

JSON

JavaScript Object Notation

Enrique Barra

INTRODUCCIÓN – EL PROBLEMA

- Los desarrolladores necesitan enviar y recibir datos de manera sencilla pero utilizando un formato común para estructuras complejas.
- Se han desarrollado muchas soluciones ad-hoc donde se separan un conjunto de valores separados por comas, puntos y comas u otros separadores pero de serialización y deserialización complicadas.
- Hay que evitar tener que construir parsers (parsear es analizar, diseccionar) cada vez que queremos intercambiar mensajes con el servidor.
- XML es opción válida pero no la más adecuada por ser demasiada pesada.

INTRODUCCIÓN – UNA SOLUCIÓN

JSON (**J**ava**S**cript **O**bject **N**otation)

- Formato ligero de representación y almacenamiento de datos independiente de cualquier lenguaje de programación
- Tiene forma de texto plano, de simple de lectura, escritura y generación
- No es necesario que se construyan parsers personalizados
- Sirve para **representar** objetos en el lado de cliente, normalmente en aplicaciones RIA (Rich Internet Application) que utilizan JavaScript
- Sirve para **intercambiar y almacenar** datos

INTRODUCCIÓN – JSON

- JSON :
 - Independiente de un lenguaje específico
 - Basado en texto
 - De Formato ligero
 - Fácil de parsear
 - NO define funciones
 - NO tiene estructuras invisibles
 - NO tiene espacios de nombres (Namespaces)
 - NO tiene validador
 - NO es extensible
- Su tipo MIME es -> **application/json**
- Existe **BSON** -> Binay JSON

INTRODUCCIÓN – JSON

- Lenguajes que lo soportan:
 - ActionScript
 - C / C++
 - .NET (C#, VB.NET...)
 - Delphi
 - Java
 - JavaScript
 - Perl
 - PHP
 - Python
 - Ruby
 - Etc...

FORMAS DE REPRESENTACION

- **String.**- Colección de cero o más caracteres unicode
- **Number.**- Valor numérico sin comillas
- **Object.**- Conjunto desordenado de pares nombre/valor
- **Array.**- Colección ordenada de valores
- **Value.**- Puede ser un string, número, booleano, objeto u array

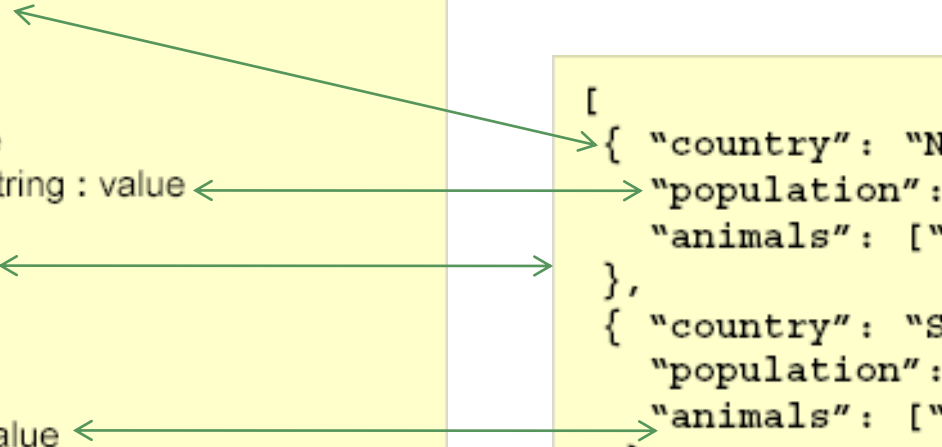
FORMAS DE REPRESENTACION

Descripción simplificada

```
object ::  
  { members }  
  {}  
members ::  
  string : value  
  members , string : value  
array ::  
  [ elements ]  
  []  
elements ::  
  value  
  elements , value  
value ::  
  string  
  number  
  object  
  array  
  true  
  false  
  null
```

