

Comunicación de datos JSON y AJAX

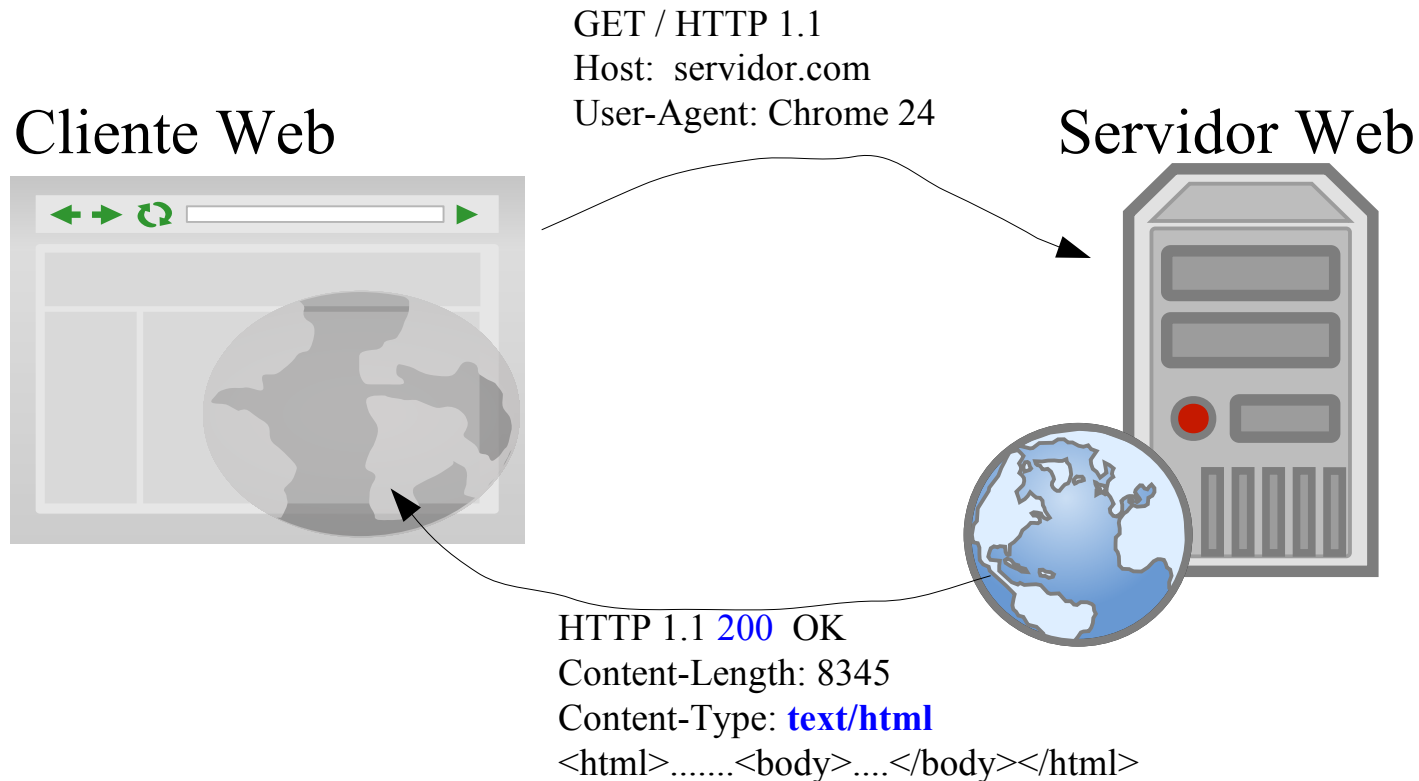
DAWE

Departamento LSI

Juanan Pereira <juanan.pereira@ehu.es>

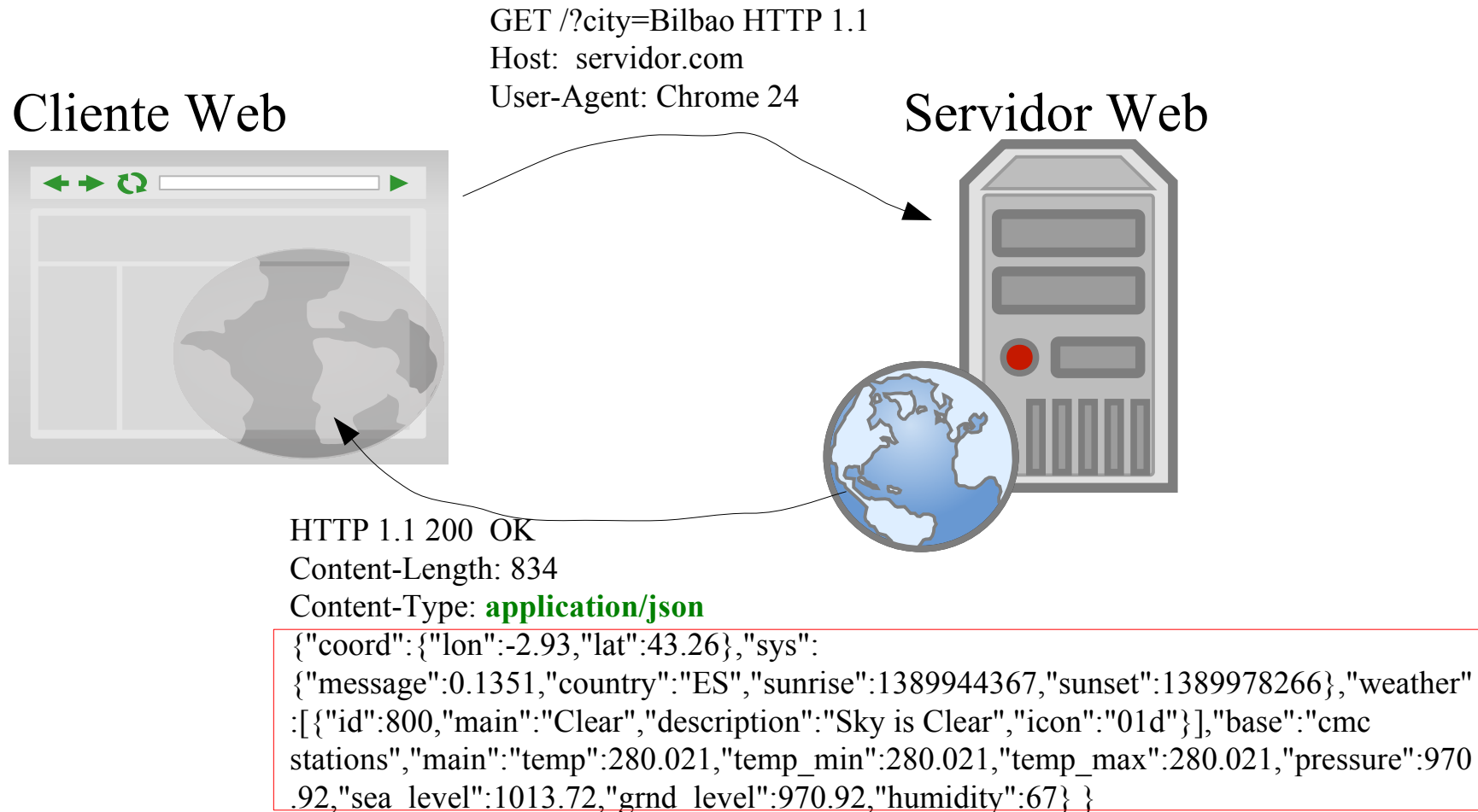
Comunicación de datos

Sabemos que los navegadores pueden solicitar páginas web a servidores externos usando el protocolo HTTP.



Comunicación de datos

Pero, en muchas ocasiones, “sólo” necesitaremos descargar datos.

