JSON - Introducción

Intercambio de datos

Al intercambiar datos entre un navegador y un servidor, los datos sólo pueden ser texto.

JSON es texto, se puede convertir cualquier objeto JavaScript en JSON, y enviar el JSON al servidor.

También podemos convertir cualquier JSON recibido del servidor en objetos de JavaScript.

De esta manera se puede trabajar con los datos como objetos de JavaScript, sin análisis complicados y traducciones.

Enviando datos

Si tiene datos almacenados en un objeto de JavaScript, se puede convertir el objeto en JSON, y enviarlo a un servidor:

Ejemplo

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Inténtalo tú mismo "

Recibiendo información

Si recibe datos en formato JSON, puede convertirlo en un objeto de JavaScript:

Ejemplo

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Inténtalo tú mismo "

Almacenamiento de datos

Al almacenar los datos, los datos tienen que ser un formato determinado, y con independencia de donde se elige para almacenarlo, el texto es siempre uno de los formatos legales.

JSON hace posible el almacenamiento de objetos JavaScript como texto.

Ejemplo

Almacenamiento de datos en local storage.

```
//Storing data:
myObj = { "name":"John", "age":31, "city":"New York" };
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

//Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

Inténtalo tú mismo "

JSON sintaxis

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

Reglas de sintaxis JSON

La sintaxis de JSON se deriva de la notación de la sintaxis de objetos de JavaScript:

- Los datos están en pares de nombre / valor
- Los datos se separan por comas
- Las llaves tienen objetos
- Los corchetes tienen matrices

Los datos JSON - Un nombre y un valor

Los datos JSON se escribe como pares nombre/valor.

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

Ejemplo

```
"name":"John"
```

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

JSON - Evalúa a Objetos de JavaScript

El formato JSON es casi idéntica a objetos JavaScript.

En JSON, claves deben ser cadenas, escritos con comillas dobles:

JSON

```
{ "name":"John" }
```

En JavaScript, las claves pueden ser cadenas, números o nombres de los identificadores:

JavaScript

```
{ name: "John" }
```

Los valores JSON

En JSON, los valores deben ser uno de los siguientes tipos de datos:

- una cadena de caracteres
- un número
- un objeto (objeto JSON)
- una matriz
- un valor lógico
- null

En JavaScript valores pueden ser todo lo anterior, además de cualquier otra expresión de JavaScript válida, incluyendo:

- Una función
- una cita
- indefinido

En JSON, valores de cadena deben escribirse entre comillas dobles:

JSON

```
{   "name":"John" }
```

En JavaScript, puede escribir los valores de cadena con comillas dobles o comillas simples:

JavaScript

```
{ name:  'John' }
```

JSON usa la sintaxis de JavaScript

Debido a que la sintaxis JSON se deriva de la notación de objetos JavaScript, se necesita muy poco software adicional para trabajar con JSON dentro de JavaScript.

Con JavaScript puede crear un objeto y asignarle datos, como este:

Ejemplo

```
var person = { "name":"John", "age":31, "city":"New York" };
```

Puede acceder a un objeto JavaScript como este:

Ejemplo

```
// returns John
person.name;
```

Inténtalo tú mismo "

También se puede acceder de esta manera:

Ejemplo

```
// returns John
person["name"];
```

Inténtalo tú mismo "

Los datos se pueden modificar de esta manera:

Ejemplo

```
person.name = "Gilbert";
```

Inténtalo tú mismo "

También se puede modificar así:

Ejemplo

```
person["name"] = "Gilbert";
```

Inténtalo tú mismo "

JSON vs XML

Tanto JSON como XML se pueden usar para recibir datos de un servidor web.

Los siguientes ejemplos de JSON y XML definen un objeto de empleado, con una matriz de 3 empleados:

Ejemplo de JSON

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

Ejemplo XML