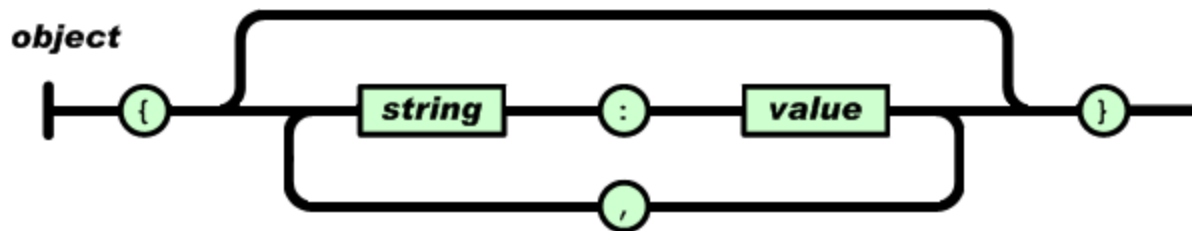
Ejemplo

```
[  
  { "country": "New Zealand",  
    "population": 3993817,  
    "animals": ["sheep", "kiwi"]  
  },  
  { "country": "Singapore",  
    "population": 4353893,  
    "animals": ["merlion", "tiger"]  
  }  
]
```



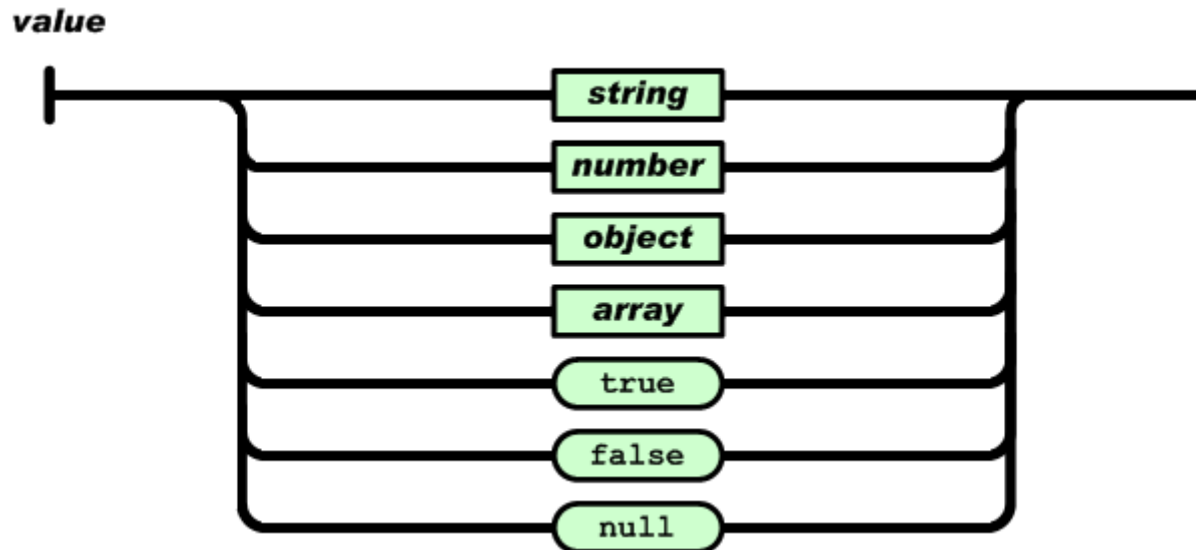
FORMA DE OBJECT / CLASE

- Es un conjunto de propiedades , cada una con su valor
- Notación
 - Empieza con una llave de apertura {
 - Terminan con una llave de cierre }
 - Sus propiedades
 - Se separan con comas
 - El nombre y el valor estan separados por dos puntos :



FORMA DE VALUE

- Puede ser
 - Una cadena de caracteres con comillas dobles
 - Un número
 - True, false, null
 - Un objeto
 - Un array

