



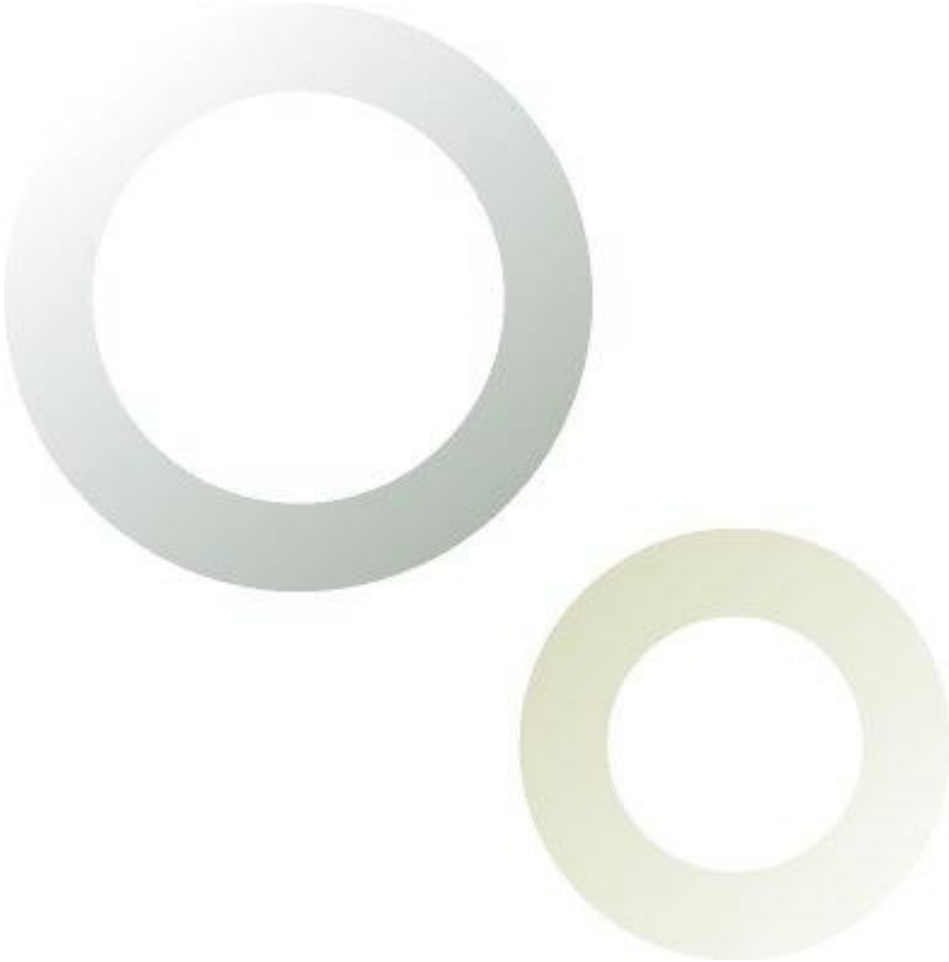
LENGUAJES DE MARCAS

UD8. Tratamiento de documentos JSON

Desarrollo de Aplicaciones Web

Profesores: Diana Bautista / Óscar Villar

Índice de contenido

- 
1. Introducción
 2. Reglas de sintaxis JSON
 3. Tipos de datos válidos
 4. JSON frente a XML
 5. Función PARSE
 6. Función stringify
 7. Objetos JSON
 8. Matrices como objetos JSON

1. Introducción

Introducción

JSON (acrónimo de **JavaScript Object Notation**, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos

.json es un archivo que contiene una serie de datos estructurados en formato de texto y se usa para transferir información entre sistemas. Es importante decir que, a pesar de su origen estar en el lenguaje JavaScript, JSON no es un lenguaje de programación.

JSON es una notación para la transferencia de datos que sigue un estándar específico. Por eso, puede emplearse en [diferentes lenguajes de programación](#) y de sistemas.

