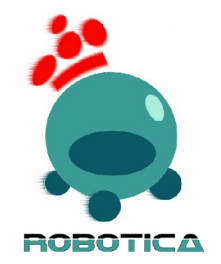
LABORATORIO TECNOLOGÍAS AUDIOVISUALES EN LA WEB

José María Cañas Plaza

jmplaza@gsyc.es



Grado Sistemas Audiovisuales y Multimedia
curso 2015-2016

JSON

Contenido

- Introducción
- Formas de representación
- Comparación con XML
- Utilización en JavaScript

Introducción

El problema: intercambio de datos en aplicaciones web

- Por ejemplo: AJAX, WebServices...
- Son necesarios formatos ligeros
- 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 pero de serialización y deserialización complicadas.
- Hay que evitar tener que construir *parsers* cada vez que queremos intercambiar mensajes con el servidor.
- XML es opción válida pero no la más adecuada por pesada.

Una posible solución: JSON

- JSON (JavaScript Object Notation)
- subconjunto de ECMAScript para objetos literales
- `https://www.json.org`
- Formato ligero de intercambio de datos
- Independiente de cualquier lenguaje de programación
- Basado en texto plano, de simple de lectura, escritura y generación
- Su tipo MIME es `application/json`
- Codificación de caracteres:
  - Estrictamente UNICODE
  - Por defecto es UTF-8
  - UTF-16 y UTF-32 también están permitidos

Características

- Ocupa menos espacio que el formato XML
- Fácil de *parsear*, no es necesario construir *parsers* personalizados
- NO define funciones
- NO tiene estructuras invisibles
- NO tiene espacios de nombres (Namespaces)
- NO tiene validador
- NO es extensible

Características

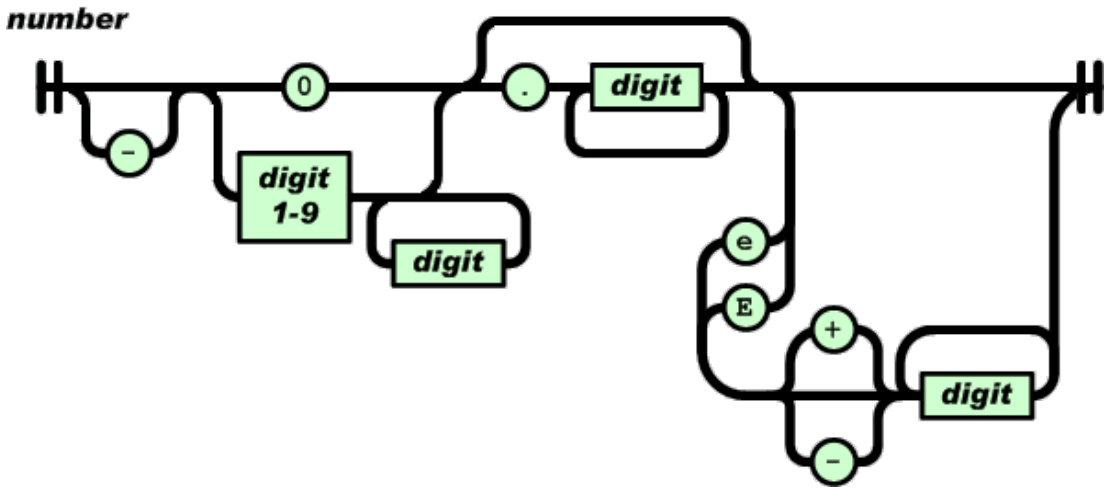
- Sirve para representar objetos en el lado de cliente, normalmente en aplicaciones RIA (Rich Internet Application) que utilizan JavaScript.
- Desde múltiples lenguajes se puede analizar-generar JSON
  - C / C++
  - .NET (C#, VB.NET...)
  - Java
  - JavaScript
  - Perl
  - PHP
  - Python
  - Ruby
  - ActionScript, Delphi...