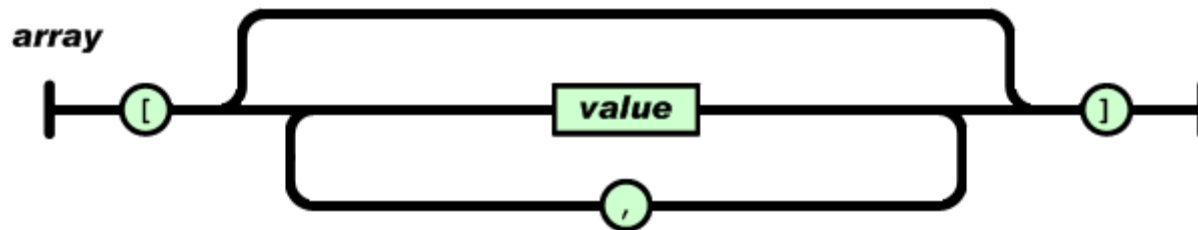


FORMA DE OBJECT / CLASE

```
[  
  {  
    "country": "New Zealand",  
    "population": 3993817,  
    "animals": ["sheep", "kiwi"]  
  },  
  {  
    "country": "Singapore",  
    "population": 4353893,  
    "animals": ["merlion", "tiger"]  
  }  
]
```

FORMA DE ARRAY

- Colección ordenada de valores u objetos
- Notación
 - Empieza con un corchete izquierdo [
 - Termina con un corchete derecho]
 - Los valores se separan con una coma ,



FORMA DE ARRAY

[

{

“country”: “New Zealand”,

“population”: 3993817,

“animals”: [“sheep”, “kiwi”]

}

,

{

“country”: “Singapore”,

“population”: 4353893,

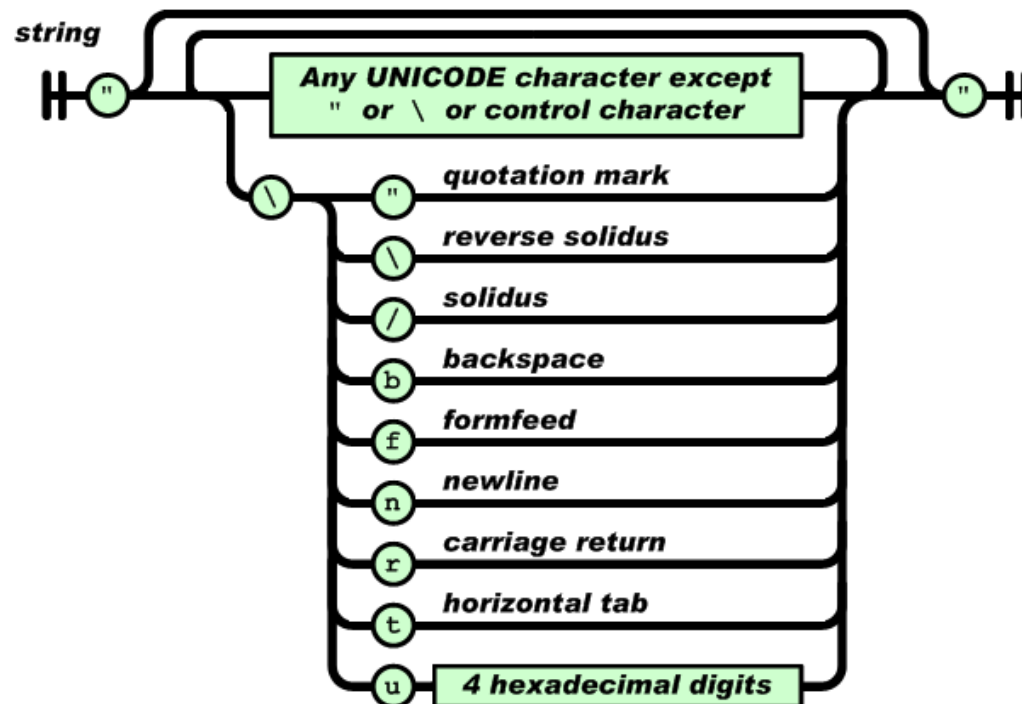
“animals”: [“merlion”, “tiger”]

