**¿Qué es JSON?**

Primero que todo empecemos por definir que es JSON. JSON es la abreviatura/acrónimo de JavaScript Object Notation, como puedes ver en su definición literal JSON utiliza el código JavaScript como modelo de datos. JSON es una forma de enviar/recibir, vaya intercambiar datos, por lo general entre un cliente y un servidor. XML nació primero y es otra manera de hacerlo, JSON apareció después como otra alternativa a este sistema de envío y recepción de datos.

Pero ¿Porque utilizar un sistema de intercambio de Datos? JSON se emplea habitualmente en entornos donde la cantidad de datos que fluyen entre cliente y servidor es de vital importancia. Según he leído Google y Yahoo! lo utilizan a diestra y siniestra en muchos de sus servicios, como los de mensajería que ya conocemos, Gmail y Yahoo! Mail y también lo utilizan las redes sociales Twitter y Facebook.

Obviamente, si intentas utilizar JSON en un entorno donde el flujo de datos es pequeño no notaras la diferencia ¿Ya te imaginas porque esos grandes sitios web lo utilizan? Tienen millones de usuarios y el tema de intercambio de datos entre sus servidores y los navegadores de los usuarios es critico.

La lista de lenguajes que soportan JSON incluye ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro.

En esta serie de ejemplos de JSON solo veremos un poco de JSON con Javascript y PHP, pues necesitaría toda una vida para explicar cómo se usa en todos los lenguajes mencionados.

**Notación y ejemplos JSON**

Voy a dar por hecho que si estas leyendo este tema es porque ya tienes los fundamentos básicos de Javascript. Para empezar, vamos repasar un poco sobre cómo se definen los arrays y objetos literales en JavaScript. Los temas que veremos a continuación son:

* Los arreglos literales
* Los objetos literales
* Los objetos SON

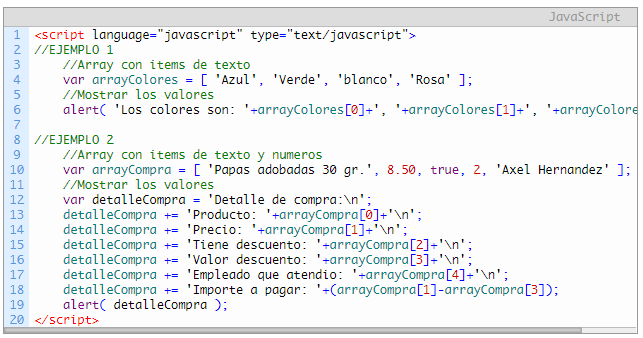
**Los arreglos literales**

La notación literal es una forma de declaración de arreglos que se introdujo a partir de JavaScript 1.2. Los arrays literales se definen con la siguiente sintaxis:



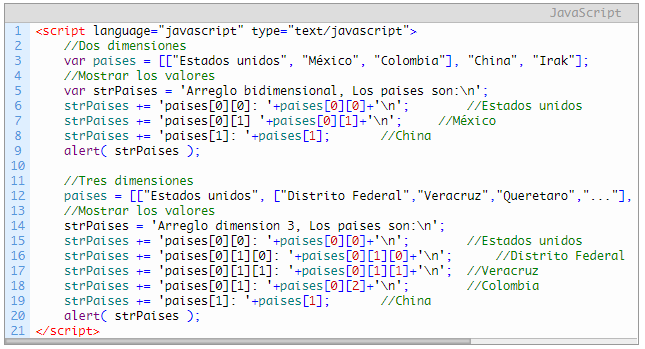
Observa como para declarar un arreglo literal se encierra entre corchetes a sus elementos y se separan por comas. Los elementos de un array literal pueden ser cadenas, números (enteros o decimales), valores booleanos, un arreglo literal, el valor null o incluso expresiones.

A continuación, dos ejemplos de este tipo de arreglos:



En el código anterior hay dos ejemplos; uno en donde se define un array únicamente con elementos de tipo texto (línea 4) y el segundo ejemplo define un array literal con elementos de diferentes tipos de datos (línea 10). Una vez declarado, un arreglo literal se comporta como un arreglo normal, es por eso que para acceder a los elementos se hace por medio de números de índice, ej. arrayColores[0] contiene el primer color.

