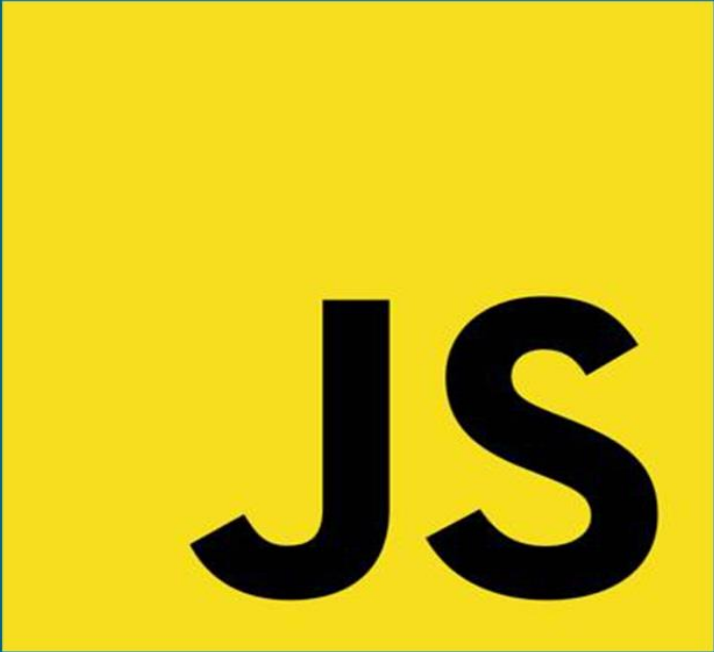


Javascript

DOCS_

JSON

A large yellow square is centered on the page. Inside the square, the letters 'JS' are written in a large, bold, black sans-serif font.

JS

Introducción

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos que se utiliza para transmitir información entre aplicaciones. Fue creado originalmente para ser utilizado con JavaScript, pero actualmente es compatible con muchos lenguajes de programación diferentes.

JSON utiliza una estructura de datos sencilla basada en pares clave-valor para representar objetos y matrices. Los datos se organizan en una estructura jerárquica, donde los objetos pueden contener otros objetos y matrices.

El formato JSON es fácil de leer y escribir para los humanos, y también es fácil de procesar para las aplicaciones. Se utiliza comúnmente en aplicaciones web para transmitir datos entre el cliente y el servidor, y también se utiliza en otras aplicaciones para el intercambio de datos.

JSON significa **J**ava **S**cript **O**bject **N**otation

JSON es un **formato de texto** para almacenar y transportar datos

JSON es "autodescriptivo" y fácil de entender

Ejemplo JSON

Este ejemplo es una cadena JSON:

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

El ejemplo define un objeto con 3 propiedades:

- nombre
- años
- auto

Cada propiedad tiene un valor.

Si analiza la cadena JSON con un programa JavaScript, puede acceder a los datos como un objeto:

```
let personName = obj.name;  
let personAge = obj.age;
```

¿Qué es JSON?

- JSON significa Java **Script Object Notation**
- JSON es un formato **ligero** de intercambio de datos
- JSON se usa para **enviar datos** entre computadoras
- JSON es **independiente** del idioma *

*La sintaxis JSON se deriva de la notación de objetos de JavaScript, pero el formato JSON es solo texto.

El código para leer y generar JSON existe en muchos lenguajes de programación.

El formato JSON fue especificado originalmente por Douglas Crockford .

¿Por qué usar JSON?

El formato JSON es sintácticamente similar al código para crear objetos JavaScript. Debido a esto, un programa JavaScript puede convertir fácilmente datos JSON en objetos JavaScript.

Dado que el formato es solo texto, los datos JSON se pueden enviar fácilmente entre computadoras y ser utilizados por cualquier lenguaje de programación.

JavaScript tiene una función integrada para convertir cadenas JSON en objetos JavaScript:

`JSON.parse()`

JavaScript también tiene una función integrada para convertir un objeto en una cadena JSON:

`JSON.stringify()`

Puede recibir texto puro de un servidor y usarlo como un objeto de JavaScript.

Puede enviar un objeto JavaScript a un servidor en formato de texto puro.

Puede trabajar con datos como objetos de JavaScript, sin análisis ni traducciones complicados.

Almacenamiento de datos

Al almacenar datos, los datos deben tener un formato determinado e, independientemente de dónde elija almacenarlos, el texto siempre es uno de los formatos legales.