}

]

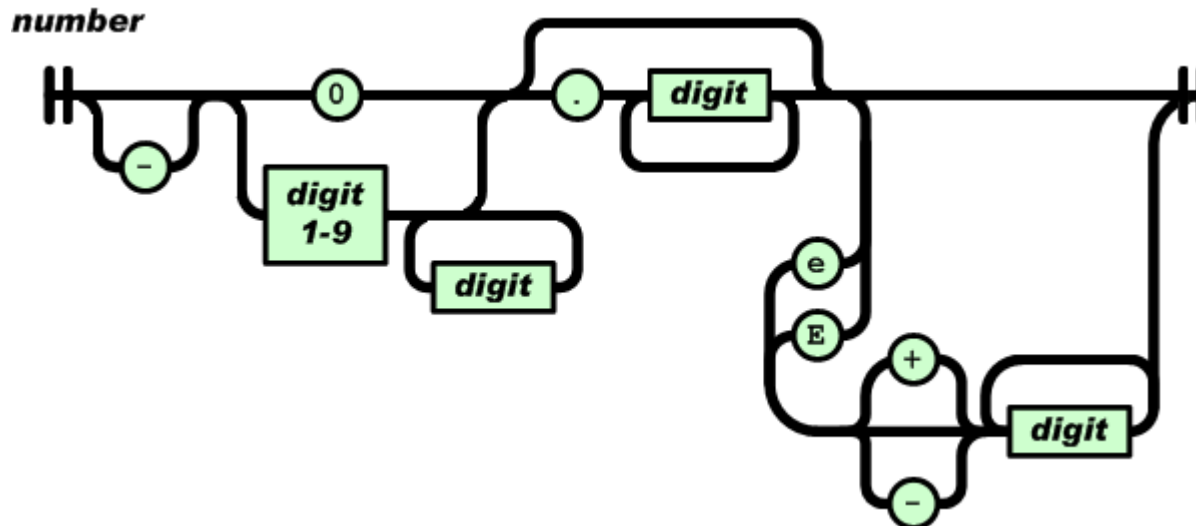
FORMA DE STRING

- Colección de cero a más caracteres Unicode encerrados entre comillas dobles
- Los caracteres de escape utilizan la barra invertida
- Es parecida a una cadena de caracteres en C o Java.