```
<  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>
```

JSON es como XML porque

- Tanto JSON como XML son "autodescriptivos" (legibles por humanos)
- Tanto JSON como XML son jerárquicos (valores dentro de los valores)
- Tanto JSON como XML pueden ser analizados y utilizados por muchos lenguajes de programación
- Tanto JSON como XML se pueden buscar con XMLHttpRequest

JSON es diferente de XML porque

- JSON no usa la etiqueta de cierre
- JSON es más corto
- JSON es más rápido para leer y escribir
- JSON puede usar matrices

La mayor diferencia es:

XML tiene que ser analizado con un analizador XML. JSON se puede analizar mediante una función estándar de JavaScript.

Por qué JSON es mejor que XML

XML es mucho más difícil de analizar que JSON.
JSON se analiza en un objeto de JavaScript listo para usar.

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

Usando XML

- Obtener un documento XML
- Use el XML DOM para recorrer el documento
- Extrae valores y almacena en variables

Usando JSON

- Obtener una cadena JSON
- JSON.Parse la cadena JSON

Tipos de datos JSON

Tipos de datos válidos

En JSON, los valores deben ser uno de los siguientes tipos de datos:

- una cuerda
- un número
- un objeto (objeto JSON)
- una matriz
- un booleano
- nulo

Los valores JSON no pueden ser uno de los siguientes tipos de datos:

- Una función
- una cita
- indefinido

Cadenas JSON

Las cadenas en JSON deben escribirse entre comillas dobles.

Ejemplo

```
{ "name":"John" }
```

Números JSON

Los números en JSON deben ser un entero o un punto flotante.

Ejemplo

```
{ "age":30 }
```

Objetos JSON

Los valores en JSON pueden ser objetos.

Ejemplo

```
{
"employee":{ "name":"John", "age":30, "city":"New York" }
}
```

Los objetos como valores en JSON deben seguir las mismas reglas que los objetos JSON.

Matrices JSON

Los valores en JSON pueden ser matrices.

Ejemplo

```
{
"employees":[ "John", "Anna", "Peter" ]
}
```

JSON Booleans

Los valores en JSON pueden ser verdaderos / falsos.

Ejemplo

```
{   "sale":true }
```

JSON nulo

Los valores en JSON pueden ser nulos.

Ejemplo

```
{   "middlename":null }
```

Objetos JSON

Sintaxis de objetos

Ejemplo

```
{   "name":"John", "age":30, "car":null }
```

Los objetos JSON están rodeados de llaves {}.

Los objetos JSON están escritos en pares clave / valor.

Las claves deben ser cadenas y los valores deben ser un tipo de datos JSON válido (cadena, número, objeto, matriz, booleano o nulo).

Las claves y los valores están separados por dos puntos.

Cada par clave / valor está separado por una coma.

Acceso a los valores del objeto

Puede acceder a los valores del objeto usando la notación de punto (.):

Ejemplo

```
myObj =   { "name":"John", "age":30, "car":null };
x = myObj.name;
```

Inténtalo tú mismo "

También puede acceder a los valores del objeto usando la notación de corchete ([]):

Ejemplo

```
myObj =   { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Inténtalo tú mismo "

Looping un objeto

Puede recorrer las propiedades del objeto mediante el uso del bucle for-in:

Ejemplo

```
myObj =   { "name":"John", "age":30, "car":null };
for (x in myObj) {
    document.getElementById("demo").innerHTML += x;
}
```

Inténtalo tú mismo "

En un bucle for-in, use la notación de corchete para acceder a los valores de la propiedad :

Ejemplo

```
myObj =   { "name":"John", "age":30, "car":null };
for (x in myObj) {
    document.getElementById("demo").innerHTML += myObj[x];
}
```

Inténtalo tú mismo "

Objetos JSON anidados

Los valores en un objeto JSON pueden ser otro objeto JSON.

Ejemplo

```
myObj =   {
    "name":"John",
    "age":30,
    "cars": {
        "car1":"Ford",
        "car2":"BMW",
        "car3":"Fiat"
    }
 }
```

Puede acceder a objetos JSON anidados utilizando la notación de puntos o la notación de corchetes:

Ejemplo

```
x = myObj.cars.car2;
//or:
x = myObj.cars["car2"];
```

Inténtalo tú mismo "

Modificar valores

Puede usar la notación de puntos para modificar cualquier valor en un objeto JSON:

Ejemplo

```
myObj.cars.car2  = "Mercedes";
```

Inténtalo tú mismo "

También puede usar la notación de corchetes para modificar un valor en un objeto JSON:

Ejemplo

```
myObj.cars[  "car2"] = "Mercedes";
```

Inténtalo tú mismo "

Eliminar propiedades del objeto

Use la palabra clave delete para eliminar propiedades de un objeto JSON:

Ejemplo

```
delete  myObj.cars.car2;
```

Inténtalo tú mismo "

Matrices JSON

Arrays como objetos JSON

Ejemplo

```
[   "Ford", "BMW", "Fiat" ]
```

Las matrices en JSON son casi las mismas que las matrices en JavaScript.

