



**JS**

# UT08: CARACTERÍSTICAS AVANZADAS DE JAVASCRIPT (II)

# ÍNDICE

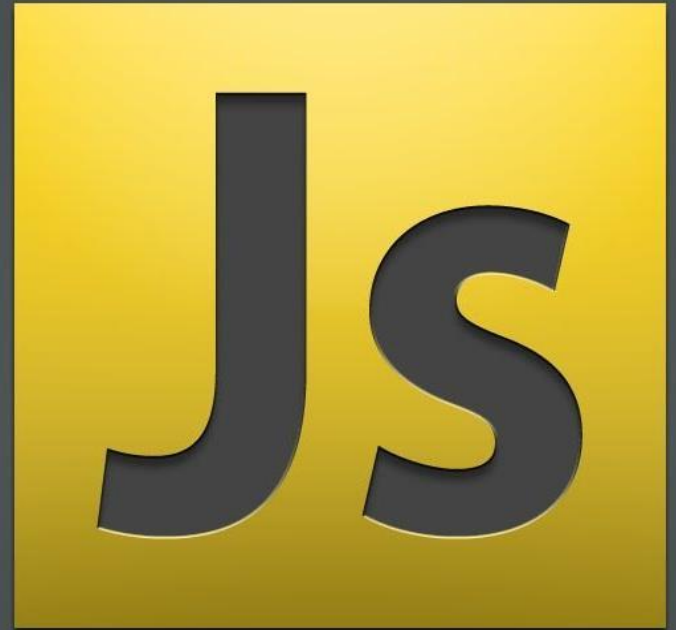
---

- 1.- JSON Web Token (JWT)
- 2.- localStorage y sessionStorage
- 3.- Fecha y hora
- 4.-
- 5.- IndexedDB



# 1

JSON WEB  
TOKEN (JWT)



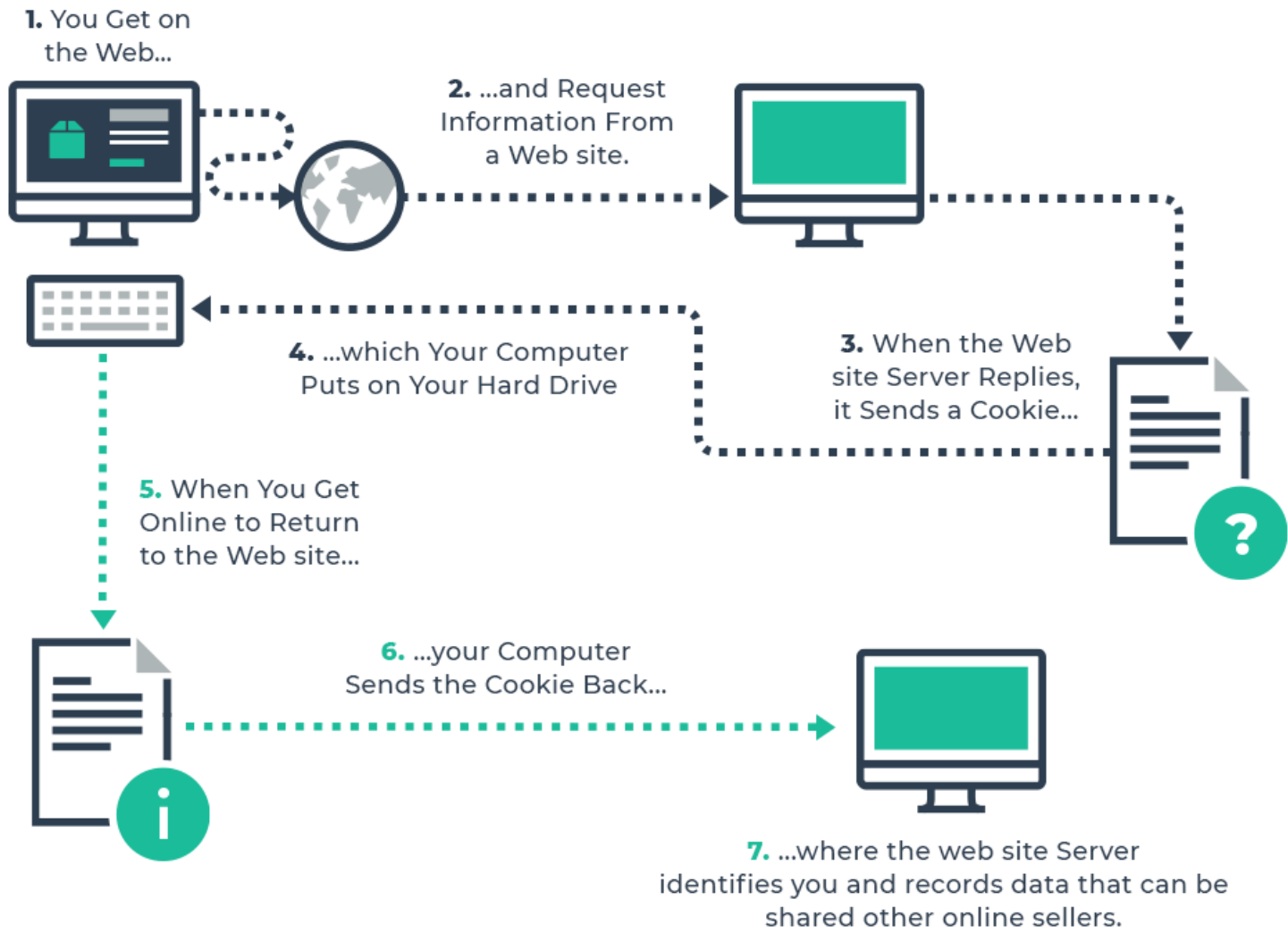
Tradicionalmente, el mantenimiento de la sesión en aplicaciones Web se ha realizado con **cookies**.