FORMA DE NUMBER

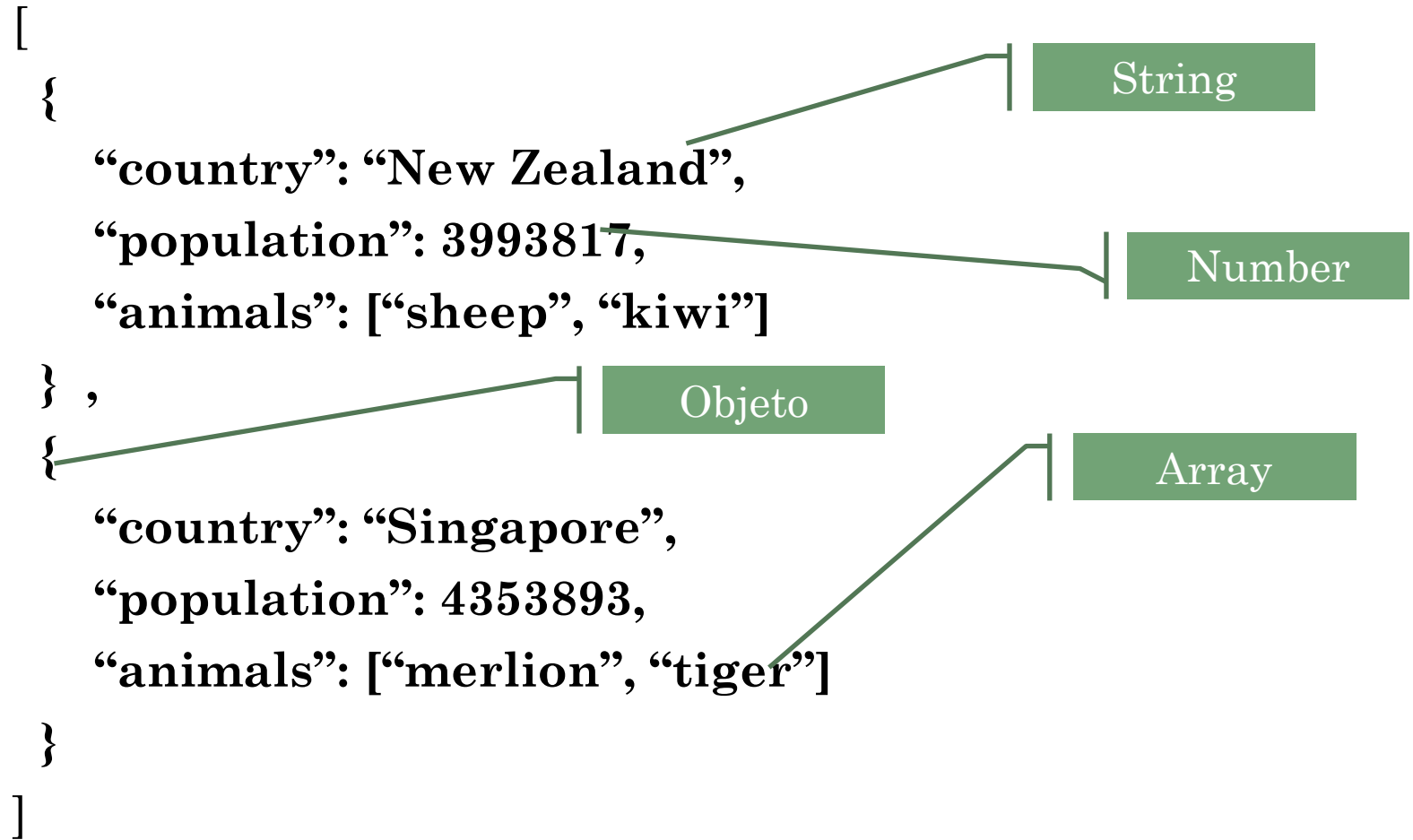
- Similar a los numeros de C o Java
- No usa formato octal o hexadecimal
- No puede ser **NaN** o **Infinity**, en su lugar se usa **null**.
- Puede representar
 - Integer
 - Real
 - Scientific



CODIFICACIÓN DE CARACTERES

- Estrictamente UNICODE
- Por defecto es UTF-8
- UTF-16 y UTF-32 también están permitidos.

