JSON en JavaScript

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

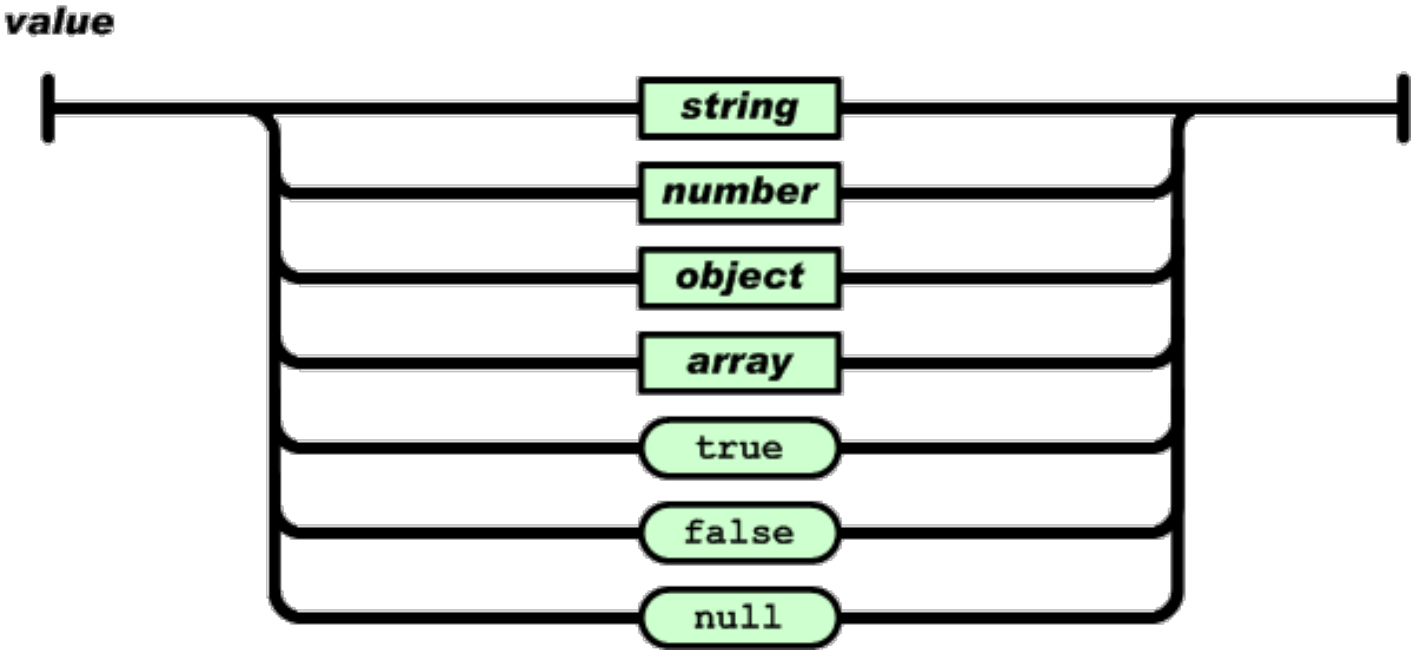
Abril de 2018

JSON

Es un formato ligero para intercambiar datos, independiente del lenguaje de programación y de la plataforma

- Estándar abierto, RFC 4627, año 2006
- Originalmente se consideraba subconjunto del lenguaje JavaScript y se denominaba *JavaScript Object Notation*, aunque ya no es parte de JavaScript
- Diseñado como alternativa a XML, más ligero. Actualmente es más popular que XML
- Carece de algunas características de XML, por ejemplo gramáticas o diferencia entre texto y metadato