Una **cookie** es un par nombre-valor que está asociada a un dominio.

El mecanismo de las cookies es muy sencillo:

- El servidor envía la cookie al navegador en la respuesta, donde es almacenada.
- En cualquier petición que se realice al servidor desde el cliente se envía **automáticamente** la cookie





Con las cookies podemos comprobar si el usuario ha iniciado sesión, pero son exclusivas del servidor que las ha generado, por lo que la sesión no se puede mantener cuando hay consultas a varios servidores.

Esto puede ser un problema cuando trabajamos con APIs, especialmente sobre **microservicios**, que pueden estar repartidos en múltiples servidores.

La alternativa es el uso de JSON Web Token (JWT).

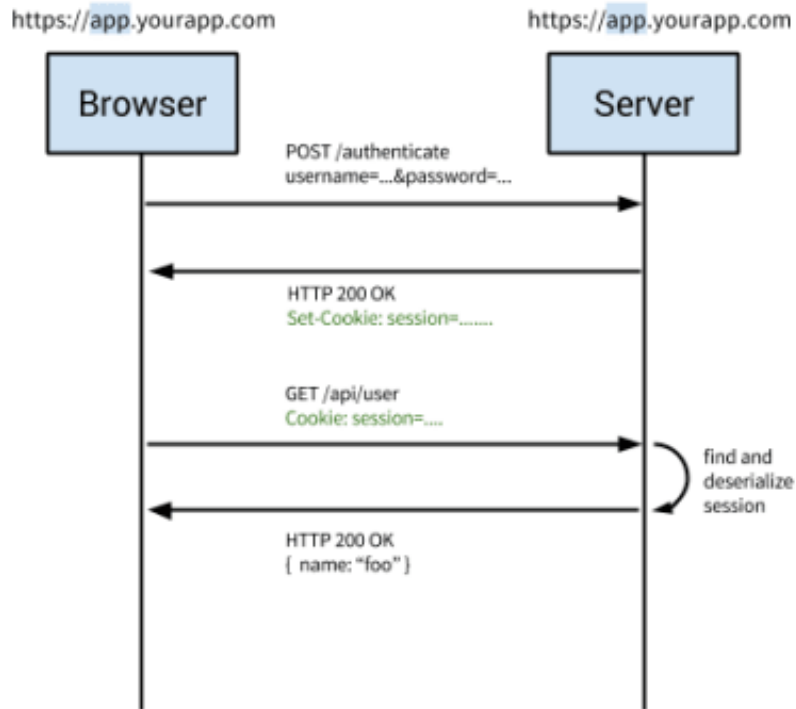
Un JWT es simplemente un **token** cuya posesión acredita que el usuario es quien dice ser:

- El usuario se autentica en el servidor, el cual le devuelve el token.
- Cada vez que el usuario realice una petición debe adjuntar manualmente el token para acreditar su identidad.

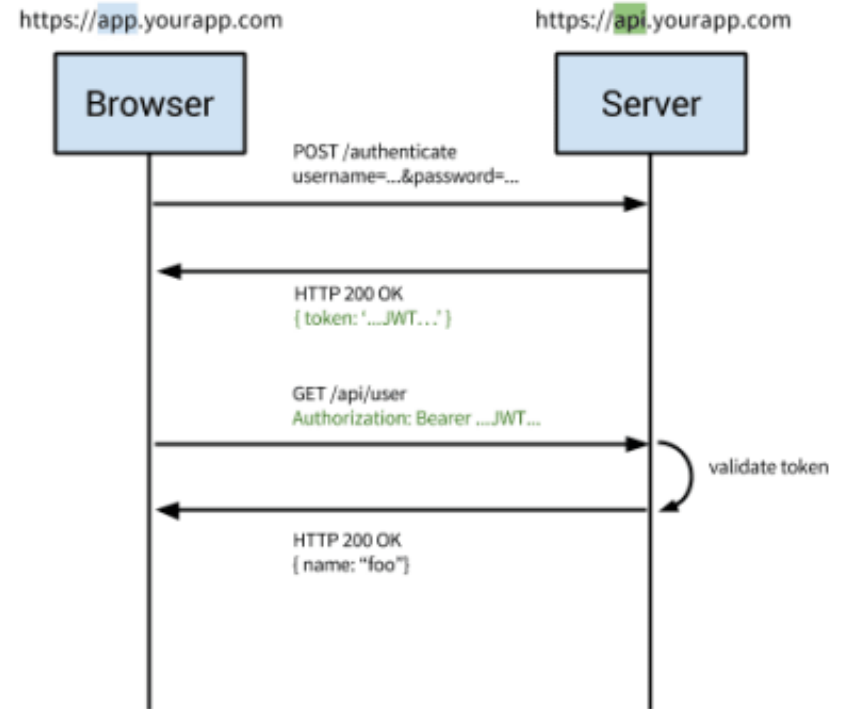
Es importante destacar que ni las cookies ni JWT son mecanismos de autenticación, sino que son utilizados después de la autenticación para acreditar al usuario.



## Traditional Cookie-Based Auth



## Modern Token-Based Auth



La autenticación basada en JWT **carece de estado** (stateless), ya que el servidor no guarda información de que usuarios hay conectados ni almacena los tokens que se han emitido.

Normalmente los tokens se envían como un *Authorization header* con el valor *Bearer {JWT}*, pero también se pueden enviar en el cuerpo de una petición POST o incluso como *query parameter*.

Los pasos son:

- El usuario envía sus credenciales
- El servidor verifica su validez y devuelve un token firmado
- El token se guarda en el cliente en el *local storage*, el *sesión storage*, en memoria o incluso como una *cookie*.
- Cualquier petición a partir de ahora incluirá ese token
- El servidor decodifica el JWT y verifica si el token es válido
- Cuando el usuario se desconecta el token es eliminado en el lado del cliente.

Todo lo relativo a JWT está definido en el estándar RFC 7519.

Un token JWT consta de tres partes, codificadas en Base64 y separadas por puntos.