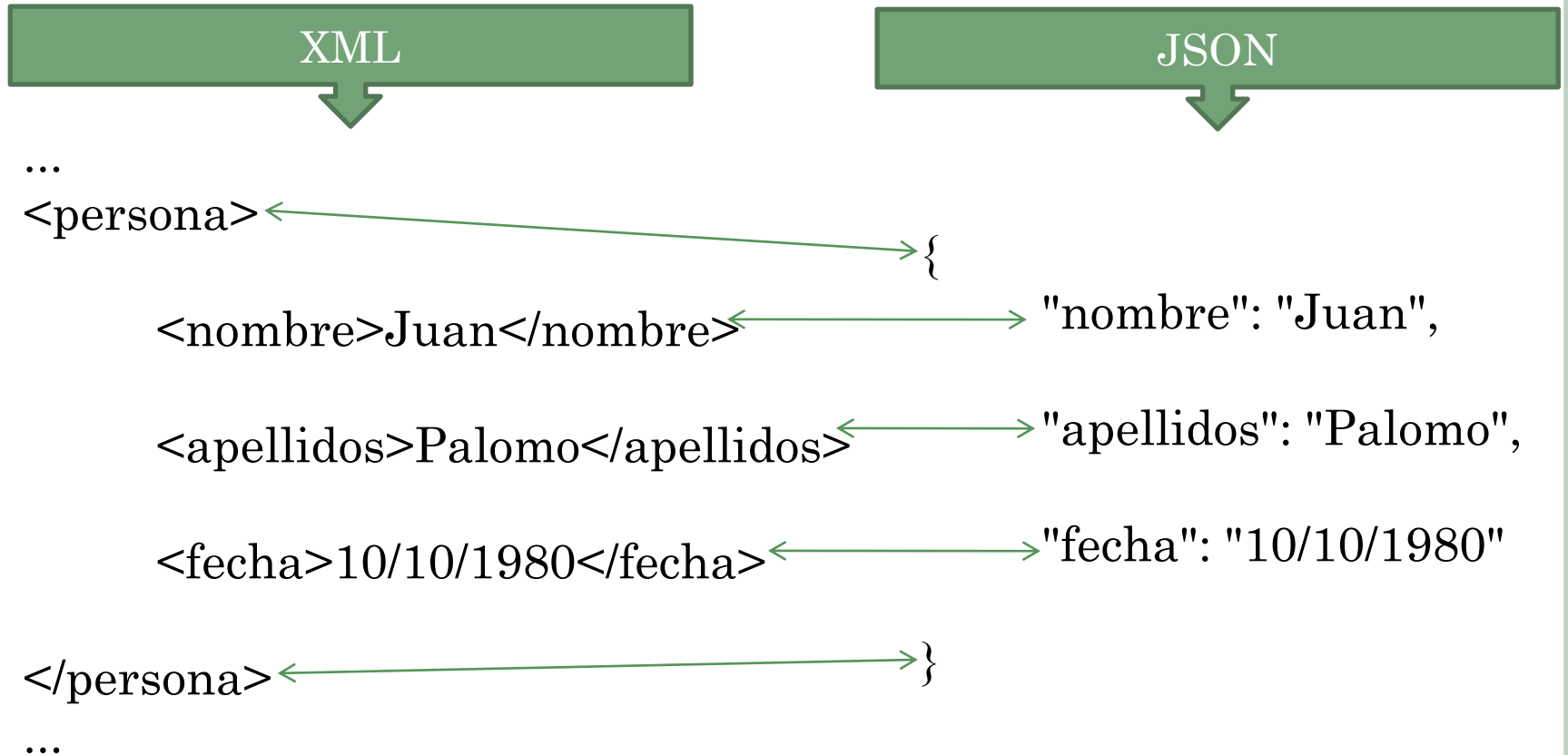
FORMA DE VALUE



OTRAS FORMAS

- Boolean -> true / false
- null

JSON VS XML (CLASE)



JSON VS XML (ARRAYS)

XML

JSON

```
...  
<listado>  
  <persona>  
    <nombre>Juan</nombre>  
    <apellidos>Palomo</apellidos>  
    <fecha>10/10/1980</fecha>  
  </persona>  
  <persona>  
    <nombre>Juan</nombre>  
    <apellidos>Palomo</apellidos>  
    <fecha>10/10/1980</fecha>  
  </persona>  
</listado>  
...
```

```
...  
{"listado": [  
  {  
    "nombre": "Juan",  
    "apellidos": "Palomo",  
    "fecha": "10/10/1980"  
  },  
  {  
    "nombre": "Juan",  
    "apellidos": "Palomo",  
    "fecha": "10/10/1980"  
  }  
]}  
...
```

JSON VS XML (SIMILITUDES)

- Ambos son legibles por los humanos
- Tienen una sintaxis muy simple
- Son jerárquicos
- Son independientes del lenguaje de programación

JSON VS XML (DIFERENCIAS)

- Sintáxis diferente

- JSON

- Es más compacto (ocupa menos que el XML)
- Puede ser parseado usando el método **eval()** de JavaScript
- Puede incluir Arrays
- Los nombres de las propiedades no pueden ser palabras reservadas de JavaScript

- XML

- Los nombres son mas extensos
- Puede ser validado bajo un conjunto de reglas

YAML

- Acrónimo inicialmente de: **Y**et **A**nother **M**arkup **L**anguage
- Cambiado a: **Y**AML **A**in't **M**arkup **L**anguage
- Es un subconjunto de JSON, con más capacidades
 - Listas, casting, etc
 - No maneja caracteres Unicode de escape
 - JSON puede ser parseado por los parsers de YAML
- Hay que tenerlo en cuenta cuando JSON no sea suficiente para nuestras necesidades.