Ahora vamos a ver como se definen los arreglos literales de más de una dimensión, para que esto siga siendo sencillo solo mostrare arreglos con dos y tres dimensiones, a continuación, el ejemplo:



**Los objetos literales**

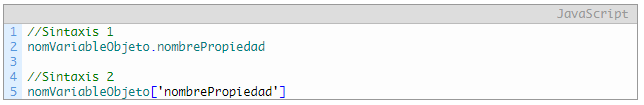
Ahora veamos cómo se definen los objetos literales de javascript, con un ejemplo…



La sintaxis para crear los objetos literales de Javascript es la siguiente;  
**var miObjeto = { ‘propiedad1’: valor1, ‘propiedad2’: valor2 }**

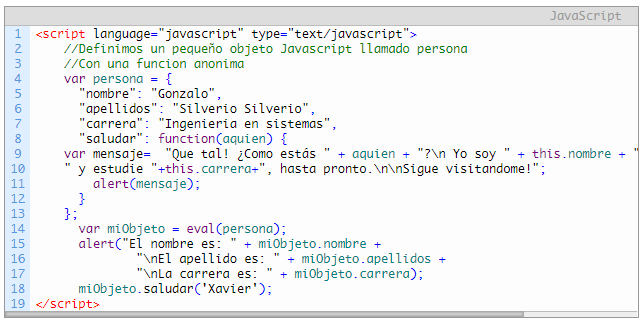
donde **propiedad** es el nombre de una propiedad que estamos creando en el nuevo objeto y **valor** es el valor que estamos asignando a la propiedad. Los valores que pueden asignarse a una propiedad son: un numero (entero o decimal), una cadena de texto, un valor booleano (true o false), null, un array literal o incluso una función anónima.

También puedes ver en el ejemplo anterior que para acceder a las propiedades de un objeto lo podemos hacer directamente por medio del nombre o por medio de índices. Estas son las dos sintaxis:



Observa como en el EJEMPLO 1 accedemos a las propiedades del objeto por medio de la sintaxis 2. Y en el EJEMPLO 2 por medio la sintaxis 1.

Bien, veamos otro ejemplo más ya para terminar con el tema de los Objetos literales de Javascript. En el ejemplo que mostrare a continuación se observa cómo incluso es posible podemos asignar una función anónima a una propiedad del objeto JSON.



Si quisiéramos agregar/modificar una propiedad a un objeto que ya existe entonces los haríamos con la siguiente sintaxis:

objeto.nom\_propiedad = valor

Donde nom\_propiedad es el nombre de una nueva propiedad (o de una propiedad ya existente en el objeto que deseamos modificar) y valor es el nuevo valor que deseamos asignarle.

Ahora que ya sabes cómo declarar arreglos y objetos literales te resultara más fácil crear los objetos JSON…

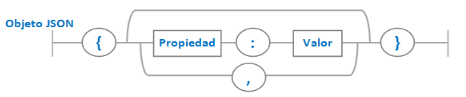
**Los objetos JSON**

Decíamos que JSON es un formato ligero de intercambio de datos. Un documento JSON es un subconjunto de JavaScript, y para esto se basa principalmente en dos estructuras:

Una colección de pares propiedad:valor (objetos)

**Los arreglos literales**

Como vimos anteriormente los objetos comienzan con {y terminan con } en medio están los pares propiedad:valor separados por comas. Veamos el diagrama sintáctico de un objeto JSON:



Donde propiedad es el nombre de una propiedad del objeto JSON y valor pueden ser cadenas, objetos, números (enteros o decimales), valores booleanos (true o false), un arreglo literal o el valor null. Observa cómo es posible anidar objetos dentro de un objeto.

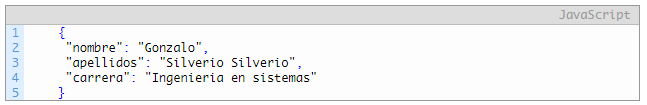
Si te fijas mencione que el valor de una propiedad puede ser un arreglo, debes saber que también es posible anidar arreglos dentro de un arreglo. Un arreglo tiene la siguiente sintaxis: 

De igual manera valor pueden ser cadenas, objetos, números (enteros o decimales), valores booleanos (true o false), un arreglo literal o el valor null. Aquí está la información más completa (de hecho, de ahi me base para mostrarte los diagramas sintácticos).

Por lo general los documentos JSON son obtenidos del servidor a través de AJAX, pero nosotros primeramente veremos dos ejemplos donde el documento JSON lo escribiremos directamente en un archivo HTML, lo convertiremos a objeto y desplegaremos sus valores ahí mismo. Posteriormente en otro ejemplo obtendremos un documento JSON desde el servidor a través de Ajax (con ayuda de JQuery), lo convertiremos a objeto Javascript y mostraremos su contenido al usuario.