Los datos contenidos en un archivo en formato JSON deben estructurarse por medio de una colección de pares con nombre y valor o deben ser una lista ordenada de valores. Sus elementos tienen que contener:

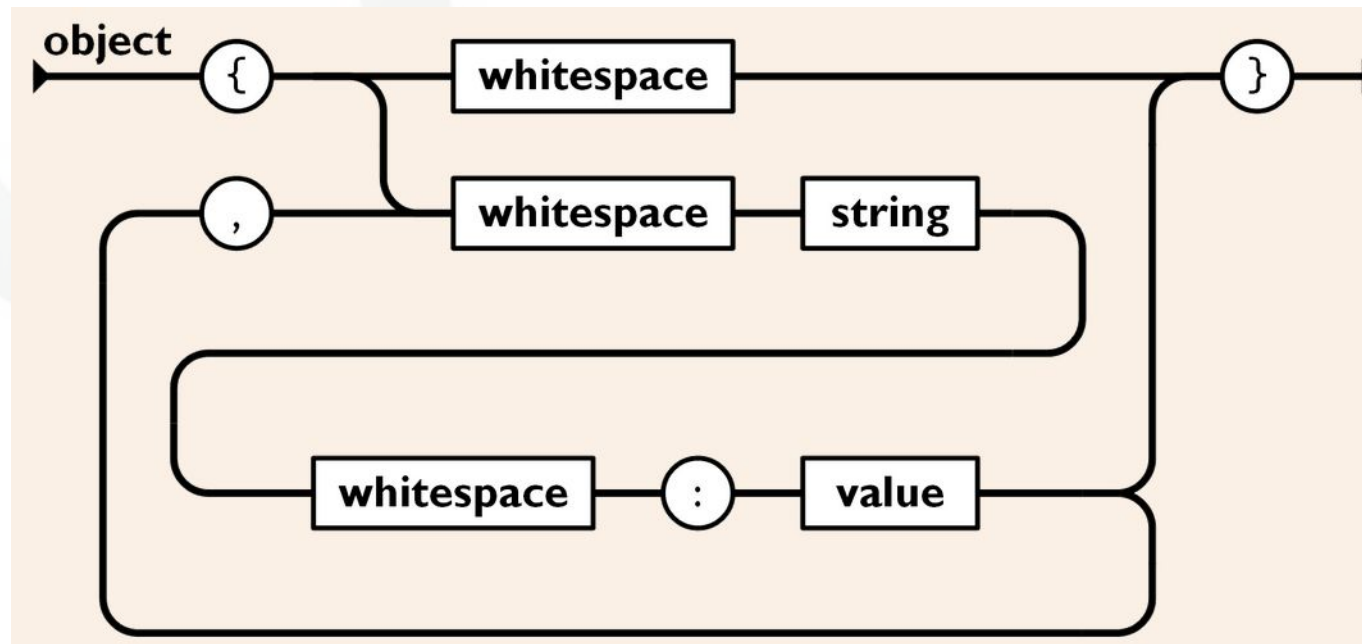
- **Clave:** corresponde al identificador del contenido. Por eso, debe ser un string delimitada por comillas.
- **Valor:** representa el contenido correspondiente y puede contener los siguientes tipos de datos: string, array, object, number, boolean o null.

Introducción

- Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico (parser) para él.
- Gran aceptación por la comunidad de desarrolladores AJAX
- Si bien se tiende a considerar JSON como una *alternativa* a XML, lo cierto es que no es infrecuente el uso de JSON y XML en la misma aplicación
 - Una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP necesita hacer uso de ambos formatos.