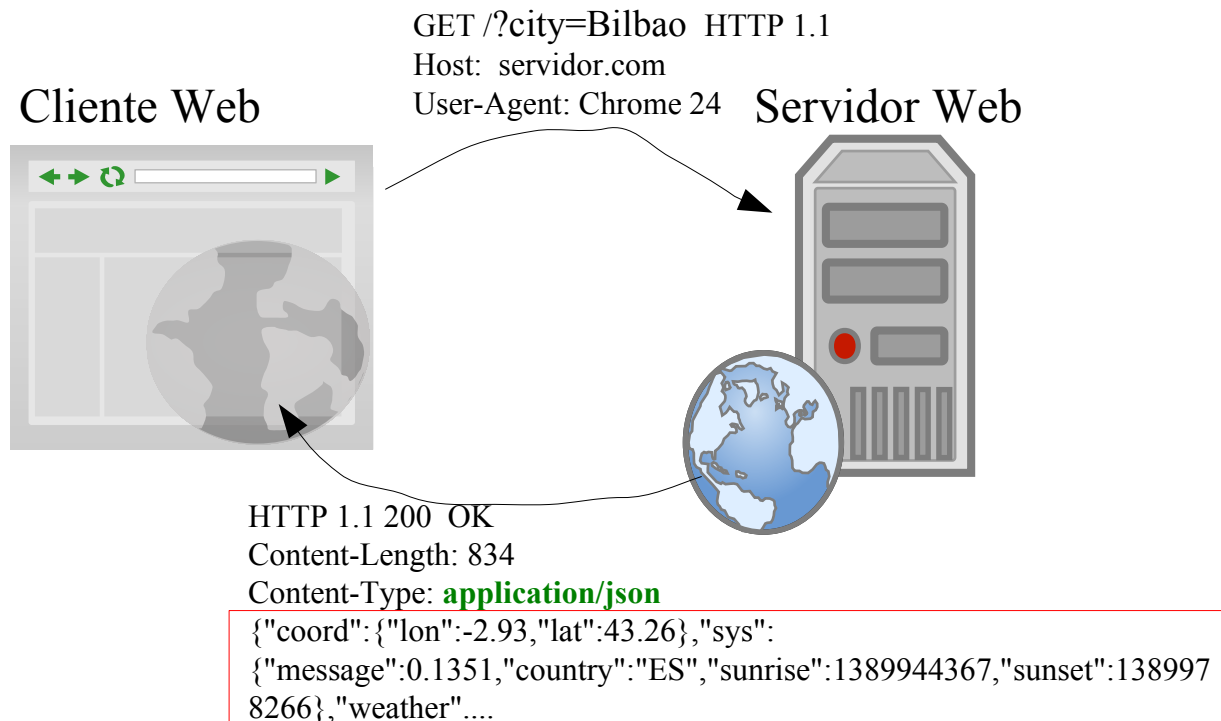


Comunicación de datos (JSON)

```
{
  "coord": {
    "lon": -2.93,
    "lat": 43.26
  },
  "sys": {
    "message": 0.1351,
    "country": "ES",
    "sunrise": 1389944367,
    "sunset": 1389978266
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "Sky is Clear",
      "icon": "01d"
    }
  ],
  "base": "cmc stations",
  "main": {
    "temp": 280.021,
    "temp_min": 280.021,
    "temp_max": 280.021,
    "pressure": 970.92,
    "sea_level": 1013.72,
    "grnd_level": 970.92,
    "humidity": 67
  },
  "wind": {
    "speed": 4.36,
    "deg": 206.004
  },
  "clouds": {
    "all": 0
  },
  "dt": 1389948981,
  "id": 3128026,
  "name": "Bilbao",
  "cod": 200
}
```



Comunicación de datos



Las peticiones de datos se hacen de forma asíncrona usando **XMLHttpRequest**. Esta técnica también se conoce como **Ajax** o **XHR**.

Comunicación de datos

HTTP 1.1 200 OK

Content-Length: 834

Content-Type: **application/json**

```
{"coord":{"lon":-2.93,"lat":43.26},"sys":  
{"message":0.1351,"country":"ES","sunrise":1389944367,"sunset":138997  
8266},"weather"....
```

Las peticiones de datos se hacen de forma asíncrona usando
XMLHttpRequest.

¡Ojo! No debe confundirnos este prefijo. Aunque inicialmente se trabajaba únicamente con datos **XML**, las peticiones XHR soportan también (y en general) **JSON** como formato de datos a devolver.