Así poco a poco iremos incrementando la complejidad de los ejemplos hasta intercambiar documentos JSON con datos provenientes de una base de datos (es lo interesante y realmente útil).

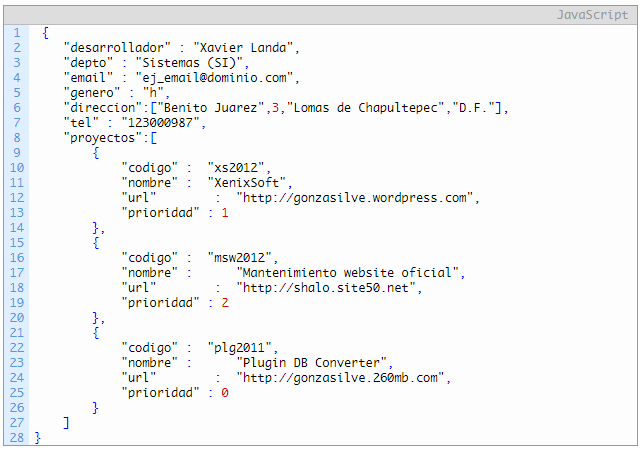
Ejemplo, creamos un documento JSON, así:



El documento anterior contiene 3 campos y sus respectivos valores. Como puedes ver la sintaxis de un documento JSON difiere de manera muy leve con la sintaxis para definir un objeto literal de Javascript:

{‘propiedad1’: valor1, ‘propiedad2’: valor2, … }

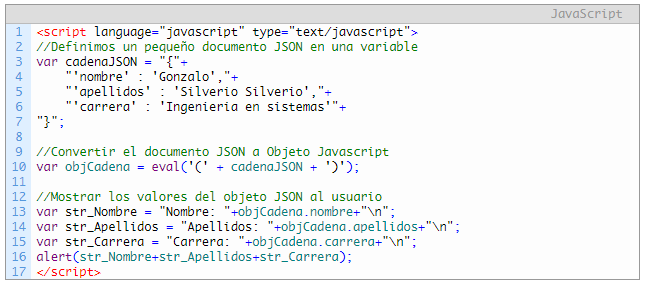
Bien, ahora veamos un documento JSON más elaborado:



El documento anterior incluye varias cosas interesantes; podemos ver que se trata de un solo objeto principal con 7 campos (desarrollador, depto, email, genero, genero, tel, proyectos). El campo dirección es un arreglo con los datos de la dirección de una persona (Nombre de la calle, numero, colonia, ciudad) y el campo proyectos es un arreglo de 3 objetos, en este caso cada objeto (un proyecto) tiene 4 campos. Pero pudiera darse el caso de que algún objeto del arreglo tuviera solo 3, menos o más campos. Por ejemplo, podría ser que un proyecto no tenga una dirección web. JSON no tiene la limitación de que el número de campos en un arreglo de objetos sea el mismo.

¿Ves cómo puedes crear documentos JSON tan flexibles como los necesites (o como tengas de imaginación)?

Ahora veamos cómo se puede convertir un documento JSON en un objeto Javascript, con un ejemplo (con el primer documento JSON para que lo entiendas mas fácilmente):



Observa cómo se usa la funcion eval; e colocado “( )” alrededor del documento JSON, esto es muy importante, no lo olvides. Otra cosa interesante del ejemplo es que una vez que el documento JSON es convertido a objeto JSON, éste tiene todas las propiedades de los objetos vistas anteriormente.

En resumen, los documentos JSON por lo general se obtienen de un servidor (con AJAX) en forma de cadena, estos documentos tienen la sintaxis JSON vista anteriormente. Después de que el documento JSON es obtenido del servidor éste es convertido en un objeto Javascript. Para convertirlo usamos la función eval() que recibe la cadena (el documento JSON) y la devuelve en forma de objeto Javascript.