Formas de representación

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

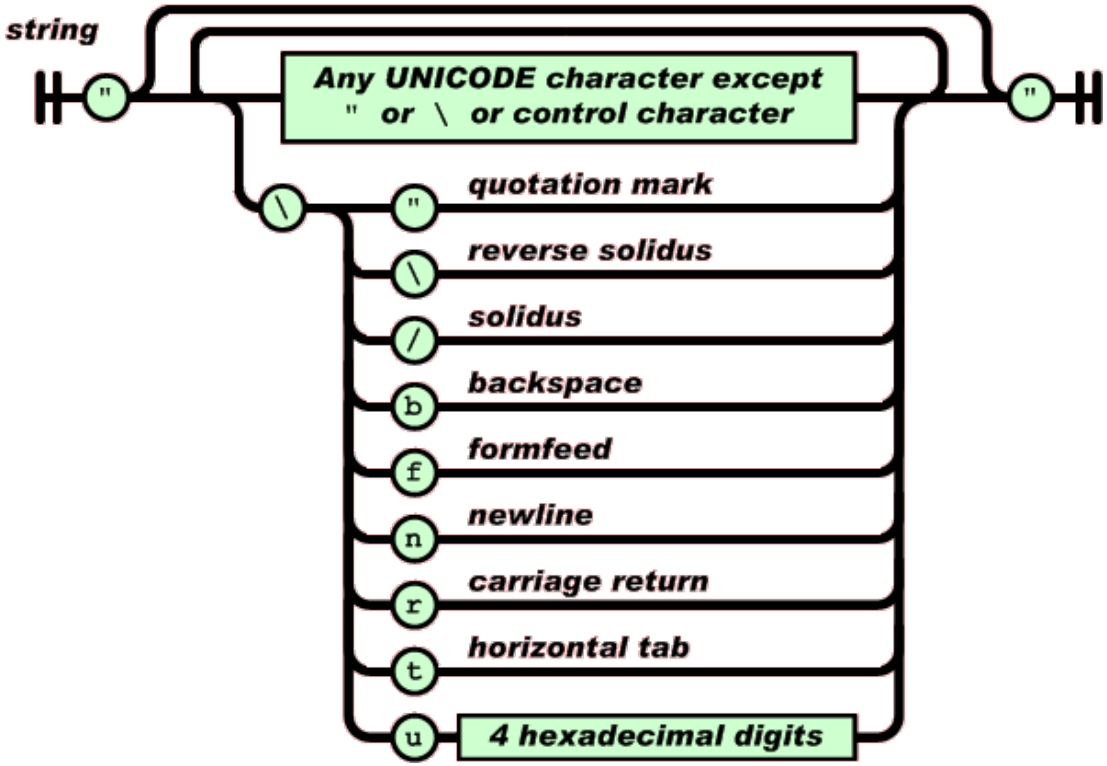
Forma Number

- 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



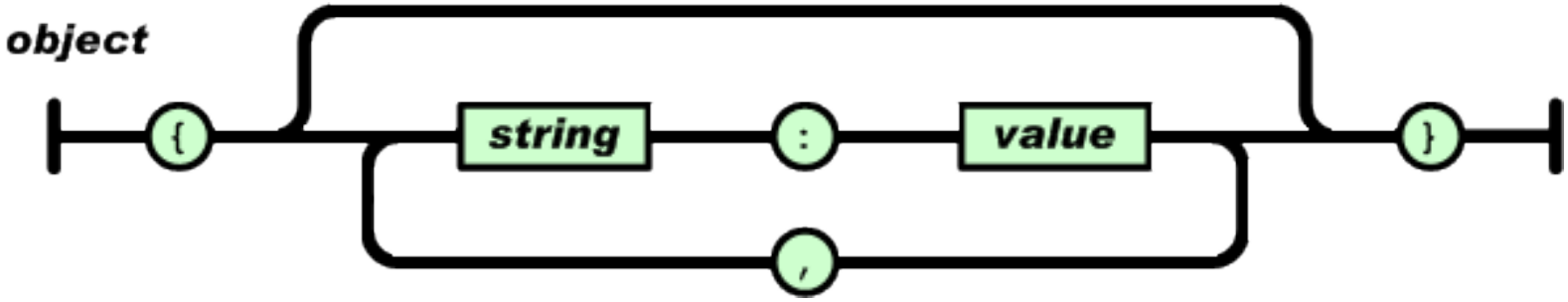
Forma String

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