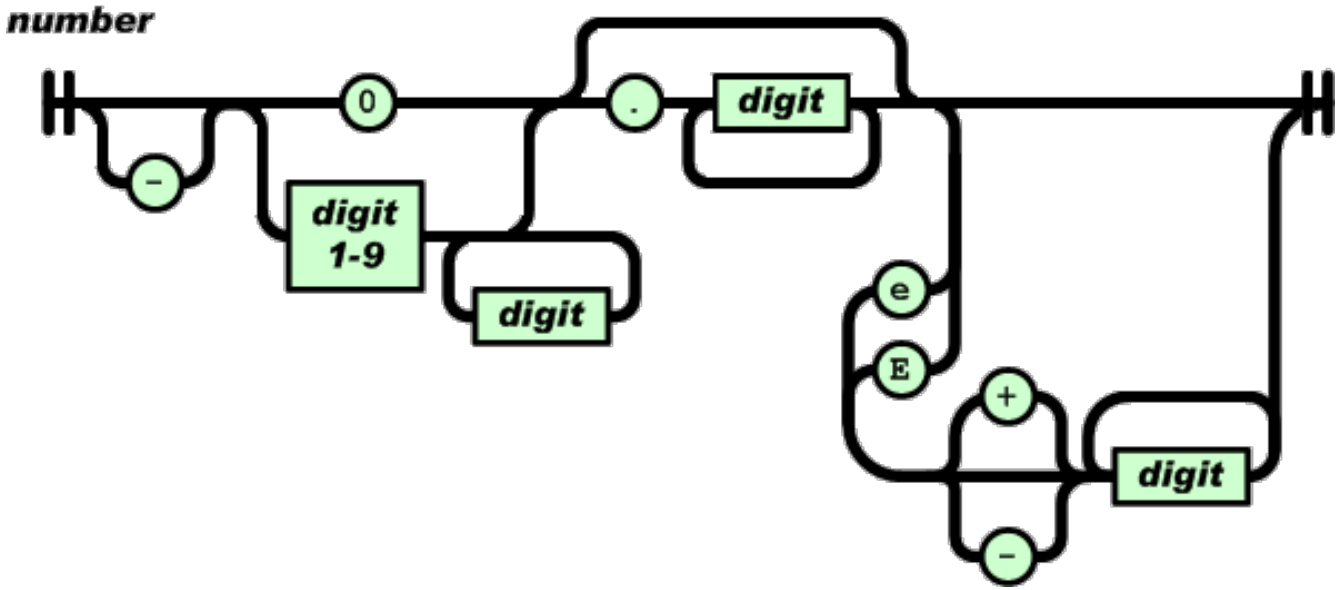
Value



Fuente:json.org

- Un valor JSON puede ser un número, una cadena, un array o un objeto, además de las constantes *true*, *false* y *null*
- Igual que JavaScript. Excepto que *undefined* no es un valor JSON

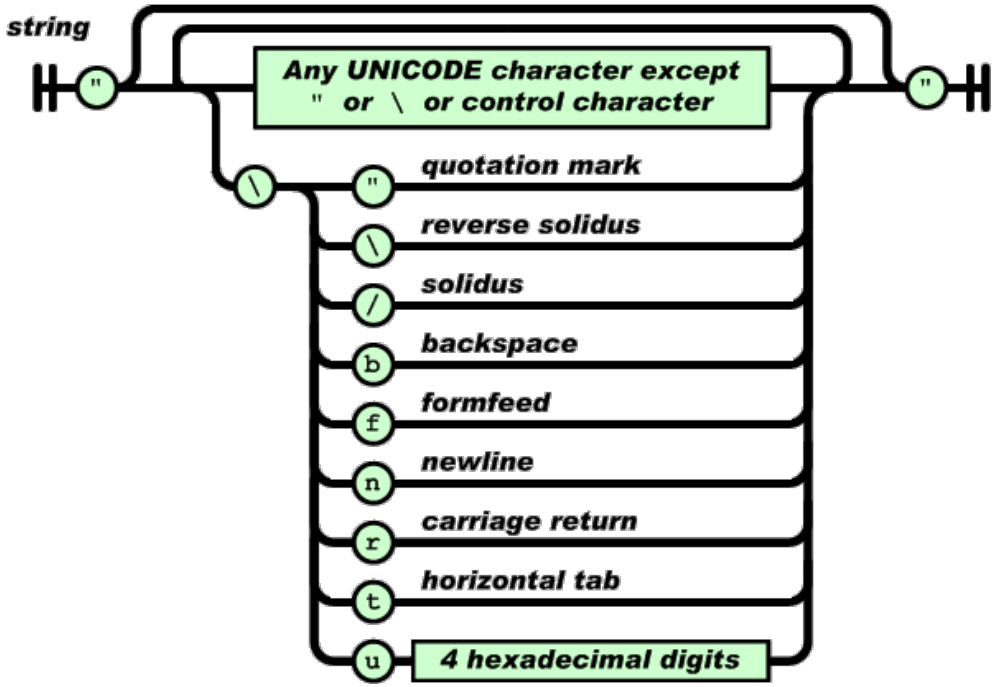
Number



Fuente:json.org

- Los números son como las constantes numéricas de cualquier lenguaje de programación moderno