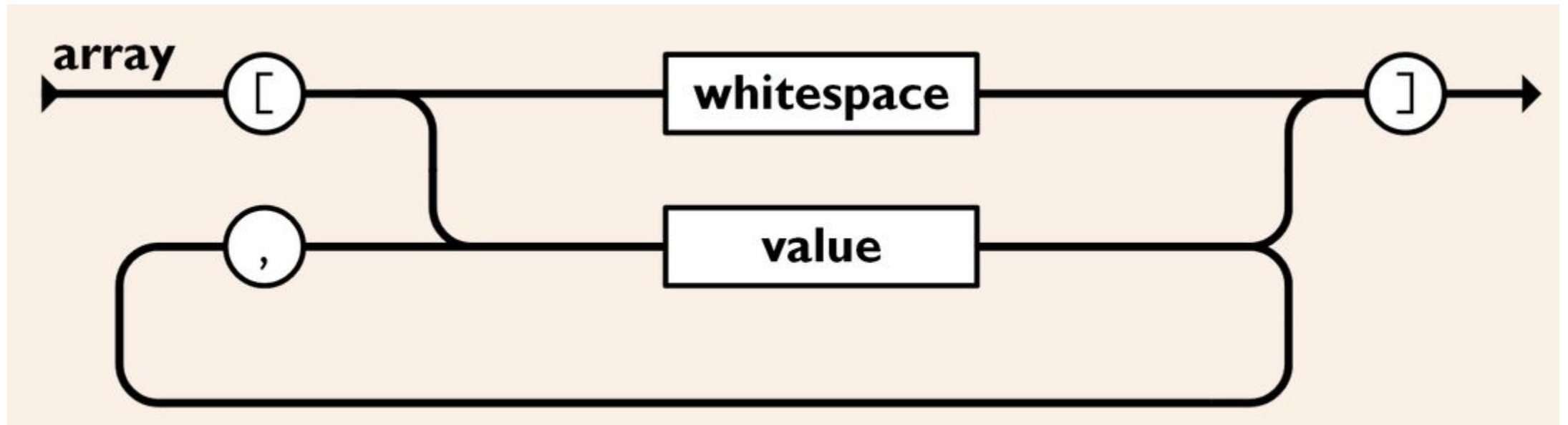
Reglas de sintaxis JSON

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con {llave de apertura y termine con }llave de cierre. Cada nombre es seguido por :dos puntos y los pares nombre/valor están separados por ,coma.