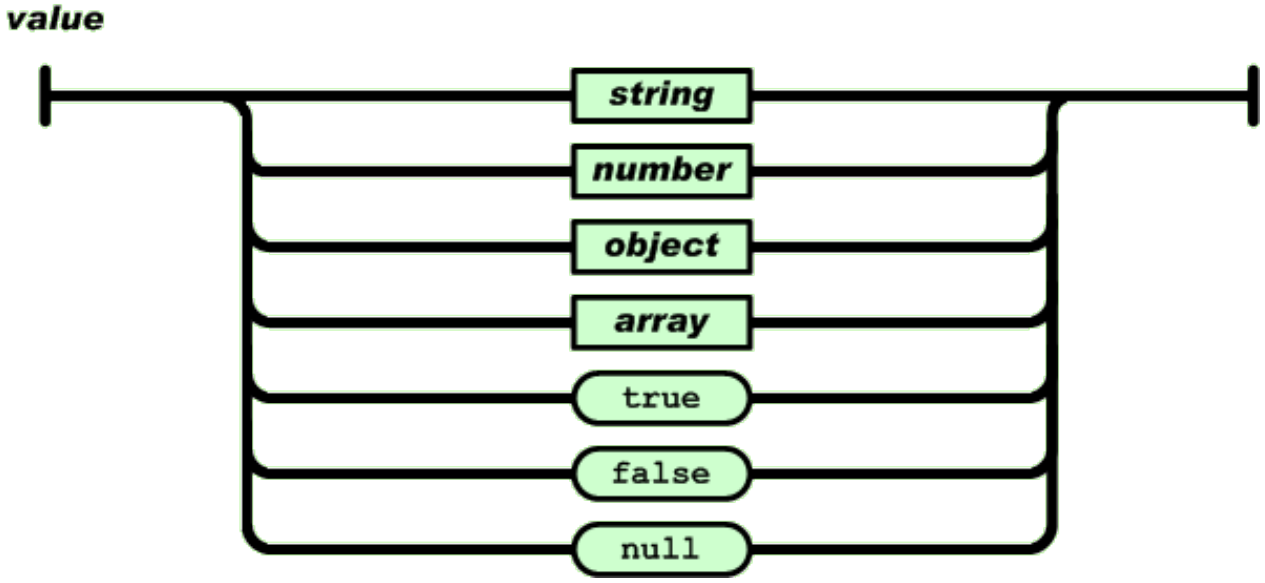
Forma Object

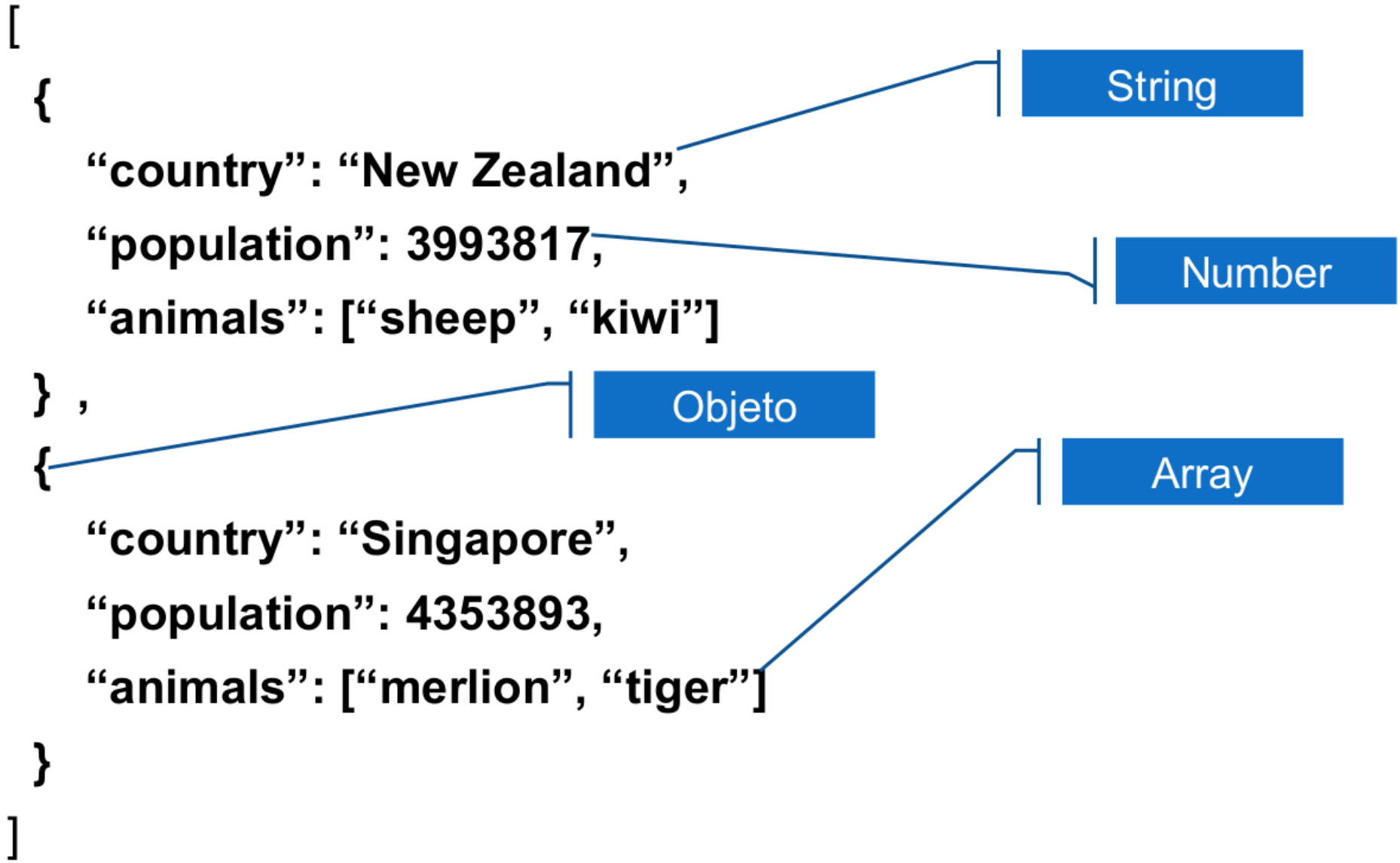
- Es un conjunto de propiedades, cada una con su valor
- Empieza con una llave de apertura { y terminan con una de cierre }
- Sus propiedades se separan con comas
- En las propiedades el nombre y el valor estan separados por dos puntos