JSON hace posible almacenar objetos de JavaScript como texto.

Sintaxis JSON

Reglas de sintaxis JSON

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

- Los datos están en pares de nombre/valor
- Los datos están separados por comas
- Las llaves { } contienen objetos
- Los corchetes [] contienen matrices

Datos JSON: un nombre y un valor

Los datos JSON se escriben como pares de nombre/valor (también conocidos como pares clave/valor).

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

Ejemplo

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

Los nombres JSON requieren comillas dobles.

JSON - objetos de JavaScript

El formato JSON es casi idéntico a los objetos de JavaScript.

En JSON, las claves deben ser cadenas, escritas con comillas dobles:

JSON

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

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

JavaScript

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

Valores JSON

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

- string
- number
- object
- array
- boolean
- null

En JavaScript, los valores pueden ser todos los anteriores, además de cualquier otra expresión válida de JavaScript, que incluye:

- Una función
- una fecha
- indefinido

En JSON, los valores de cadena deben escribirse con comillas dobles:

JSON

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

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

JavaScript

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

Objetos JavaScript

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

Con JavaScript puedes crear un objeto y asignarle datos, así:

Ejemplo

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

Puedes acceder a un objeto JavaScript como este:

Ejemplo

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

También se puede acceder así:

Ejemplo

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

Los datos se pueden modificar así:

Ejemplo

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

También se puede modificar así:

Ejemplo

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

JSON frente a 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 empleados, con una matriz de 3 empleados:

Ejemplo 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 valores)
- Muchos lenguajes de **programación** pueden analizar y utilizar tanto JSON como XML.
- Tanto JSON como XML se pueden obtener con **XMLHttpRequest**

JSON es diferente a XML porque

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

La mayor diferencia es:

XML debe analizarse 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 JavaScript listo para usar.

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

Usando XML

- Obtener un documento XML
- Use el DOM XML para recorrer el documento
- Extraer valores y almacenar en variables

Usando JSON

- Obtener una cadena JSON
- JSON. Analizar la cadena JSON

Tipos de datos JSON

Tipos de datos válidos

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

- String
- number
- objeto (objeto JSON)
- array
- booleano
- nulo

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

- Una función
- una fecha
- indefinido

JSON String

Las cadenas en JSON deben escribirse entre comillas dobles.

Ejemplo

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

JSON Number

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

Ejemplo

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

JSON Objects

Los valores en JSON pueden ser objetos.

Ejemplo

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

Los objetos como valores en JSON deben seguir la sintaxis de JSON.

JSON Arrays

Los valores en JSON pueden ser matrices.

Ejemplo

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

JSON Booleans

Los valores en JSON pueden ser verdadero/falso.

Ejemplo

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

JSON null

Los valores en JSON pueden ser nulos.

Ejemplo

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

JSON .parse()

Un uso común de JSON es intercambiar datos hacia/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 de JavaScript.

Ejemplo: análisis de JSON

Imagina 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:

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

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

Usa el objeto JavaScript en tu página:

Ejemplo

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

<h2>Creating an Object from a JSON String</h2>

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

<script>
const txt = '{"name":"John", "age":30, "city":"New York"}'
const obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>

</body>
</html>
```

Array como JSON

Cuando se usa `JSON.parse()` en un JSON derivado de una matriz, el método devolverá una matriz de JavaScript, en lugar de un objeto de JavaScript.

Ejemplo

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

<h2>Parsing a JSON Array.</h2>
<p>Data written as an JSON array will be parsed into a JavaScript array.</p>
<p id="demo"></p>

<script>
const text = '[ "Ford", "BMW", "Audi", "Fiat" ]';
const myArr = JSON.parse(text);
document.getElementById("demo").innerHTML = myArr[0];
</script>

</body>
</html>
```

Excepciones

Parsing Dates

Los objetos de fecha no están permitidos en JSON.

Si necesita incluir una fecha, escríbala como una cadena.

Puede volver a convertirlo en un objeto de fecha más tarde:

Ejemplo

Convierte una cadena en una fecha:

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

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

Ejemplo

Convierta una cadena en una fecha, usando la función `reviver` :

```
const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';  
const 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;
```

Parsing Functions

Las funciones no están permitidas en JSON.

Si necesita incluir una función, escríbala como una cadena.