**HEADER.PAYLOAD.SIGNATURE**

## Header

Consta generalmente de dos valores: el tipo de token y el algoritmo de firma o cifrado que se ha utilizado.

```
{ "alg": "HS256", "typ": "JWT" }
```

Algoritmos:

- **HS256**: HMAC-SHA256
- **RS256**: RSA con SHA-256
- **ES256**: ECDSA con SHA-256

## Payload

Contiene la información real que se transmitirá a la aplicación. Se indica como pares clave-valor denominadas **claims**, que pueden ser de tres tipos:

- **Claims registrados:** que figuran en el [IANA JSON Web Token Claim Register](#).
- **Claims públicos:** pueden definirse a voluntad, pero hay que registrarlos en el IANA para evitar conflictos.
- **Claims privados:** cuando los utilizamos en comunicaciones entre nuestras propias aplicaciones.

Todos los *claims* son opcionales, debiendo limitarse a la información estrictamente necesaria.

Ejemplo:

```
{  
  "name": "victor",  
  "admin": "true",  
  "exp": "1674453570"  
}
```

## Signature


Se crea utilizando una codificación **Base64** del *header* y del *payload*, así como el método de firma o cifrado.

Es necesaria una **clave secreta** únicamente conocida por la aplicación original.


La firma garantiza que el mensaje no haya sido modificado.



Hay múltiples librerías en JavaScript para crear el JWT. Por ejemplo, **jsonwebtoken**.

jsonwebtoken 

9.0.0 • Public • Published a month ago

 Readme

 Code Beta

 4 Dependencies

 22.232 Dependents

 79 Versions

## jsonwebtoken

Build	Dependency
	<a href="#">Dependency Status</a>

An implementation of **JSON Web Tokens**.

This was developed against `draft-ietf-oauth-json-web-token-08`. It makes use of **node-jws**

## Install

```
$ npm install jsonwebtoken
```


### Install

```
> npm i jsonwebtoken
```

### Repository

 [github.com/auth0/node-jsonwebtoken](https://github.com/auth0/node-jsonwebtoken)

### Homepage

 [github.com/auth0/node-jsonwebtoken#...](https://github.com/auth0/node-jsonwebtoken#readme)