Comunicación de datos

Cómo ejecutar una consulta Ajax usando JavaScript

¡OJO! Desde hace unos meses es necesario añadir el APIKey de OpenWeatherMap para realizar consultas AJAX contra este servicio. Ver: <http://jsfiddle.net/q3L1wsqa/>

```
// especificar el servidor y la consulta
var url = "http://api.openweathermap.org/data/2.5/weather?q=Bilbao,es";

// Crear el objeto XHR que gestionará la consulta
var consulta = new XMLHttpRequest();

// indicar que la consulta hacia esa URL se realizará usando el método GET
consulta.open("GET", url);

// especificar la función que gestionará el resultado de la consulta asíncrona
consulta.onload = function() {
    if (consulta.status == 200) {
        console.log("Consulta ejecutada con éxito");
        console.log( consulta.responseText );
    }
};

// para finalizar, es necesario lanzar la petición
consulta.send();
```



¡OjO! En IE <= 8 este código no funciona.

Comunicación de datos

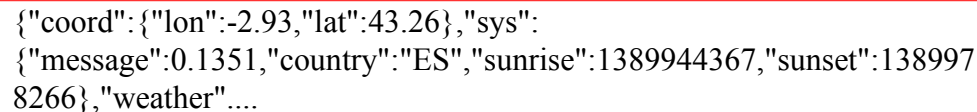
Cómo tratar una respuesta JSON (I)

HTTP 1.1 200 OK

Content-Length: 834

Content-Type: **application/json**

```
{"coord":{"lon":-2.93,"lat":43.26},"sys":  
{"message":0.1351,"country":"ES","sunrise":1389944367,"sunset":138997  
8266},"weather"....
```



Las respuestas JSON que nos devuelve el servidor son “simples” cadenas de texto (Strings). Necesitamos convertir esas cadenas en **objetos** JSON para poder manipularlos desde JavaScript.

Comunicación de datos

Cómo tratar una respuesta JSON (II)

HTTP 1.1 200 OK

Content-Length: 834

Content-Type: **application/json**

```
{"coord":{"lon":-2.93,"lat":43.26},"sys":  
{"message":0.1351,"country":"ES","sunrise":1389944367,"sunset":138997  
8266},"weather"....
```

Convertir consulta.responseText
a objeto JSON



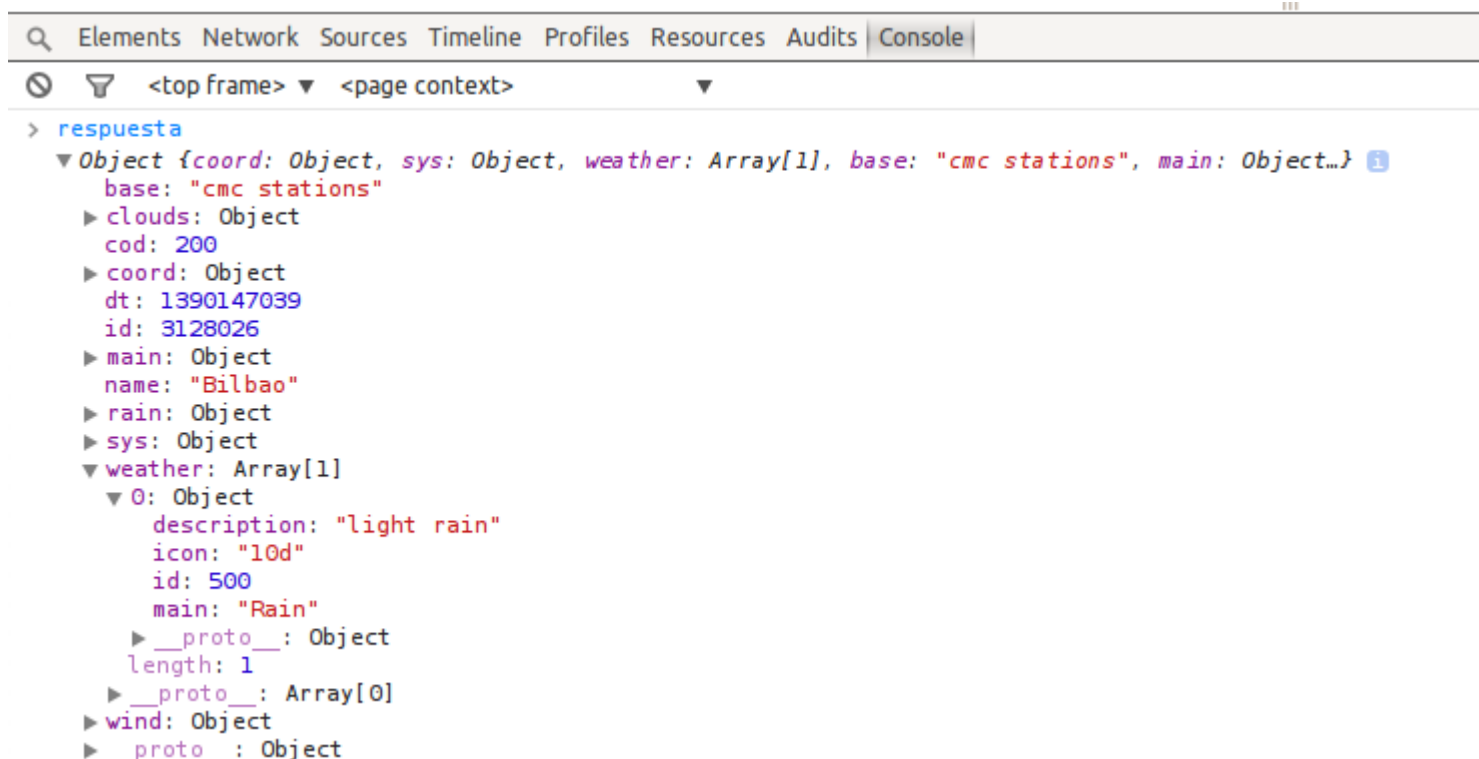
```
var respuesta = JSON.parse(consulta.responseText)
```