En JSON, los valores de matriz deben ser de tipo cadena, número, objeto, matriz, booleano o nulo .

En JavaScript, los valores de matriz pueden ser todos los anteriores, más cualquier otra expresión válida de JavaScript, incluidas funciones, fechas y no definidas.

Matrices en objetos JSON

Las matrices pueden ser valores de una propiedad de objeto:

Ejemplo

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accediendo a valores de matriz

Accedes a los valores de la matriz usando el número de índice:

Ejemplo

```
x = myObj.cars[   0];
```

Inténtalo tú mismo "

Looping a través de una matriz

Puede acceder a valores de matriz mediante un bucle for-in:

Ejemplo

```
for (i in myObj.cars) {
  x += myObj.cars[i];
}
```

Inténtalo tú mismo "

O puede usar un ciclo for:

Ejemplo

```
for (i = 0; i < myObj.cars.length; i++) {
  x += myObj.cars[i];
}
```

Inténtalo tú mismo "

Matrices anidadas en objetos JSON

Los valores en una matriz también pueden ser otra matriz, o incluso otro objeto JSON:

Ejemplo

```
myObj = {
  "name":"John",
  "age":30,
  "cars": [
    { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
    { "name":"BMW", "models":[ "320", "X3", "X5" ] },
    { "name":"Fiat", "models":[ "500", "Panda" ] }
  ]
}
```

Para acceder a matrices dentro de matrices, use un bucle for-in para cada matriz:

Ejemplo

```
for (i in myObj.cars) {
    x += "<h1>" + myObj.cars[i].name + "</h1>";
    for (j in myObj.cars[i].models) {
        x += myObj.cars[i].models[j];
    }
}
```

Inténtalo tú mismo "

Modificar valores de matriz

Use el número de índice para modificar una matriz:

Ejemplo

```
myObj.cars[1] = "Mercedes";
```

Inténtalo tú mismo "

Eliminar elementos de matriz

Use la palabra clave delete para eliminar elementos de una matriz:

Ejemplo

```
delete myObj.cars[1];
```

Inténtalo tú mismo "

JSON .parse ()

Un uso común de JSON es intercambiar datos a / desde un servidor web.

Al recibir datos de un servidor web, los datos siempre son una cadena.

Analice los datos con JSON.parse () y los datos se convierten en un objeto JavaScript.

Ejemplo: análisis JSON

Imagine que recibimos este texto de un servidor web:

```
'{ "name":"John", "age":30, "city":"New York"}'
```

Utilice la función JavaScript JSON.parse () para convertir texto en un objeto JavaScript:

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

Asegúrese de que el texto esté escrito en formato JSON o de lo contrario obtendrá un error de sintaxis.

Use el objeto JavaScript en su página:

Ejemplo

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

<script>
document.getElementById("demo").innerHTML = obj.name + ", " +
obj.age;
</script>
```

Inténtalo tú mismo "

JSON desde el servidor

Puede solicitar JSON desde el servidor utilizando una solicitud AJAX

Siempre que la respuesta del servidor esté escrita en formato JSON, puede analizar la cadena en un objeto JavaScript.

Ejemplo

Use XMLHttpRequest para obtener datos del servidor:

```
var   xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

Inténtalo tú mismo "

Eche un vistazo a json_demo.txt

Matriz como JSON

Al usar JSON.parse () en un JSON derivado de una matriz, el método devolverá una matriz de JavaScript, en lugar de un objeto de JavaScript.