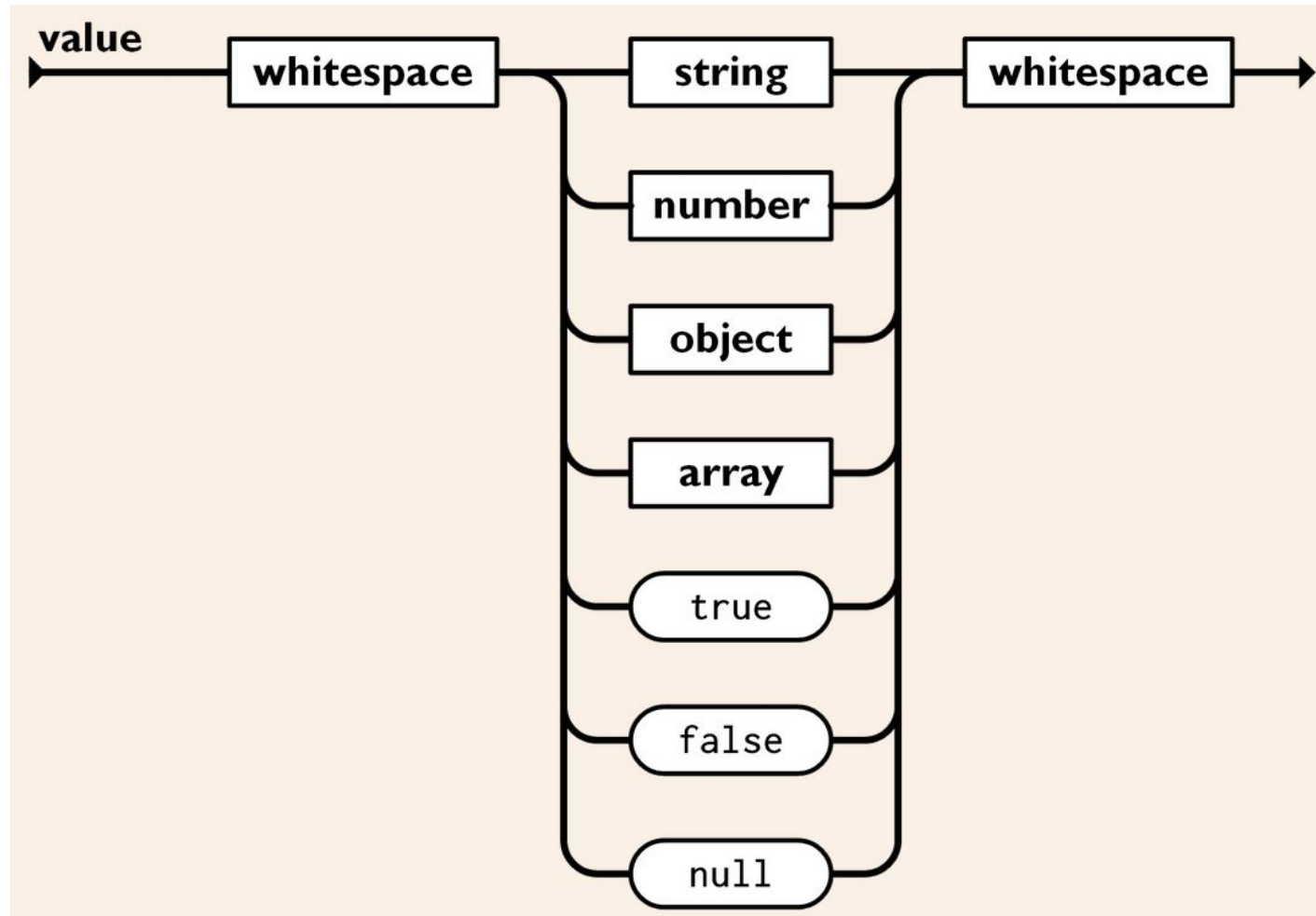
Reglas de sintaxis JSON

Un *arreglo* es una colección de valores. Un arreglo comienza con [corchete izquierdo y termina con] corchete derecho. Los valores se separan por , coma.



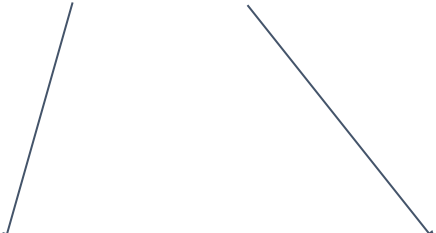
Reglas de sintaxis JSON

Un *valor* puede ser una *cadena de caracteres* con **comillas dobles**, o un *número*, o true o false o null, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse.



Reglas de sintaxis JSON

```
{  
  "libro" : [  
    {  
      "autor" : "Arturo Pérez Reverte",  
      "titulo" : "La tabla de Flandes",  
      "paginas" : 200  
    },  
    {  
      "autor" : "Arturo Pérez Reverte",  
      "titulo" : "El maestro de esgrima",  
      "paginas" : 300  
    }  
  ]  
}
```



NOMBRE

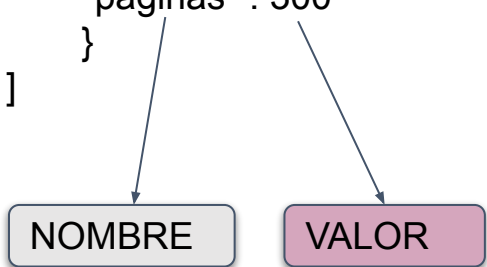
VALOR

Reglas de sintaxis JSON

Documentos JSON

- El tipo de archivo de los archivos JSON es ".json"
- El tipo MIME para el texto JSON es "application / json"

```
{  
  "libro": [  
    {  
      "autor" : "Arturo Pérez Reverte",  
      "titulo" : "La tabla de Flandes",  
      "paginas" : 200  
    },  
    {  
      "autor" : "Arturo Pérez Reverte",  
      "titulo" : "El maestro de esgrima",  
      "paginas" : 300  
    }  
  ]  
}
```



NOMBRE VALOR

Tipos de datos válidos

VALORES PERMITIDOS

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

- una cadena
- un número
- un objeto (objeto JSON)
- una matriz o array
- un booleano
- *nulo*

VALORES NO PERMITIDOS

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

- Una función
- una fecha
- *indefinido*

Tipos de datos válidos

CADENAS EN JSON

Las cadenas en JSON deben escribirse entre comillas dobles.

```
{ "nombre": "Pedro" }
```

NÚMEROS JSON

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

```
{ "edad": 30 }
```

MATRICES JSON

Los valores en JSON pueden ser matrices.

```
{  
  "empleados": [ "Pedro", "Luis", "Ana" ]  
}
```

OBJETOS JSON

Los valores en JSON pueden ser objetos.

```
{  
  "empleado": { "nombre": "Pedro", "edad": 30, "ciudad": "Valencia" }  
}
```

BOOLEANOS EN JSON

Los valores en JSON pueden ser verdadero / falso.

```
{ "vendido": true }
```

JSON NULO

Los valores en JSON pueden ser nulos.

```
{ "software": null }
```

Ejercicio

EJEMPLO XML

```
<empleados>
  <empleado>
    <nombre>Pedro</nombre>
    <apellido>Sanchis</apellido>
  </empleado>
  <empleado>
    <nombre>Ana</nombre>
    <apellido>Lopez</apellido>
  </empleado>
  <empleado>
    <nombre>Juan</nombre>
    <apellido>Vidal</apellido>
  </empleado>
</empleados>
```

Convierte este archivo XML a Json utilizando las estructuras permitidas.