Ejemplo SON

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> |
|  | <html xmlns="http://www.w3.org/1999/xhtml"> |
|  | <head> |
|  | <link REL="SHORTCUT ICON" href="tux\_icon.ico"> |
|  | <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> |
|  | <title> Ejemplo con Objetos documento JSON </title> |
|  | <script type="text/javascript" src="js/jquery-1.7.2.min.js"></script> |
|  | <script language="javascript" type="text/javascript"> |
|  | $(document).ready(function() { |
|  | var miEnlace = $('#enlace'); |
|  | //Asignar funcion controladora a evento click del enlace |
|  | miEnlace.click(mostrarDatos); |
|  | mostrarTabla(); |
|  | }); |
|  |  |
|  | //Definimos un pequeño documento JSON en una variable |
|  | var cadenaJSON = "{"+ |
|  | "'nombre' : 'Gonzalo',"+ |
|  | "'apellidos' : 'Silverio Silverio',"+ |
|  | "'carrera' : 'Ingenieria en sistemas'"+ |
|  | "}"; |
|  |  |
|  | //Muestra una tabla con los datos del objeto JSON |
|  | function mostrarTabla() { |
|  | //Convertir el documento JSON a Objeto Javascript |
|  | var objCadena = eval('(' + cadenaJSON + ')'); |
|  |  |
|  | //Crear filas con datos |
|  | var str\_FilaNombre = "<tr><td>Nombre</td><td>"+objCadena.nombre+"</td></tr>"; |
|  | var str\_FilaApellidos = "<tr><td>Apellidos</td><td>"+objCadena.apellidos+"</td></tr>"; |
|  | var str\_FilaCarrera = "<tr><td>Carrera</td><td>"+objCadena.carrera+"</td></tr>"; |
|  |  |
|  | //Agregar filas a la tabla |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaNombre); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaApellidos); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaCarrera); |
|  |  |
|  | //Agregar clases |
|  | $('#tbl\_Datos td:first-child').addClass("classGris"); //Fondo gris solo primera columna |
|  | $('#tbl\_Datos td').addClass("classBorde"); //Borde de celdas |
|  |  |
|  | } |
|  |  |
|  | //Muestra una alerta con los datos del objeto JSON |
|  | function mostrarDatos() { |
|  | //Convertir el documento JSON a Objeto Javascript |
|  | var objCadena = eval('(' + cadenaJSON + ')'); |
|  |  |
|  | //Mostrar los valores del objeto JSON al usuario |
|  | var str\_Nombre = "Nombre: "+objCadena.nombre+"\n"; |
|  | var str\_Apellidos = "Apellidos: "+objCadena.apellidos+"\n"; |
|  | var str\_Carrera = "Carrera: "+objCadena.carrera+"\n"; |
|  | alert(str\_Nombre+str\_Apellidos+str\_Carrera); |
|  |  |
|  | } |
|  |  |
|  | </script> |
|  |  |
|  | <style type="text/css"> |
|  | .classGris { background-color: #F0F0F0; } |
|  | .classBorde { border: 1px solid #F0F0F0; } |
|  | </style> |
|  | </head> |
|  | <body > |
|  |  |
|  | <h3>Ejemplo documento JSON</h3> |
|  | Da click en el enlace para realizar el test<br/> |
|  | &nbsp;&nbsp;<a href="#" id="enlace" >Realizar test</a><br/><br/> |
|  |  |
|  | <!-- Tabla de datos --> |
|  | <table id="tbl\_Datos" border="0"> |
|  | <tbody> |
|  |  |
|  | </tbody> |
|  | </table> |
|  | </body> |
|  | <html> |