Comunicación de datos

Cómo tratar una respuesta JSON (III)

```
var respuesta = JSON.parse(consulta.responseText)
```

// Ahora **respuesta** es un objeto JavaScript que ya sabemos tratar



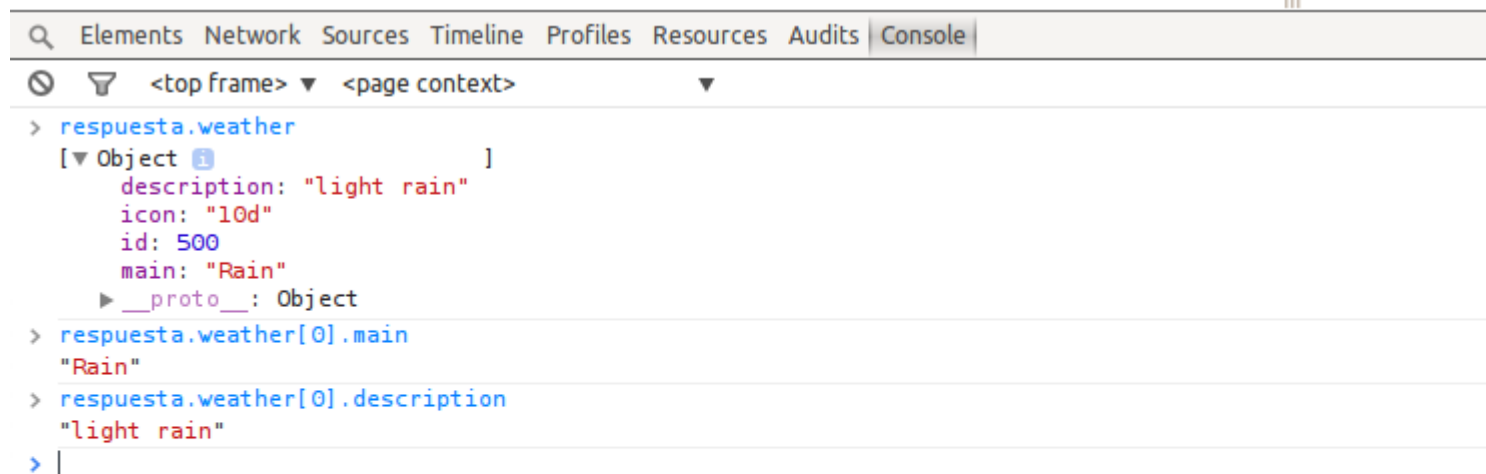
```
> respuesta
Object {coord: Object, sys: Object, weather: Array[1], base: "cmc stations", main: Object...}
  base: "cmc stations"
  clouds: Object
  cod: 200
  coord: Object
    dt: 1390147039
    id: 3128026
  main: Object
    name: "Bilbao"
  rain: Object
  sys: Object
  weather: Array[1]
    0: Object
      description: "light rain"
      icon: "10d"
      id: 500
      main: "Rain"
      __proto__: Object
    length: 1
    __proto__: Array[0]
  wind: Object
  __proto__: Object
```