JSON frente a XML

EJEMPLO JSON

```
{ "empleados": [
  { "nombre": "Pedro", "apellido": "Sanchis" },
  { "nombre": "Ana", "apellido": "Lopez" },
  { "nombre": "Juan", "apellido": "Vidal" }
]}
```

EJEMPLO XML

```
<empleados>
  <empleado>
    <nombre>Pedro</nombre>
    <apellido>Sanchis</apellido>
  </empleado>
  <empleado>
    <nombre>Ana</nombre>
    <apellido>Lopez</apellido>
  </empleado>
  <empleado>
    <nombre>Juan</nombre>
    <apellido>Vidal</apellido>
  </empleado>
</empleados>
```

JSON frente a XML

SEMEJANZAS

JSON es como XML porque:

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

DIFERENCIAS

JSON es diferente a XML porque:

- JSON no usa etiqueta de cierre
- JSON es más corto
- JSON es más rápido de leer y escribir
- JSON puede usar matrices
- Cadenas de caracteres entre comillas dobles

La mayor diferencia es:

XML debe analizarse con un analizador XML. JSON se puede analizar mediante una función estándar de JavaScript (función eval()).

¿Por qué el archivo JSON cada vez es más utilizado?

La simplicidad del formato JSON es una de las principales razones por las que es bastante utilizado.

Eso porque las [peticiones AJAX](#), que permiten la actualización de la página sin la necesidad de recargarla completamente, deben ser ejecutadas con mucha rapidez para que esas actualizaciones sean transparentes para el usuario.

Por ser liviano y compacto, el formato JSON atiende a esa necesidad. Por lo tanto, los datos pueden transferirse de forma rápida e interpretarse con facilidad por la aplicación.

Cabe mencionar que el formato XML también se puede utilizar en peticiones AJAX. Sin embargo, es un archivo más grande, por contener más información en lo que se refiere al gran número de tags de apertura y cierre. Algo que torna su transferencia y procesamiento más lentos que el modelo JSON.

Acceder a los elementos de un array

The screenshot displays a web browser window with the address bar showing `https://mdn.github.io/learning-area/javascript/ojs/json/JSONTest.html`. The browser's developer tools are open, showing the console with four log entries. The first log entry is `>> superHeroes.squadName` followed by `<- "Super hero squad"`. The second log entry is `>> superHeroes.homeTown` followed by `<- "Metro City"`. The third log entry is `>> superHeroes.members[0].name` followed by `<- "Molecule Man"`. The fourth log entry is `>> superHeroes["members"][0].name` followed by `<- "Molecule Man"`. A large grey arrow points from the first log entry to the second. The browser's developer tools are open, showing the console with four log entries. The first log entry is `>> superHeroes.squadName` followed by `<- "Super hero squad"`. The second log entry is `>> superHeroes.homeTown` followed by `<- "Metro City"`. The third log entry is `>> superHeroes.members[0].name` followed by `<- "Molecule Man"`. The fourth log entry is `>> superHeroes["members"][0].name` followed by `<- "Molecule Man"`. A large grey arrow points from the first log entry to the second.

Función Parse

ANÁLISIS

- 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 convertirán en un objeto JavaScript.