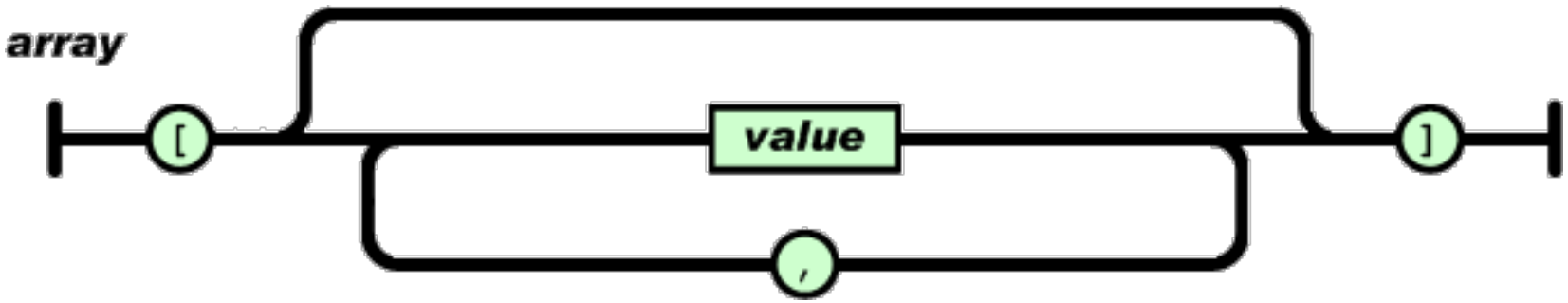
String



Fuente:json.org

- Las cadenas son como las de cualquier lenguaje de programación moderno
- El delimitador es la comilla doble
  - JavaScript admite la comilla doble y la simple, aunque la más habitual es la simple

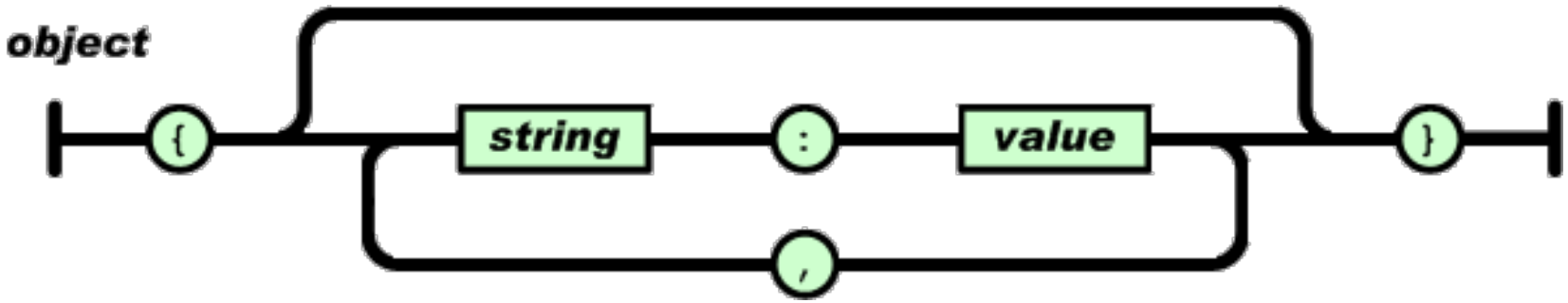
Array



Fuente:json.org

- Un array es una secuencia de valores entre corchetes, separados por comas
- Igual que JavaScript

Objetos



Fuente:json.org

- Un objeto es una secuencia de pares clave:valor, separados por comas
- Las claves son cadenas
- Igual que JavaScript

Ejemplos correctos

- `"hola, mundo"`
- `4243.12`
- `-947e-5`
- `null`
- `[1,2,3,4]`

- `[1, "azul", [1,2,3]]`

- ```
  [
      1,
      "azul",
      [
          1,
          2,
          3
      ]
  ]
  ```

- `["as" ,   "dos", "tres"]`

- `["sota", "caballo", "rey"]`

- ```
  [
      "sota",
      "caballo",
      "rey"
  ]
  ```