Forma Value

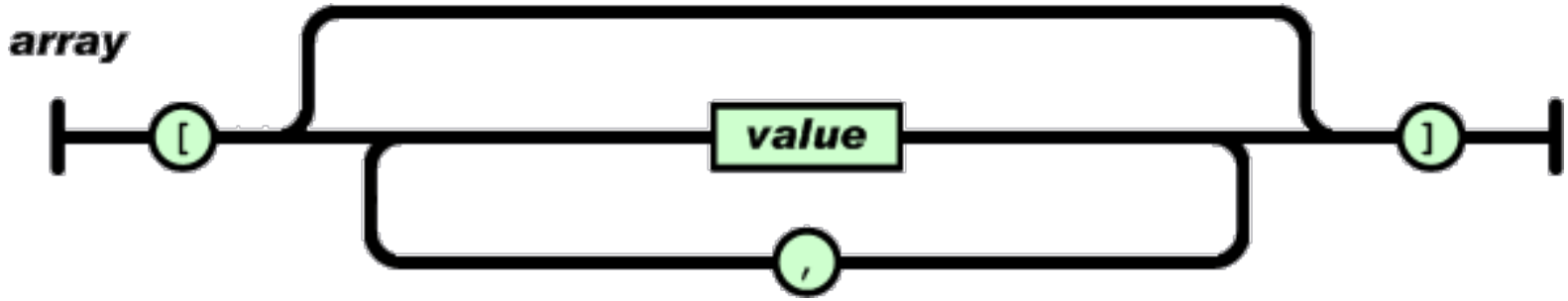
- Una cadena de caracteres con comillas dobles
- Un número
- `true`, `false`, `null`
- Un objeto
- Un array

