiar datos usando AJAX.

La esencia de JSON es muy similar a la del literal de un objeto javaScript o a la de una colección de pares nombre/valor. Veamos un ejemplo:

*{*

*"nombre" : "pepe",*

*"apellido" : "Alberdi",*

*"ciudad" : "Hernani"*

*}*

Como podemos apreciar, los nombres de las propiedades JSON, a diferencia de los objetos javaScript normales, deben aparecer entre comillas, al igual que todos los valores de las cadenas. Esta definición sería incorrecta:

*{*

*"nombre" : "pepe",*

*apellido : "Alberdi",*

*"ciudad" : "Hernani"*

*}*

En realidad, el servidor web no envía código javaScript al responder a una petición AJAX, sino que únicamente envía texto al que se ha dado un formato similar al de un objeto javaScript: eso es JSON. No será un *código* javaScript real y utilizable hasta que esa cadena de texto se convierta en un verdadero objeto javaScript. JavaScript dispone de una función específica para encargarse de todo el proceso, *$.getJSON()*. Es un método muy similar, tanto en apariencia como en funcionamiento, a *$.get()* y *$.post()*. Su estructura básica es:

*$.getJSON(url, datos, callback)*

Los tres argumentos son los mismos que *utilizábamos* para *$.get()* y *$.post()*. La diferencia está en que *$.getJSON()* se encargará, además, de procesar la respuesta del servidor (la cadena de texto con formato JSON) para convertirla en un objeto javaScript utilizable.

Por tanto, solo necesitaremos aprender a procesar el objeto javaScript mediante la función **callback**. Por ejemplo, supongamos que queremos que AJAX obtenga información de un único contacto, almacenado en un archivo del servidor denominado ***contactos.php***. El archivo devuelto por el servidor contendrá sus datos en formato JSON. La solicitud sería:

*$.getJSON('contactos.php','contactoID=123', procesarContacto);*

La página del servidor utilizará esta información para localizar el contacto en la base de datos, recuperando sus datos y devolviendo al navegador los resultados, en donde la función *callback*, ***procesarContacto***, se encargará de procesarlo.

La función ***procesarContacto()*** tiene un único argumento, **data**, que contiene el objeto javaScript enviado al servidor. Ahora veremos qué labor realiza.

|  |
| --- |
| **Acceso a los datos JSON** |

Tenemos dos formas de acceder a los datos convertidos en objeto a partir de la cadena JSON:

* Sintaxis basada en puntos.
* La notación matricial.

La sintaxis basada en puntos es una forma de indicar las propiedades de un objeto añadiendo un punto entre su nombre y la propiedad a la que se desea acceder. Sabemos cómo se utiliza en las propiedades de muchos objetos javaScript, como cadenas y matrices. Por ejemplo, *‘abc’.length* accede la propiedad *length* de la cadena, devolviendo su longitud.

Supongamos que creamos una variable en la que almacenamos el literal de un objeto:

*var fechaNacimiento ={*

*"nombre" : "pepe",*

*"fecha" : "10/27/1980"*

*}*

En este caso, la variable *fechaNacimiento* contiene el literal del objeto. Por tanto, para recuperar el valor de la propiedad *nombre* del objeto *fechaNacimiento* con la sintaxis basada en puntos deberíamos usar este código:

*fechaNacimiento.nombre*

Ocurre exactamente lo mismo con los datos JSON enviados por el servidor web. Usemos el siguiente código *$.getJSON()* como ejemplo:

*$.getJSON('contactos.php','contactoID=123', procesarContacto);*

*function procesarContacto() {*

}

Asumiendo que el servidor ha devuelto el objeto JSON que vimos antes, éste estará asignado a la variable data (el argumento de la función callback denominada ***procesarContacto***, como si se hubiese ejecutado este código:

*var data = {*

*"nombre" : "pepe",*

*"apellido" : "Alberdi",*

*"ciudad" : "Hernani"*

*}*

Ahora, podremos acceder al valor de nombre:

*data.nombre // "pepe"*

Si el objetivo del programa AJAX es recuperar la información del contacto y mostrarla dentro de un *div* que tenga un *ID* ***info***, el código sería:

*$.getJSON('contactos.php','contactoID=123', procesarContacto);*

*function procesarContacto(data) {*

*var infoHTML='<p>Contacto: ' +* ***data.nombre****;*

*infoHTML+=' ' +* ***data.apellido****;*

*infoHTML+='Ciudad: ' +* ***data.ciudad*** *+ '</p>';*

*$('#info').html(infoHTML);*

*}*

|  |
| --- |
| **Datos JSON complejos** |

Para crear grupos de información más complejos, tendremos que utilizar literales de objetos como parte de los valores de una cadena codificada con JSON. Es decir, anidar literales de objetos dentro de otros literales de objetos.

Un ejemplo: digamos que deseamos que el servidor devuelva información de más de un contacto usando JSON. Enviaremos una petición a un archivo de servidor denominado ***contacto.php*** con una cadena de búsqueda que establece el número de contactos que necesita. Su código podría ser:

*$.getJSON('contactos.php','limite=2', procesarContacto);*

En este caso la información que se envía al servidor es **limite=2**, que indica el número de contactos que deseamos obtener. El servidor debe devolver 2 contactos. Imaginemos que la información del contacto del primero de ellos es la misma que la del ejemplo anterior, y que la del segundo elemento de contacto es un objeto JSON como:

*{*

*"nombre" : "Remedios",*

*"apellido" : "Pena",*

*"ciudad" : "Muelas"*

*}*

En este caso, el servidor podría devolver una cadena JSON que representa a un único objeto que combina los dos anteriores:

*{*

*"****contacto1****" {*

*"nombre" : "pepe",*

*"apellido" : "Alberdi",*

*"ciudad" : "Hernani"*

*},*

*"****contacto2****" {*

*"nombre" : "Remedios",*

*"apellido" : "Pena",*

*"ciudad" : "Muelas"*

*}*

*}*

Asumiendo que la función callback únicamente aceptase un parámetro denominado **data** (por ejemplo ***function procesarContacto(data)***), los datos de la variable se asignarán al objeto JSON igual que si se hubiese ejecutado el siguiente código:

*var* ***data*** *= {*

*"****contacto1****" {*

*"nombre" : "pepe",*

*"apellido" : "Alberdi",*

*"ciudad" : "Hernani"*

*},*

*"****contacto2****" {*

*"nombre" : "Remedios",*

*"apellido" : "Pena",*

*"ciudad" : "Muelas"*

*}*

*}*

Ahora podríamos acceder al objeto del primer contacto contenido en la función callback de esta forma:

*data.contacto1*

Dado que queremos procesar varios contactos al mismo tiempo, deberemos utilizar una función de jQuery que permita pasar de uno a otro: el método ***&.each().*** Su estructura básica es la siguiente:

*$.each(JSON, funcion(nombre,valor){*

*}*

Este método *each()* debe recibir una función y los datos JSON, además del nombre y el valor de cada uno de los elementos del objeto. Este es el aspecto que tendrían los datos JSON en el ejemplo que estamos elaborando:

***// Creamos la petición AJAX con los datos (limite=2) y asignamos la función callback******(procesarContacto)***

*$.getJSON('contactos.php','limite=2', procesarContacto);*

***// Creamos la función callback que aceptará el objeto JSON devuelto por el servidor,***

***// almacenándolo en la variable data.***

*function procesarContacto(data) {*

***// Creamos una variable con una cadena vacía***

*var infoHTML = '';*

***// Seleccionamos todos los objetos de los datos JSON y los procesamos con una función anónima***

***// La función anónima recibe el nombre de cada uno de los objetos principales como una***

***// cadena (el parámetro contacto), así como el valor del objeto (parámetro infoContacto)***

*$.each(JSON, function(contacto,infoContacto){*

*var infoHTML='<p>Contacto: ' +* ***data.nombre****;*

*infoHTML+=' ' +* ***data.apellido +*** *'<br />';*

*infoHTML+='Ciudad: ' +* ***data.ciudad*** *+ '</p>';*

*$('#info').html(infoHTML);*

*} // Fin de each()*

***// Añadimos el HTML terminado a la página***

*$('#info').html(infoHTML);*

*}*