Recepción desde un servidor web

```
'{ "nombre": "Pedro", "edad": 30, "ciudad": "Valencia" }'
```

Conversión a objeto javascript

```
var obj = JSON.parse('{ "nombre": "Pedro", "edad": 30, "ciudad": "Valencia" }');
```

```
<!DOCTYPE html>
<html>
<body>
  <h2>Crear un objeto desde un texto en JSON</h2>
  <p id="demo"></p>
<script>
var txt = '{"nombre": "Pedro", "edad": 30, "ciudad": "Valencia"}'
//obtenemos un objeto Javascript
var obj = JSON.parse(txt);
//Colocamos en el párrafo con id demo el nombre y la edad
document.getElementById("demo").innerHTML = obj.nombre
+ ", " + obj.edad; </script>
</body>
</html>
```



Crear un objeto desde un texto en JSON

Pedro, 30

Función Parse

OBTENER JSON DESDE EL 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.
- Para obtener datos desde el servidor es necesario usar XMLHttpRequest

FICHERO JSON LEÍDO

```
{
  "nombre": "Pedro",
  "edad": 30,
  "mascotas": [
    { "animal": "perro", "nombre": "Fido" },
    { "animal": "gato", "nombre": "Felix" },
    { "animal": "hamster", "nombre": "Lightning" }
  ]
}
```

```
<!DOCTYPE html>
<html>
<body>
  <h2>Utilice XMLHttpRequest para obtener el contenido de un archivo.
</h2>
  <p>El contenido está escrito en formato JSON y se puede convertir
  fácilmente en un objeto JavaScript.</p>
  <p id="demo"></p>
  <script>
    var xmlhttp = new XMLHttpRequest();
    //Esta función inicializa una nueva request
    xmlhttp.open("GET", "json_demo.txt", true);
    xmlhttp.send();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        var miObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = miObj.nombre;
      }
    };
  </script>
</body>
</html>
```

Función Parse

ESTADOS DE LA XMLHttpRequest

Value	State	Description
0	UNSENT	Client has been created. <code>open()</code> not called yet.
1	OPENED	<code>open()</code> has been called.
2	HEADERS_RECEIVED	<code>send()</code> has been called, and headers and status are available.
3	LOADING	Downloading; <code>responseText</code> holds partial data.
4	DONE	The operation is complete.

ESTADOS DE LA RESPUESTA HTTP

1. Informational responses (100 – 199)
2. Successful responses (200 – 299)
3. Redirects (300 – 399)
4. Client errors (400 – 499)
5. Server errors (500 – 599)

Función stringify

JSON A CADENA PARA MANDARLA AL SERVIDOR

- 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 JavaScript en una cadena con `JSON.stringify()`.

Envío a un servidor web

```
var obj = { nombre: "Pedro", edad: 30, ciudad: "Valencia" };
```

Conversión de objeto a cadena

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

```
<!DOCTYPE                                html>
<html>
<body>
  <h2>Creación de una cadena JSON desde un objeto
  Javascript.</h2>
  <p                                     id="demo"></p>

  <script>
var obj = { nombre: "Pedro", edad: 30, ciudad: "Valencia" };
//Transformamos                          a                          cadena
var      miJSON                        =      JSON.stringify(obj);
document.getElementById("demo").innerHTML = miJSON ;
</script>
</body>
</html>
```



Creación de una cadena JSON desde un objeto Javascript.

```
{"nombre":"Pedro","edad":30,"ciudad":"Valencia"}
```

Función stringify

MATRIZ JSON A CADENA PARA MANDARLA AL SERVIDOR

- 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 JavaScript en una cadena con `JSON.stringify()`.

Envío a un servidor web

```
var array = [ "Pedro", "Juan", "Ana", "Luis" ];
```

Conversión de matriz a cadena

```
var miJSON = JSON.stringify(array);
```

```
<!DOCTYPE                                html>
<html>
<body>
  <h2>Creación de una cadena JSON desde un objeto
  Javascript.</h2>
  <p                                     id="demo"></p>

  <script>
var array = [ "Pedro", "Juan", "Ana", "Luis" ];
var      miJSON      =      JSON.stringify(array);
document.getElementById("demo").innerHTML = miJSON;
</script>
</body>
</html>
```

Objetos JSON

Las características de los objetos JSON son:

- Los objetos JSON están rodeados por llaves {}.
- Los objetos JSON se escriben 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.

```
{ "nombre": "Pedro", "edad": 30, "coche": null }
```

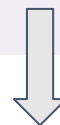
ACCESO A LOS VALORES JSON EN JAVASCRIPT

```
miObj = { "nombre": "Pedro", "edad": 30, "coche": null };  
x = miObj.nombre;
```

```
miObj = { "nombre": "Pedro", "edad": 30, "coche": null };  
x = miObj["nombre"];
```