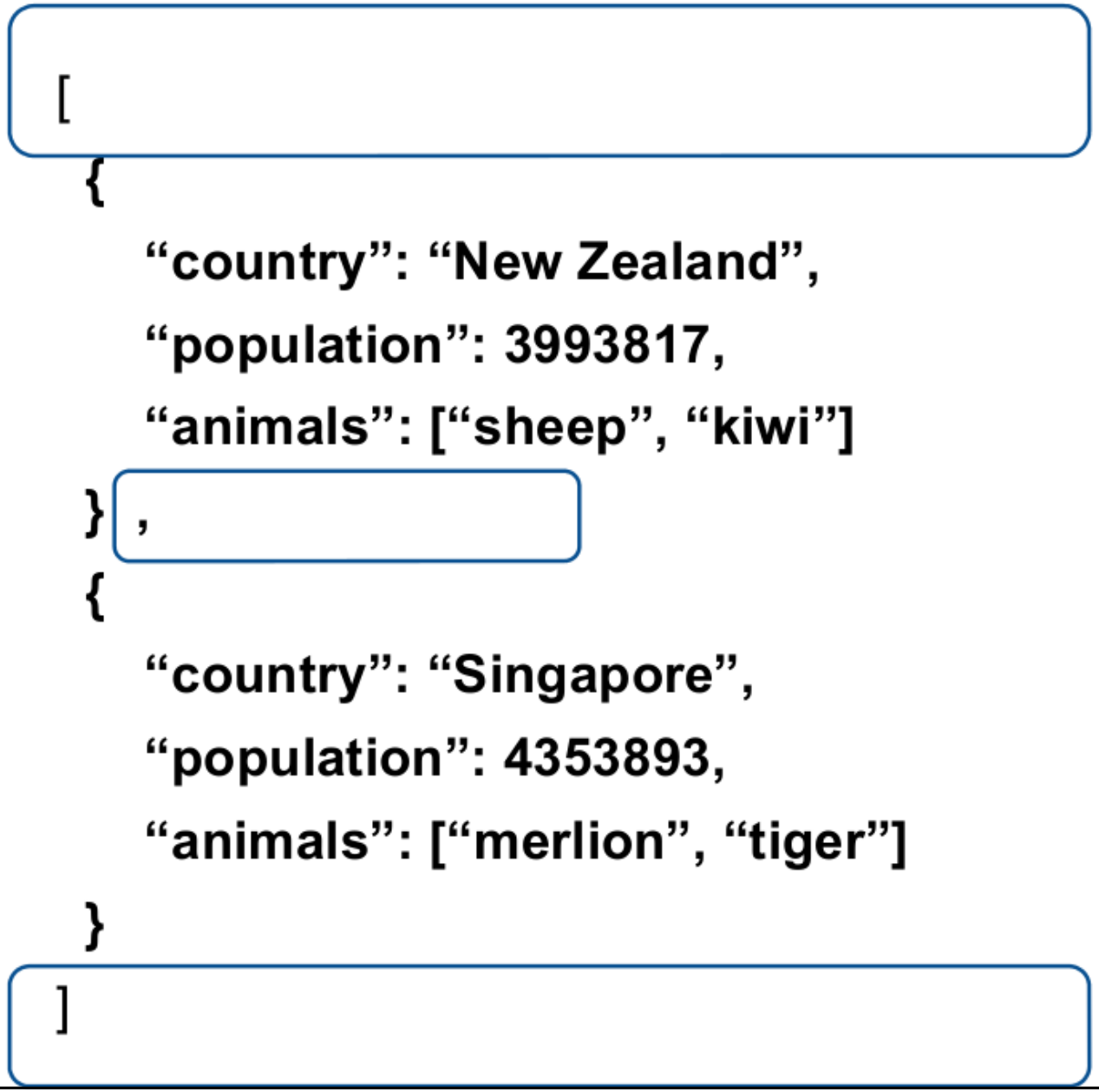



[
{
"country": "New Zealand",
"population": 3993817,
"animals": ["sheep", "kiwi"]
} ,
{
"country": "Singapore",
"population": 4353893,
"animals": ["merlion", "tiger"]
}
]
String
Number
Objeto
Array