JSON- USO

23

JSON - UTILIZACIÓN

- **Serialización:** Transformación de objetos a cadenas de texto
- **Deserialización:** Transformación de cadenas de texto a objetos
- Directamente con JavaScript
- Mediante Librerías (en cualquier lenguaje)
- Frameworks de cliente
- NoSQL
- package.json

JSON – JAVASCRIPT

- Define los siguientes métodos
 - JSON.parse
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse
 - JSON.stringify
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify

EJEMPLOS JSON.PARSE()

```
JSON.parse('{}');           // {}
JSON.parse('true');         // true
JSON.parse('"foo"');        // "foo"
JSON.parse('[1, 5, "false"]'); // [1, 5, "false"]
JSON.parse('null');         // null
```

```
var obj = JSON.parse('{ "employees" : [' +
    '{ "firstName":"John" , "lastName":"Doe" },' +
    '{ "firstName":"Anna" , "lastName":"Smith" },' +
    '{ "firstName":"Peter" , "lastName":"Jones" } ]}');
```

```
var obj = JSON.parse('{ "employees" : [{ "firstName":"John"
, "lastName":"Doe" },{ "firstName":"Anna" ,
"lastName":"Smith" },{ "firstName":"Peter" ,
"lastName":"Jones" } ]}');
```

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

EJEMPLOS JSON.STRINGIFY()

```
JSON.stringify({});           // '{}'  
JSON.stringify(true);        // 'true'  
JSON.stringify('foo');       // '"foo"'  
JSON.stringify([1, 'false', false]); // '[1,"false",false]'  
JSON.stringify({ x: 5 });     // '{"x":5}'
```

```
JSON.stringify(new Date(2006, 0, 2, 15, 4, 5))  
// '"2006-01-02T15:04:05.000Z"'
```

```
JSON.stringify({ x: 5, y: 6 });  
// '{"x":5,"y":6}' o '{"y":6,"x":5}'
```

GSON

- Librería para convertir objetos Java a JSON y vice-versa.
 - <http://sites.google.com/site/gson/Home>
 - <http://code.google.com/p/google-gson>
- Objetivos
 - Proporcionar mecanismos sencillos para convertir los objetos
 - Dar capacidad de utilizar representaciones personalizadas de objetos.

GSON - EJEMPLO


Clase personalizada

```
class BagOfPrimitives {  
    private int value1 = 1;  
    private String value2 = "abc";  
    private transient int value3 = 3;  
}
```

Serialización

```
BagOfPrimitives obj = new BagOfPrimitives();  
Gson gson = new Gson();  
String json = gson.toJson(obj);  
==> {"value1":1,"value2":"abc"}
```

Deserialización



```
BagOfPrimitives obj2 = gson.fromJson(json, BagOfPrimitives.class);  
==> obj2 is just like obj
```

FRAMEWORKS CLIENTE

- Actualmente existen frameworks que utilizan de forma nativa JSON para presentar y tratar la información proveniente del servidor.
 - **YUI (Yahoo User Interface)**
 - Dojo
 - **jQuery**
 - **Extjs**
 - Otros toolkits Ajax.