Puede volver a convertirlo en una función más tarde:

Ejemplo

Convierte una cadena en una función:

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

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

Debe evitar usar funciones en JSON, las funciones perderán su alcance y tendrá que usarlas `eval()` para volver a convertirlas en funciones.

JSON .stringify()

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

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

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

Stringify un objeto de JavaScript

Imagina que tenemos este objeto en JavaScript:

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

Utilice la función de JavaScript `JSON.stringify()` para convertirlo en una cadena.

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

El resultado será una cadena siguiendo la notación JSON.

`myJSON` ahora es una cadena y está lista para ser enviada a un servidor:

Ejemplo

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

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

Stringify un array de JavaScript

También es posible stringify un array de JavaScript:

Imagina que tenemos esta matriz en JavaScript:

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

Utilice la función de JavaScript `JSON.stringify()` para convertirlo en una cadena.

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

El resultado será una cadena siguiendo la notación JSON.

`myJSON` ahora es una cadena y está lista para ser enviada a un servidor:

Ejemplo

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

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


Almacenamiento de datos

Al almacenar datos, los datos deben tener un formato determinado e, independientemente de dónde elija almacenarlos, el texto siempre es uno de los formatos legales.

JSON hace posible almacenar objetos de JavaScript como texto.

Ejemplo

Almacenamiento de datos en el almacenamiento local

```
// Storing data:

const myObj = {name: "John", age: 31, city: "New York"};

const myJSON = JSON.stringify(myObj);

localStorage.setItem("testJSON", myJSON);
```

```
// Retrieving data:

let text = localStorage.getItem("testJSON");

let obj = JSON.parse(text);

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

Excepciones

Stringify Dates

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

Ejemplo

```
const obj = {name: "John", today: new Date(), city : "New York"};

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

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

Stringify Functions

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

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

Ejemplo

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

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

Ejemplo

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

Si envía funciones usando JSON, las funciones perderán su alcance y el receptor tendrá que usar `eval()` para volver a convertirlas en funciones.

JSON Object

Esta es una cadena JSON:

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

Dentro de la cadena JSON hay un objeto literal JSON:

Stringify Functions

Los literales de objeto JSON están rodeados por llaves {}.

Los literales de objeto JSON contienen pares clave/valor.

Las claves y los valores están separados por dos puntos.

Las claves deben ser cadenas y los valores deben ser un tipo de datos JSON válido:

- string
- number
- object
- array
- boolean
- null

Cada par clave/valor está separado por una coma.

Es un error común llamar a un objeto JSON literal "un objeto JSON".

JSON no puede ser un objeto. JSON es un formato de cadena.

Los datos son solo JSON cuando están en formato de cadena. Cuando se convierte en una variable de JavaScript, se convierte en un objeto de JavaScript.

Objetos JavaScript

Puede crear un objeto JavaScript a partir de un objeto literal JSON:

Ejemplo

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

Normalmente, creas un objeto JavaScript analizando una cadena JSON:

Ejemplo

```
myJSON = '{"name":"John", "age":30, "car":null}';  
myObj = JSON.parse(myJSON);
```

Acceso a valores de objetos

Puede acceder a los valores de los objetos utilizando la notación de punto (.):

Ejemplo

```
const myJSON = '{"name":"John", "age":30, "car":null}';  
const myObj = JSON.parse(myJSON);  
x = myObj.name;
```

También puede acceder a los valores de los objetos utilizando la notación de corchetes ([]):

Ejemplo

```
const myJSON = '{"name":"John", "age":30, "car":null}';  
const myObj = JSON.parse(myJSON);  
x = myObj["name"];
```

Bucle de un objeto

Puede recorrer las propiedades del objeto con un bucle for-in:

Ejemplo

```
const myJSON = '{"name":"John", "age":30, "car":null}';

const myObj = JSON.parse(myJSON);


let text = "";

for (const x in myObj) {

    text += x + ", ";

}
```

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

Ejemplo

```
const myJSON = '{"name":"John", "age":30, "car":null}';

const myObj = JSON.parse(myJSON);


let text = "";

for (const x in myObj) {

    text += myObj[x] + ", ";

}
```

JSON Array

Esta es una JSON STRING:

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

Dentro de la cadena JSON hay un literal de matriz JSON:

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

Las matrices en JSON son casi lo mismo 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, además de cualquier otra expresión de JavaScript válida, incluidas funciones, fechas e indefinidas.