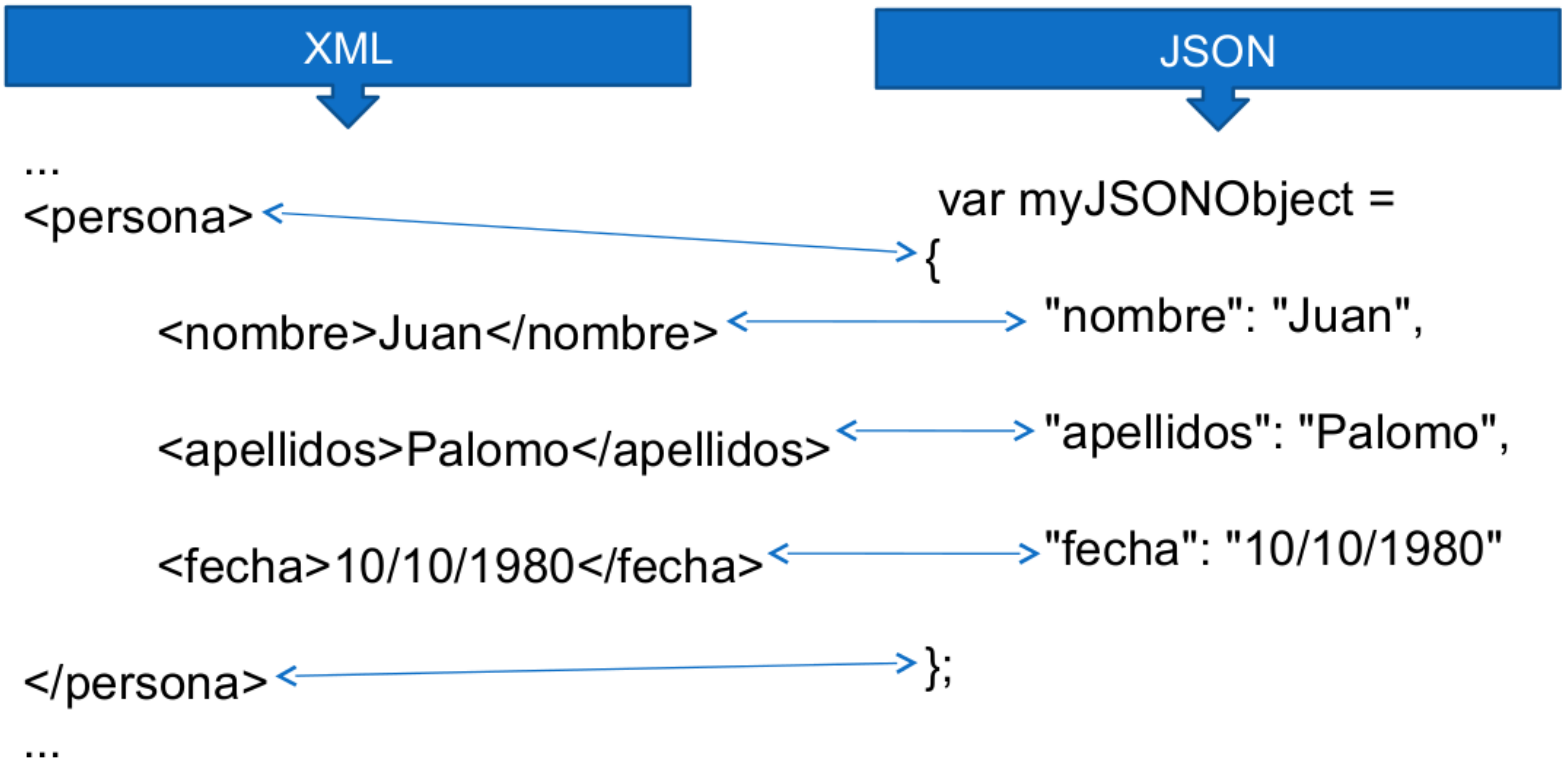
Forma Array

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




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

Comparación con XML

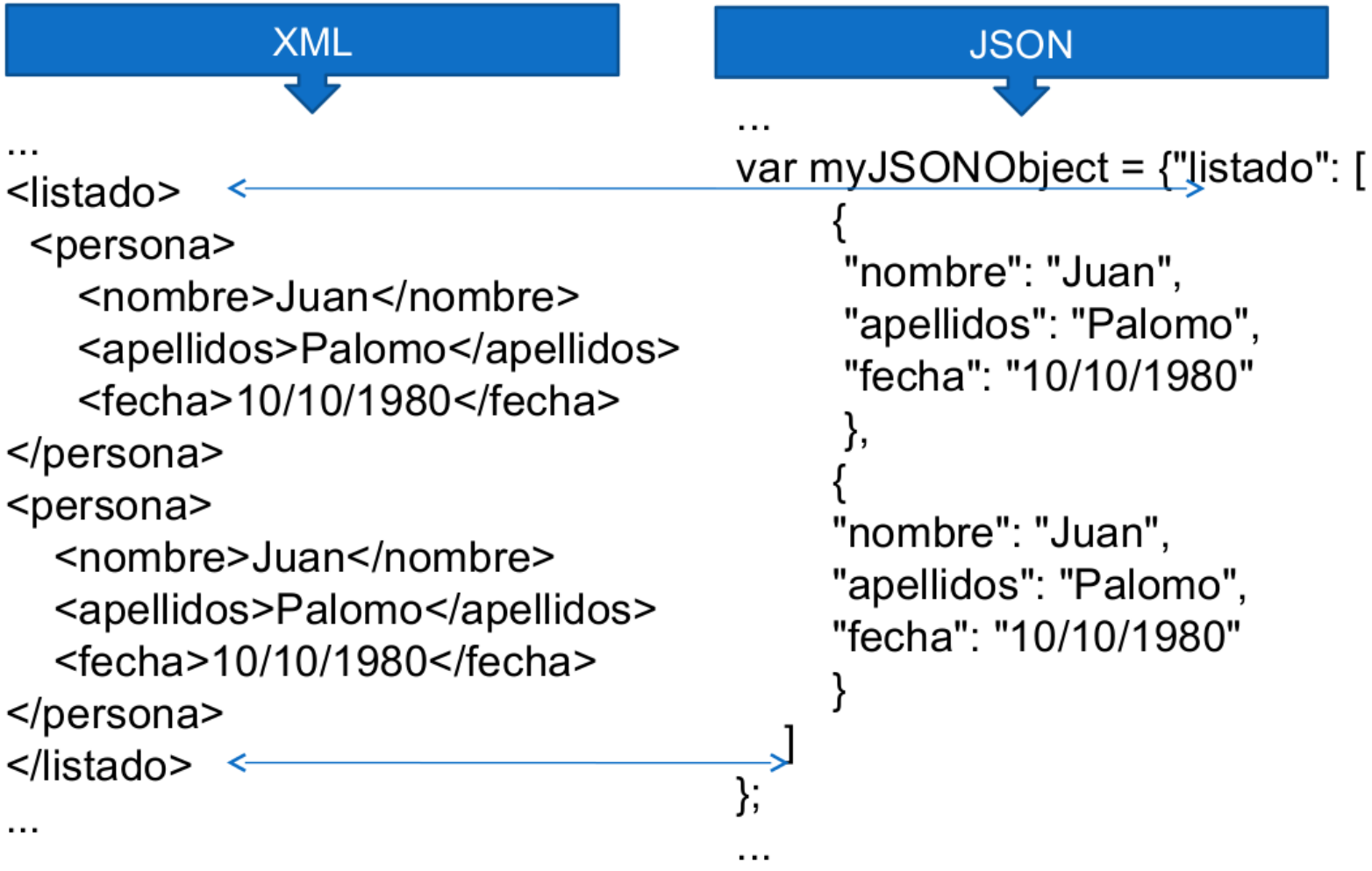


Similitudes con XML

- Ambos son legibles por los humanos
- Tienen una sintaxis muy simple
- Son jerárquicos
- Son independientes del lenguaje de programación
- Se pueden usar en AJAX y en Servicios Web

Diferencias con XML

- Sintaxis dispar
- JSON
  - Es más compacto
  - 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 más extensos
  - Puede ser validado bajo un conjunto de reglas


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>
...
...
var myJSONObject = {"listado": [
{
"nombre": "Juan",
"apellidos": "Palomo",
"fecha": "10/10/1980"
},
{
"nombre": "Juan",
"apellidos": "Palomo",
"fecha": "10/10/1980"
}
]
};
...

Utilización en JavaScript

- Mediante Librerías o nativo
- Transformación de cadenas de texto a objetos
- Transformación de objetos a cadenas de texto
- Personalización de las transformaciones

```
var myJSONObject = {"bindings": [
     {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
     {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
     {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
  ]
};

myJSONObject.bindings[0].method    // "newURI"
```

De texto a objetos: `eval`

```
var myObject = eval('(' + myJSONtext + ')');
```

- Cuidado con lo que hay en el texto que lo 'compila' y ejecuta! (problema de seguridad)
- Paréntesis para acotar ambigüedades

De texto a objetos: utilizar un parser

```
var myObject = JSON.parse(myJSONtext);

var myData = '{"fecha":"mivalorfecha","numero":85}'.parseJSON();
alert( myData.fecha);
alert( myData.numero);
```

- Rápido
- Más seguro: distingue lo que es JSON y lo que no, no ejecuta lo que no es.
- Ya está en el standard de JavaScript de modo nativo

De objetos a texto

```
var myJSONText = JSON.stringify(myObject, replacer);
```

- Si el método `stringify` ve un objeto que contiene el método `toJSON`, lo llama y pasa a cadena el valor retornado

Referencias

- Emmerson Miranda http://emmersonmiranda.blogspot.com/
- http://www.json.org
- Wikipedia: https://en.wikipedia.org/wiki/JSON
- http://www.w3schools.com/json/default.asp