- `{ "nombre":"Juan", "apellido":"Pérez"}`
- `{ "v1":true, "v2":null, "v3":false}`
- `{ "nombre": "Juan", "notas":[5.5, 7.2, 6.1]}`
-
```
{
    "nombre": "Juan",
    "notas": [
        5.5,
        7.2,
        6.1
    ]
}
```

Ejemplos incorrectos

- `True`
- `'hola, mundo'`
- `{"hola,mundo"}`
- `{1:"uno", 2:"dos"}`

Same-origin policy

Same-origin policy es una norma que aparece en Netscape 2 (año 1995), que se ha convertido en un estándar. Consiste en que el código JavaScript solo puede acceder a datos que provengan del mismo origen desde el que se ha cargado el script

- Se entiende por *origen* el mismo protocolo, máquina y puerto

Ejemplo

- El usuario accede a una página web en *molamazo.com*,
- Esta página web puede tener código JavaScript que acceda a datos que estén en *molamazo.com*, pero solamente en este sitio
- No puede acceder a datos en *bancofuenla.es*
  - De lo contrario, una vez que el usuario se autentica en *bancofuenla* con una página de *bancofuenla*, un script malicioso en *molamazo* podría acceder a información sensible en *bancofuenla*

JSOP

JSONP (*JSON with padding, JSON con relleno*) es una técnica que permite que una página web obtenga datos desde un sitio web distinto al suyo, sin vulnerar la *same-origin policy*

- Es un protocolo del año 2005, soluciona el problema pero no es especialmente elegante. También tiene algunos problemas de seguridad potenciales
- Una alternativa más avanzada pero menos extendida es CORS, *Cross-origin resource sharing*

JSONP requiere de la colaboración del servidor. Es necesario que un servidor ofrezca datos no solo en JSON, también en JSONP

JSONP se basa en que la *same-origin policy* se aplica solo a datos, no a scripts

- Ejemplo: un script descargado desde *molamazo.com*, no puede descargar datos desde *bancofuenla.es*, pero sí puede descargar otro script
- Se entiende que esto no es especialmente peligroso.
  Bancofuenla podría enviar una contraseña a una página web, como un dato, esperando que la lea el usuario.
  Pero en un script nunca debería haber información sensible

Ejemplo de uso de JSOP

- Enviar 3 no está permitido, es un dato.
- Pero la cadena `"f(3)"` no es un dato. Es un script. Se puede enviar.
  - Si *bancofuenla* accede a enviar `"f(3)"` como dato JSON, sabe que esta información podría ser usada por un script de cualquier otro sitio, p.e. *molamazo*. Por tanto, nunca enviará de este modo información sensible
  - Un problema distinto, que JSONP no contempla, es que sea *bancofuenla* quien aproveche esto para enviar código dañino

En JSONP, el cliente solicita un dato a un sitio web (en nuestro ejemplo, *bancofuenla*), y también le indica el nombre de la función en la que se *envuelve* el dato.

- Esto se hace con un parámetro que suele denominarse `callback`
  Una petición podría ser

  `bancofuenla.es/divisas.html?par=USDEUR&fecha=hoy&callback=procesaDivisas`

- A la que el servidor podría responder `procesaDivisas(0.865)`
- `procesaDivisas()` será una función en el script cliente, que tratará el dato (0.865)

Conversión de objeto en cadena JSON

En JavaScript, para convertir un objeto en una cadena JSON usamos la *built-in function* JSON.stringify

```
'use strict'
let lista=[ "sota", "caballo", "rey" ];
console.log(typeof(lista),lista);
// object ["sota","caballo","rey"]

let cadena=JSON.stringify(lista);
console.log(typeof(cadena),cadena);
// string ["sota","caballo","rey"]
```

Conversión de cadena JSON en objeto

Para convertir una cadena de texto con un JSON en un objeto JavaScript, disponemos de la *built-in function* JSON.parse

```
'use strict'
let cadena='{ "nombre":"redes", "curso":1, "horario":["L1500", "X1700"] }'
console.log(typeof(cadena),cadena);
 // string { "nombre":"redes", "curso":1, "horario":["L1500", "X1700"] }

let objeto=JSON.parse(cadena);
console.log(typeof(objeto),objeto);
/*
object { nombre: 'redes',
  curso: 1,
  horario: [ 'L1500', 'X1700' ] }
*/
```

Si la cadena no cumple el formato JSON, se genera una excepción *SyntaxError*