Comunicación de datos

Cómo tratar una respuesta JSON (IV)

```
var respuesta = JSON.parse(consulta.responseText)
```

// Ahora `respuesta` es un objeto JavaScript que ya sabemos tratar



The screenshot shows a web browser's developer console with the 'Console' tab selected. The breadcrumb path is '<top frame> > <page context>'. The console contains the following commands and their outputs:

- `> respuesta.weather` outputs an array containing one object: `[Object]`. The object's properties are: `description: "light rain"`, `icon: "10d"`, `id: 500`, `main: "Rain"`, and `__proto__: Object`.
- `> respuesta.weather[0].main` outputs the string `"Rain"`.
- `> respuesta.weather[0].description` outputs the string `"light rain"`.
- `> |` (The cursor is on a new line, indicating the next command has not been executed).

Comunicación de datos

Cómo tratar una respuesta JSON (V)

// también podemos convertir un objeto JavaScript en un String JSON

```
var libro1 = new Libro("Dublinés", "Alfonso Zapico", 18);
```

```
var libroJSON = JSON.stringify(libro1);
```

// este libroJSON podremos enviarlo como datos a un servidor externo

// o guardarlo localmente para usos futuros (como veremos más adelante)

Comunicación de datos

Cómo tratar una respuesta JSON (VI)

The screenshot shows a web browser window with the address bar displaying `dawe/productos10.html`. The page content lists three products:

- Producto: Dublinés
Autor: Alfonso Zapico
Precio: 18
- Producto: El arte de volar
Autor: Antonio Altarriba y Kim
Precio: 20.9
- Producto: Próxima estación: Esperanza
Autor: Manu Chao
Precio: 15

An alert dialog box is displayed in the foreground with the title "The page at dawe says:" and the message `{"titulo": "Dublinés", "autor": "Alfonso Zapico", "precio": 18}`. The dialog has a red exclamation mark icon and an "OK" button.

The browser's developer console is open, showing the following JavaScript code:

```
> libro1
  ▶ Libro {titulo: "Dublinés", autor: "Alfonso Zapico", precio: 18, marcador: function, pasarPagina: function...}
> var libroJSON = JSON.stringify(libro1)
  undefined
> alert(libroJSON)
>
```

Comunicación de datos

Cómo ejecutar una consulta Ajax usando JavaScript+jQuery



```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"> </script>
<script>
var direccion = "http://api.openweathermap.org/data/2.5/weather?q=Bilbao,es";

$.ajax({url:direccion,
        success:function(result){
            console.log("Consulta ejecutada con éxito");
        }
    });
</script>
```

¡OJO! Desde hace unos meses es necesario añadir el APIKey de OpenWeatherMap para realizar consultas AJAX contra este servicio. Ver: <http://jsfiddle.net/q3L1wsqa/>

Comunicación de datos

Cosas que nos hemos dejado en el tintero...

- * Seguridad:

- + Same-Origin-Policy

- http://en.wikipedia.org/wiki/Same_origin_policy

- + Estándar CORS (Cross-Origin Resource Sharing)

- http://en.wikipedia.org/wiki/Cross-origin_resource_sharing

- + Same-Origin-Policy y CORS con jQuery

- <http://www.adobe.com/devnet/html5/articles/understanding-cross-origin-resource-sharing-cors.html>

- + JSONP ("JSON with padding")