JQUERY

- Puede recuperar datos en formato JSON
- API
 - `jQuery.parseJSON(json)`
 - <http://api.jquery.com/jquery.parsejson/>
 - `jQuery.getJSON(url, [data], [callback(data, textStatus, xhr)])`
 - <http://api.jquery.com/jquery.getjson/>

NoSQL

- Este término se refiere a bases de datos “no relacionales” que no dan garantías ACID, su característica que más llama la atención es que no existen esquemas de tablas predefinidos.
- Algunas de las bases de datos que exponen sus datos mediante JSON/BSON son:
 - CouchDB
 - MongoDB
 - RavenDB
 - Riak
 - Keyspace
 - Pincaster
 - Sones

PACKAGE.JSON

- Fichero de configuración de un proyecto nodejs. Usado por el “node package manager” o npm
- Es formato JSON, e indica todo lo que hace falta saber para un proyecto, nombre, versión, descripción, dependencias, repositorios, bugs, etc
- Info completa: <https://docs.npmjs.com/files/package.json>
- Ejemplo:

```
{ "name": "ethopia-waza",  
  "description": "a delightfully fruity coffee varietal",  
  "version": "1.2.3",  
  "devDependencies": {  
    "coffee-script": "~1.6.3"  
  },  
  "scripts": {  
    "prepublish": "coffee -o lib/ -c src/waza.coffee"  
  },  
  "main": "lib/waza.js"  
}
```



AUTOEVALUACIÓN

35

TESTS JSON

- Which of the following expressions are valid JSON documents for MongoDB?
 - Remember, MongoDB doesn't require quotation marks around keys, as they must always be strings.
- A. { a : 1, b : 2, c : 3 }
- B. { a,1; b, 4, c, 6 }
- C. { a : 1; b : 1; c : 4 }
- D. (A, 1; b : 2; c, 4 }

TESTS JSON

- Which of the following expressions are valid JSON documents for MongoDB?
 - A. `{ a : 1, b : 2, c : 3 }`
 - B. `{ a,1; b, 4, c, 6 }`
 - C. `{ a : 1; b : 1; c : 4 }`
 - D. `(A, 1; b : 2; c, 4)`

TESTS JSON

- Which of the following are JSON documents that the MongoDB shell will accept?
 - A. `{ a : 1, b : 2, c : 3 }`
 - B. `{ a : 1, b : 2, c : [1, 2, 3, 4, 5] }`
 - C. `{ a : 1, b : {}, c : [{ a : 1, b : 2 }, 5, 6] }`
 - D. `{ }`

TESTS JSON

- Which of the following are JSON documents that the MongoDB shell will accept?
 - A. `{ a : 1, b : 2, c : 3 }`
 - B. `{ a : 1, b : 2, c : [1, 2, 3, 4, 5] }`
 - C. `{ a : 1, b : {}, c : [{ a : 1, b : 2 }, 5, 6] }`
 - D. `{ }`

JSON EJERCICIOS

- Write the JSON for a simple document containing a single key "fruit" that has as its value an array containing three strings: "apple", "pear", and "peach"
- Write a JSON document with a single key, "address" that has as its value another document with the keys "street_address", "city", "state", "zipcode", with the following values:
"street_address" is "23 Elm Drive", "city" is "Palo Alto", "state" is "California", "zipcode" is "94305"

JSON EJERCICIOS

- `{"fruit" : ["apple", "pear", "peach"]}`
- `{ "address" : { "street_address" : "23 Elm Drive",
"city" : "Palo Alto", "state" : "California", "zipcode"
: "94305" } }`

A decorative graphic on the left side of the slide. It consists of several vertical lines of varying shades of green and grey. Overlaid on these lines are several circles of different sizes, also in shades of green. One large circle is positioned near the top, and several smaller circles are arranged below it, some overlapping the lines.

CONCLUSIONES

CONCLUSIONES

- Formato de intercambio de datos, potente, flexible y **sobre todo ligero** para intercambiar datos (por ejemplo vía HTTP) o almacenarlos.
- Independiente de cualquier lenguaje de programación.
- Es soportado por los principales lenguajes del lado servidor
 - Java, .Net, PHP (pueden serializar y deserializar objetos en formato JSON)
- Ideal para construir aplicaciones RIA con frameworks JavaScript
 - Ej.: jQuery
- Existen diferentes bases de datos **NoSQL** que guardan sus datos en formato JSON plano o binario (BSON)