Ejemplo

El JSON devuelto por el servidor es una matriz:

```
var   xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myArr[0];
    }
};
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

Inténtalo tú mismo "

Echa un vistazo a json_demo_array.txt

Excepciones

Fechas de análisis

Los objetos de fecha no están permitidos en JSON.

Si necesita incluir una fecha, escríbala como una cadena.

Puede convertirlo de nuevo en un objeto de fecha más tarde:

Ejemplo

Convierte una cadena en una fecha:

```
var   text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text);
obj.birth = new Date(obj.birth);

document.getElementById("demo").innerHTML = obj.name + ", " +
obj.birth;
```

Inténtalo tú mismo "

O bien, puede usar el segundo parámetro, de la función JSON.parse (), llamado reviver .

El parámetro reviver es una función que verifica cada propiedad antes de devolver el valor.

Ejemplo

Convierte una cadena en una fecha, usando la función reviver :

```
var   text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text, function (key, value) {
    if (key == "birth") {
        return new Date(value);
    } else {
        return value;
    }});

document.getElementById("demo").innerHTML = obj.name + ", " +
obj.birth;
```

Inténtalo tú mismo "

Funciones de análisis

Las funciones no están permitidas en JSON.

Si necesita incluir una función, escríbala como una cadena.

Puede convertirlo nuevamente en una función más tarde:

Ejemplo

Convierta una cadena en una función:

```
var   text = '{ "name":"John", "age":"function () {return 30;}", "city":"New
York"}';
var obj = JSON.parse(text);
obj.age = eval("(" + obj.age + ")");

document.getElementById("demo").innerHTML = obj.name + ",
" + obj.age();
```

Inténtalo tú mismo "

Debe evitar el uso de funciones en JSON, las funciones perderán su alcance y deberá usar eval () para convertirlas nuevamente en funciones.

JSON .stringify ()

Un uso común de JSON es intercambiar datos a / desde un servidor web.

Al enviar datos a un servidor web, los datos deben ser una cadena.

Convierta un objeto JavaScript en una cadena con JSON.stringify ().

Stringify un objeto de JavaScript

Imagina que tenemos este objeto en JavaScript:

```
var obj = { "name":"John", "age":30, "city":"New York"};
```

Use la función JavaScript JSON.stringify () para convertirla en una cadena.

```
var myJSON = JSON.stringify(obj);
```

El resultado será una cadena siguiendo la notación JSON.

myJSON ahora es una cadena y está listo para ser enviado a un servidor:

Ejemplo

```
var obj = { "name":"John", "age":30, "city":"New York"};
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
```

Inténtalo tú mismo "

Aprenderá cómo enviar JSON al servidor en el próximo capítulo.

Stringify una matriz de JavaScript

También es posible codificar matrices de JavaScript:

Imagina que tenemos esta matriz en JavaScript:

```
var arr = [ "John", "Peter", "Sally", "Jane" ];
```

Use la función JavaScript JSON.stringify () para convertirla en una cadena.

```
var   myJSON = JSON.stringify(arr);
```

El resultado será una cadena siguiendo la notación JSON.

myJSON ahora es una cadena y está listo para ser enviado a un servidor:

Ejemplo

```
var   arr = [ "John", "Peter", "Sally", "Jane" ];
var myJSON = JSON.stringify(arr);
document.getElementById("demo").innerHTML = myJSON;
```

Inténtalo tú mismo "

Aprenderá cómo enviar JSON al servidor en el próximo capítulo.

Excepciones

Fechas de Stringify

En JSON, los objetos de fecha no están permitidos. La función JSON.stringify () convertirá cualquier fecha en cadenas.

Ejemplo

```
var   obj = { "name":"John", "today":new Date(), "city":"New York"};
var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;
```

Inténtalo tú mismo "

Puede convertir la cadena nuevamente en un objeto de fecha en el receptor.

Funciones de Stringify

En JSON, las funciones no están permitidas como valores de objeto.

La función JSON.stringify () eliminará cualquier función de un objeto JavaScript, tanto la clave como el valor:

Ejemplo

```
var   obj = { "name":"John", "age":function () {return 30;}, "city":"New
York"};
var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;
```

Inténtalo tú mismo "

Esto puede omitirse si convierte sus funciones en cadenas antes de ejecutar la función JSON.stringify ().

Ejemplo

```
var   obj = { "name":"John", "age":function () {return 30;}, "city":"New
York"};
obj.age = obj.age.toString();
var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;
```

Inténtalo tú mismo "

Debe evitar el uso de funciones en JSON, las funciones perderán su alcance y deberá usar eval () para convertirlas nuevamente en funciones.