Notación de Objetos de JavaScript (JSON)

JUAN CARLOS CONDE RAMÍREZ

WEB-TECHNOLOGIES

Objetivos

- Conocer el origen, la utilidad y las bases sintácticas que dieron lugar a JSON.
- Entender las diferencias de notación entre XML y JSON.
- Distinguir las ventajas estructurales que tiene JSON para el intercambio de datos.
- Analizar una implementación típica usando JSON.

Definición y utilidad

- JSON es el acrónimo de ***JavaScript Object Notation***.

- JSON es un formato alternativo de **envío y recepción** de datos, para algunos casos, remplaza a XML o se usa para el envío de texto plano.

- Este formato de datos es más liviano que XML y veremos que hace el código más sencillo ya que utiliza el código JavaScript como modelo de datos.

Base sintáctica: *arreglos literales I*

- Tenemos que recordar un poco como se definen los *arreglos literales* y *objetos literales* en JavaScript, ya que serán las estructuras para la transmisión de datos:

```
var usuario = ['juan', 26];
```

- Como vemos los elementos de un *arreglo literal* se encierran entre corchetes y los valores contenidos van separados por coma.

Base sintáctica: *arreglos literales II*

- Cuando definimos un *arreglo literal* **no le indicamos** a cada elemento de que tipo se trata, podemos almacenar cadenas (entre comillas), números, valores lógicos (*true*, *false*) y el valor *null*.

- Para acceder a los elementos de un *arreglo literal* lo hacemos por su **nombre** y **entre corchetes** indicamos que elementos queremos acceder:

```
alert(usuario[0]); //Imprimimos el primer elemento del array
alert(usuario[1]); //Imprimimos el segundo elemento del array
```

Base sintáctica: *objetos literales I*

- Veamos como se definen los *objetos literales* en JavaScript:

```
var persona={
     'nombre':'juan',
     'clave':'xyz',
     'edad':26 };
```

- Los objetos literales se definen por medio de pares *"nombre":"valor"*. Todo objeto literal tiene un nombre, en el ejemplo se llama persona.

Base sintáctica: *objetos literales II*

- Un objeto literal contiene una serie de propiedades con sus respectivos valores.

- Todas las propiedades de los objetos se encuentran encerradas entre llaves.
  - Cada propiedad y valor se separan por dos puntos.
  - Cada propiedad se separan de las otras propiedades por medio de la coma.

- Para acceder a las propiedades del objeto literal lo podemos hacer de dos formas:

```
//Imprime el valor de la propiedad nombre del objeto persona
alert(persona.nombre);
```

Base sintáctica: *objetos literales III*

- La segunda forma es indicando la propiedad entre corchetes:

```
alert(persona['nombre']);
```

- Luego se pueden combinar *objetos literales* y *arreglos literales*:

```
var persona={
    'nombre':'juan',
    'edad':22,
    'estudios':['primario','secundario']
    };
```

Sintaxis JSON I

- Como podemos ver podemos crear estructuras de datos complejas combinando objetos literales y arreglos literales. Luego para acceder a los distintos elementos tenemos:

```
alert(persona.nombre);
alert(persona.estudios[0]);
alert(persona.estudios[1]);
```

- La sintaxis de JSON difiere levemente de lo visto anteriormente:

```
{
  'nombre':'juan',
  'edad':22,
  'estudios':['primaria','secundaria']
}
```

Sintaxis JSON II

- Como podemos ver en la sintaxis de JSON:
  - no aparecen variables, sino directamente indicamos entre llaves las propiedades y sus valores.
  - También se ha eliminado el punto y coma luego de la llave final.
  - El resto es igual.

- Ahora veamos la diferencia entre JSON y XML utilizando este ejemplo:

Ejemplo: XML vs JSON

- XML:

```
<persona>
    <nombre>juan</nombre>
    <edad>22</edad>
    <estudios>
        <estudio>primaria</estudio>
        <estudio>secundaria</estudio>
    </estudios>
</persona>
```

- Y como vimos en JSON:

```
{
'nombre':'juan',
'edad':22,
'estudios':['primaria','secundaria']
}
```

Ventajas de JSON I

- Podemos ver que la definición de esta estructura en JSON es mucho más simple.
- Aunque cuando la estructura a representar es muy compleja, XML sigue siendo la mejor opción.
- Sin embargo, si tenemos que transmitir estas estructuras por Internet la notación JSON es más liviana.

Ventajas de JSON II

- Otra ventaja es como recuperamos los datos en el navegador:
  - Si se trata de un archivo XML llamamos al método `requestXML` y luego accedemos por medio del DOM
  - En cambio con JSON al llegar el archivo procedemos a generar una variable en JavaScript que recree el objeto literal, esto mediante la función `eval`:

```
var persona=eval('(' + conexion1.responseText + ')');
```

Implementación típica I

- Ya veremos más adelante cómo recuperar los datos del servidor mediante el objeto `XMLHttpRequest`.

- Para probar y generar un objeto a partir de una notación JSON consideremos, el siguiente problema:

1. Implementar una página que contenga un botón.
2. Al ser presionado evaluar un *string* que almacena un objeto literal con notación JSON.
3. El objeto literal debe representar las características de una computadora (procesador, memoria RAM y capacidad de cada disco duro).

Implementación típica II

4. Mostrar los datos mediante el método `alert`.
5. Hay que tener bien en cuenta que en este problema no hay nada de AJAX, ya que no nos comunicaremos con el servidor para el envío de datos.

NOTA: El archivo ***pagina1.html*** y ***funciones.js*** se adjuntan por separado.

- Cuando se presiona el botón se ejecuta la función `presionBoton`. Lo primero que se hace es definir un *string* que contiene un objeto con notación JSON:

```
var cadena="{ 'microprocesador':'pentium'," +
" 'memoria':1024," +
" 'discos':[80,250] }";
```

Implementación típica III

- Enseguida pasamos a evaluar este string:

```
var maquina=eval('(' + cadena + ')');
```

- Ahora si tenemos un objeto JavaScript; esto se logra utilizando la función `eval` de JavaScript.

- Es importante que siempre al *string* que contiene la notación JSON le debamos anteceder el paréntesis de apertura y finalizarlo con el paréntesis de cerrado, esto para que JavaScript no tenga problemas con las llaves de apertura y cierre de la notación JSON.

Implementación típica VI

- Una vez que tenemos el objeto en JavaScript, ya procedemos a acceder a sus atributos:

```
alert('microprocesador:'+maquina.microprocesador);
alert('Memoria ram:'+maquina.memoria);
alert('Capacidad disco 1:'+maquina.discos[0]);
alert('Capacidad disco 2:'+maquina.discos[1]);
```