JavaScript Array

Puede crear una matriz de JavaScript a partir de un literal:

Ejemplo

```
myArray = ["Ford", "BMW", "Fiat"];
```

Puede crear una matriz de JavaScript analizando una cadena JSON:

Ejemplo

```
myJSON = '["Ford", "BMW", "Fiat"]';
```

```
myArray = JSON.parse(myJSON);
```

Acceder a valores del array

Accede a los valores de la matriz por índice:

Ejemplo

```
myArray[0];
```

Array en Objetos

Los objetos pueden contener matrices:

Ejemplo

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

Accede a los valores de la matriz por índice:

Ejemplo

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

Bucle a través de un Array

Puede acceder a los valores de la matriz mediante el uso de un for inbucle:

Ejemplo

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

O puedes usar un forbucle:

Ejemplo

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


Servidor JSON

Un uso común de JSON es intercambiar datos hacia/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 de JavaScript.

Enviando datos

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

Ejemplo

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

<h2>Convert a JavaScript object into a JSON string, and send it to the server.</h2>

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

</body>
</html>
```

Recibiendo información

Si recibe datos en formato JSON, puede convertirlos fácilmente en un objeto JavaScript:

Ejemplo

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

<h2>Convert a JSON string into a JavaScript object.</h2>

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

<script>
const myJSON = '{"name":"John", "age":31, "city":"New York"}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
</script>

</body>
</html>
```

JSON desde un servidor

Puede solicitar JSON desde el servidor mediante 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:

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

<h2>Fetch a JSON file with XMLHttpRequest</h2>
<p id="demo"></p>

<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myObj.name;
}
xmlhttp.open("GET", "json_demo.txt");
xmlhttp.send();
</script>

</body>
</html>
```

Eche un vistazo a json_demo.txt

```
{
  "name":"John",
  "age":31,
  "pets":[
    { "animal":"dog", "name":"Fido" },
    { "animal":"cat", "name":"Felix" },
    { "animal":"hamster", "name":"Lightning" }
  ]
}
```

Array como JSON

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

Ejemplo

JSON devuelto desde un servidor como una matriz:

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

<h2>Fetch a JSON file with XMLHttpRequest</h2>
<p>Content written as an JSON array will be converted into a JavaScript array.</p>
<p id="demo"></p>

<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myArr = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myArr[0];
}
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
</script>

</body>
</html>
```

Eche un vistazo a `json_demo_array.txt`

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

JSON HTML

JSON se puede traducir muy fácilmente a JavaScript.

JavaScript se puede utilizar para hacer HTML en sus páginas web.

Tabla HTML

Haz una tabla HTML con los datos recibidos como JSON:

Ejemplo

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

<h2>Make a table based on JSON data.</h2>

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

<script>
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  let text = "<table border='1'>"
  for (let x in myObj) {
    text += "<tr><td>" + myObj[x].name + "</td></tr>";
  }
  text += "</table>"
  document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_html_table.php");
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
</script>

</body>
</html>
```

Tabla HTML dinámica

Haz la tabla HTML basada en el valor de un menú desplegable:

Ejemplo

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

<h2>Make a table based on the value of a drop down menu.</h2>

<select id="myselect" onchange="change_myselect(this.value)">
  <option value="">Choose an option:</option>
  <option value="customers">Customers</option>
  <option value="products">Products</option>
  <option value="suppliers">Suppliers</option>
</select>

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

<script>
function change_myselect(sel) {
  const dbParam = JSON.stringify({table:sel,limit:20});
  const xmlhttp = new XMLHttpRequest();
  xmlhttp.onload = function() {
    myObj = JSON.parse(this.responseText);
    text = "<table border='1'"
    for (x in myObj) {
      text += "<tr><td>" + myObj[x].name + "</td></tr>";
    }
    text += "</table>"
    document.getElementById("demo").innerHTML = text;
  }
  xmlhttp.open("POST", "json_demo_html_table.php", true);
  xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xmlhttp.send("x=" + dbParam);
}
</script>

</body>
</html>
```

Lista desplegable HTML

Cree una lista desplegable HTML con los datos recibidos como JSON:

Ejemplo

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

<h2>Make a drop down list based on JSON data.</h2>
<p id="demo"></p>

<script>
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  let text = "<select>"
  for (let x in myObj) {
    text += "<option>" + myObj[x].name + "</option>";
  }
  text += "</select>"
  document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_html_table.php");
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
</script>

</body>
</html>
```