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> |
|  | <html xmlns="http://www.w3.org/1999/xhtml"> |
|  | <head> |
|  | <link REL="SHORTCUT ICON" href="tux\_icon.ico"> |
|  | <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> |
|  | <title> Ejemplo con Objetos documento JSON </title> |
|  | <script type="text/javascript" src="js/jquery-1.7.2.min.js"></script> |
|  | <script language="javascript" type="text/javascript"> |
|  | $(document).ready(function() { |
|  | var miEnlace = $('#enlace'); |
|  | //Asignar funcion controladora a evento click del enlace |
|  | miEnlace.click(mostrarDatos); |
|  | }); |
|  |  |
|  |  |
|  | //Definimos un pequeño documento JSON en una variable |
|  | var cadenaJSON = "{"+ |
|  | "'desarrollador' : 'Xavier Landa',"+ |
|  | "'depto' : 'Sistemas (SI)',"+ |
|  | "'email' : 'ej\_email@dominio.com',"+ |
|  | "'genero' : 'h',"+ |
|  | "'direccion':['Benito Juarez',3,'Lomas de Chapultepec','D.F.'],"+ |
|  | "'tel' : '123000987',"+ |
|  | "'proyectos':["+ |
|  | "{"+ |
|  | "'codigo' : 'xs2012',"+ |
|  | "'nombre' : 'XenixSoft',"+ |
|  | "'url' : 'http://gonzasilve.wordpress.com',"+ |
|  | "'prioridad' : 1"+ |
|  | "},"+ |
|  | "{"+ |
|  | "'codigo' : 'msw2012',"+ |
|  | "'nombre' : 'Mantenimiento website oficial',"+ |
|  | "'url' : 'http://shalo.site50.net',"+ |
|  | "'prioridad' : 2"+ |
|  | "},"+ |
|  | "{"+ |
|  | "'codigo' : 'plg2011',"+ |
|  | "'nombre' : 'Plugin DB Converter',"+ |
|  | "'url' : 'http://gonzasilve.260mb.com',"+ |
|  | "'prioridad' : 0"+ |
|  | "}"+ |
|  | "]"+ |
|  | "}"; |
|  |  |
|  |  |
|  | function mostrarDatos() { |
|  | //Convertir el documento JSON a Objeto Javascript |
|  | var objCadena = eval('(' + cadenaJSON + ')'); |
|  |  |
|  | //Crear filas con datos |
|  | var str\_FilaNombre = "<tr><td>Desarrollador</td><td>"+objCadena.desarrollador+"</td></tr>"; |
|  | var str\_FilaDepto = "<tr><td>Departamento</td><td>"+objCadena.depto+"</td></tr>"; |
|  | var str\_Genero = (objCadena.genero=='h') ? "Masculino" : "Femenino"; |
|  | var str\_FilaGenero = "<tr><td>Sexo</td><td>"+str\_Genero+"</td></tr>"; |
|  | var str\_direccion = "calle "+objCadena.direccion[0]+ |
|  | " #"+objCadena.direccion[1]+ |
|  | ", colonia "+objCadena.direccion[2]+ |
|  | ", "+objCadena.direccion[3]; |
|  | var str\_FilaDireccion = "<tr><td>Direccion</td><td>"+str\_direccion+"</td></tr>"; |
|  | //alert(str\_direccion); |
|  |  |
|  | var tx\_prioridad = new Array("Importante","Normal","Baja"); |
|  | var projects = objCadena.proyectos; |
|  | var strProyectos = ""; |
|  | for (var g = 0 ; g < (projects.length-1) ; g++) { |
|  | strProyectos += 'Nombre del proyecto: ' + "<b>["+projects[g].codigo+"] "+ projects[g].nombre + '</b><br/>'+ |
|  | 'Sitio web: <b><a href="' + projects[g].url + '">' + projects[g].url + '</a></b><br/>'+ |
|  | 'Prioridad: <b>'+tx\_prioridad[ projects[g].prioridad ]+'</b><br/><br/>'; |
|  | } |
|  | strProyectos += 'Nombre del proyecto: ' + "<b>["+projects[g].codigo+"] "+ projects[g].nombre + '</b><br/>'+ |
|  | 'Sitio web: <b><a href="' + projects[g].url + '">' + projects[g].url + '</a></b><br/>'+ |
|  | 'Prioridad: <b>'+tx\_prioridad[ projects[g].prioridad ]+'</b><br/><br/>'; |
|  |  |
|  | //alert(strProyectos); |
|  | var str\_FilaProyectos = "<tr><td>Proyectos</td><td>"+strProyectos+"</td></tr>"; |
|  |  |
|  | //Agregar filas a la tabla |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaNombre); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaDepto); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaGenero); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaDireccion); |
|  | $('#tbl\_Datos > tbody:last').append(str\_FilaProyectos); |
|  |  |
|  | //Agregar clases |
|  | $('#tbl\_Datos td:first-child').addClass("classGris"); //Fondo gris solo primera columna |
|  | $('#tbl\_Datos td').addClass("classBorde"); //Borde de celdas |
|  |  |
|  | } |
|  | </script> |
|  |  |
|  | <style type="text/css"> |
|  | .classGris { background-color: #F0F0F0; } |
|  | .classBorde { border: 1px solid #F0F0F0; } |
|  | </style> |
|  | </head> |
|  | <body > |
|  |  |
|  | <h3>Ejemplo documento JSON</h3> |
|  | Da click en el enlace para realizar el test<br/> |
|  | &nbsp;&nbsp;<a href="#" id="enlace" >Realizar test</a><br/><br/> |
|  |  |
|  | <!-- Tabla de datos --> |
|  | <table id="tbl\_Datos" border="0"> |
|  | <tbody> |
|  |  |
|  | </tbody> |
|  | </table> |
|  | </body> |
|  | <html> |

https://github.com/notaspro/ejemplos\_json