RECORRER LAS PROPIEDADES EN JAVASCRIPT

```
miObj = { "nombre": "Pedro", "edad": 30, "coche": null };  
for (dato in miObj) {  
    document.getElementById("demo").innerHTML += dato;  
}
```



nombre

edad

coche

Objetos JSON

Las características de los objetos JSON son:

- Los objetos JSON están rodeados por llaves {}.
- Los objetos JSON se escriben 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.

```
{ "nombre": "Pedro", "edad": 30, "coche": null }
```

RECORRER LOS VALORES EN JAVASCRIPT

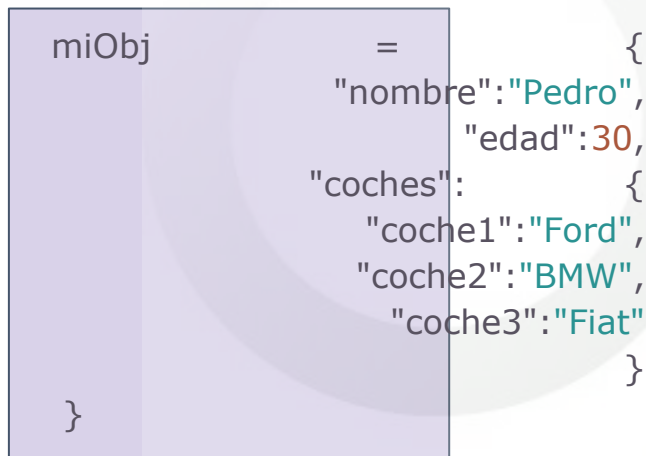
```
miObj = { "nombre": "Pedro", "edad": 30, "coche": null };  
for (dato in miObj) {  
    document.getElementById("demo").innerHTML += miObj[dato];  
}
```



Pedro
30
null

Objetos JSON

- Los valores de un objeto JSON pueden ser otro objeto JSON.



```
miObj = {  
  "nombre": "Pedro",  
  "edad": 30,  
  "coches": [  
    {  
      "coche1": "Ford",  
      "coche2": "BMW",  
      "coche3": "Fiat"  
    }  
  ]  
}
```

ACCESO A LOS VALORES EN JAVASCRIPT

```
x = miObj.coches.coche2;  
// o:  
x = miObj.coches["coche2"];
```

Matrices como objetos JSON

- Las matrices en JSON son casi las mismas que las matrices en JavaScript.
- En JSON, los valores de la 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 de JavaScript válida, incluidas funciones, fechas e *indefinidas*.

```
{  
  "nombre": "Pedro",  
  "edad": 30,  
  "coches": [ "Ford", "BMW", "Fiat" ]  
}
```

ACCEDER A LOS VALORES DE LA MATRIZ EN JAVASCRIPT

```
x = miObj.coches[0];
```

RECORRER LOS VALORES EN JAVASCRIPT

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

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

Matrices como objetos JSON

- Los valores de una matriz también pueden ser otra matriz o incluso otro objeto JSON:

```
miObj = {  
  "nombre": "Pedro",  
  "edad": 30,  
  "coches": [  
    { "nombre": "Ford", "modelos": [ "Fiesta", "Focus", "Mustang" ] },  
    { "nombre": "BMW", "modelos": [ "320", "X3", "X5" ] },  
    { "nombre": "Fiat", "modelos": [ "500", "Panda" ] }  
  ]  
}
```

ACCEDER A LAS MATRICES DENTRO DE OTRAS MATRICES

```
for (i in miObj.coches) {  
  x += "<h1>" + miObj.coches[i].nombre + "</h1>";  
  for (j in miObj.coches[i].modelos) {  
    x += miObj.coches[i].modelos[j];  
  }  
}
```

MODIFICAR LOS DATOS DE LA MATRIZ

```
miObj.coches[1] = "Mercedes";
```

ELIMINAR DATOS DE LA MATRIZ

```
delete miObj.coches[1];
```

Enlaces de interés

1. [XMLHttpRequest - Web APIs](#)
2. [W3 Schools JSON Introduction](#)

Bibliografía

Prácticamente todo contenido del documento ha sido extraído de https://www.w3schools.com/js/js_json_intro.asp