### Weekly Downloads

11.653.880



Version

9.0.0

License

MIT

En el servidor:

```
const jwt = require('jsonwebtoken');
const secret = 'S3cr3T';

app.post("/login", (req, res) => {
  const {username, pass} = req.body;
  // Aquí comprobaría si las credenciales son válidas
  const token = generateAccessToken({
    username: req.body.username });
  res.json(token);
})

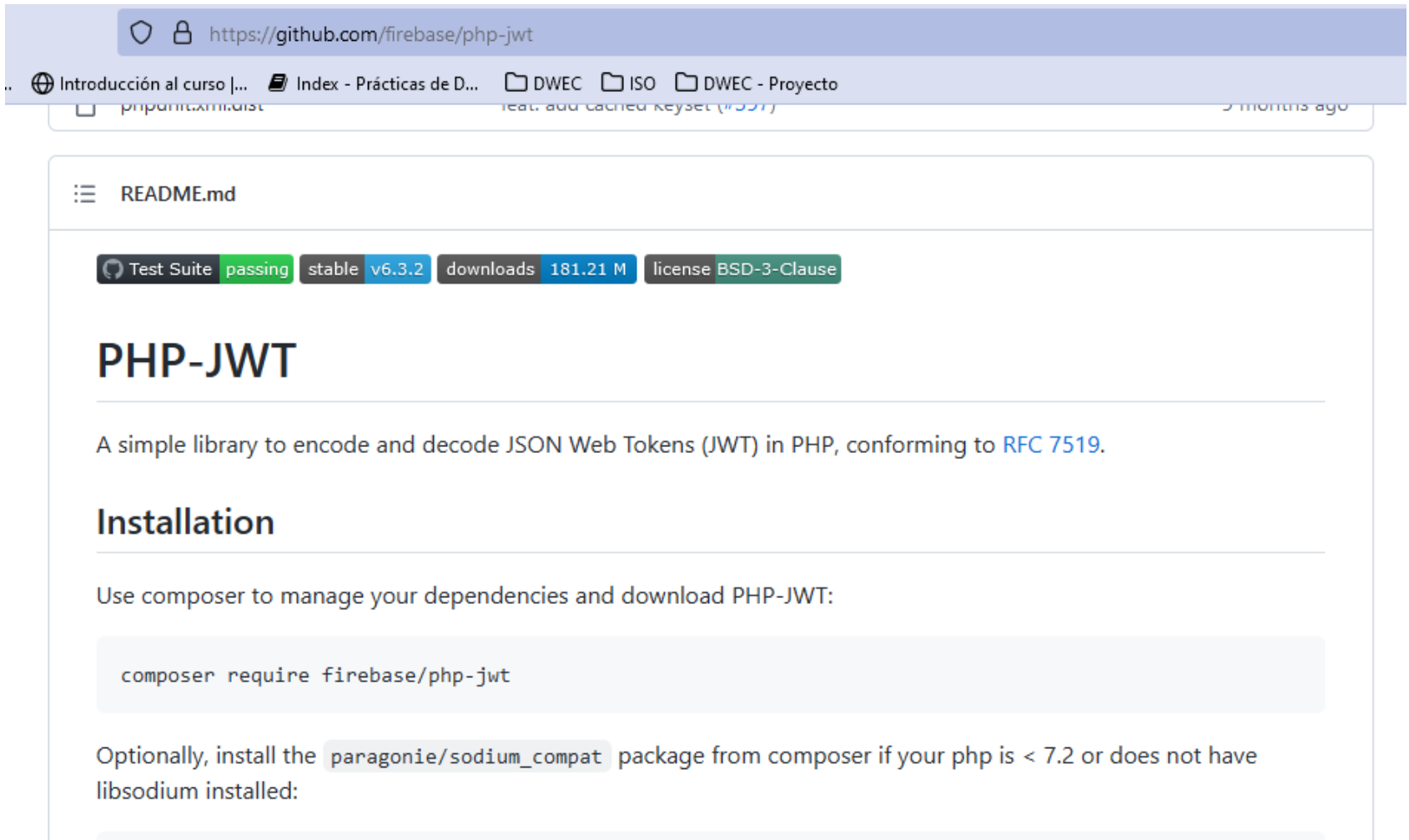
app.get("/test", (req, res) => {
  const headers = req.headers;
  let decoded = jwt.verify(headers.authorization.split('
')[1], secret);
  console.log(decoded);
})
```

En el cliente: solicitud de login para obtener jwt

```
fetch('http://localhost:5000/login', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    username: 'victor',
    pass: '1234'
  })
})
.then( response => response.json())
.then( jwt => {
  // Se almacenaría de alguna manera
  console.log(jwt);
});
```

En el cliente: llamada adjuntando el token

```
fetch('http://localhost:5000/test', {  
  headers: {  
    Authorization: `Bearer ${jwt}`  
  }  
})  
  .then( response => console.log(response));
```



The screenshot shows the GitHub repository page for `firebase/php-jwt`. The browser's address bar displays the URL `https://github.com/firebase/php-jwt`. The repository's README is visible, featuring a header with repository statistics: "Test Suite passing", "stable v6.3.2", "downloads 181.21 M", and "license BSD-3-Clause". The main heading is "PHP-JWT", followed by a description: "A simple library to encode and decode JSON Web Tokens (JWT) in PHP, conforming to RFC 7519." The "Installation" section instructs users to use Composer to manage dependencies and download PHP-JWT, providing the command `composer require firebase/php-jwt`. It also mentions an optional step to install `paragonie/sodium_compat` for older PHP versions or those without libsodium.

https://github.com/firebase/php-jwt

Introducción al curso |... Index - Prácticas de D... DWECC ISO DWECC - Proyecto

phpunit:xml:dist feat: add cached keyset (#557) 5 months ago

☰ README.md

Test Suite passing stable v6.3.2 downloads 181.21 M license BSD-3-Clause

## PHP-JWT

A simple library to encode and decode JSON Web Tokens (JWT) in PHP, conforming to [RFC 7519](#).

### Installation

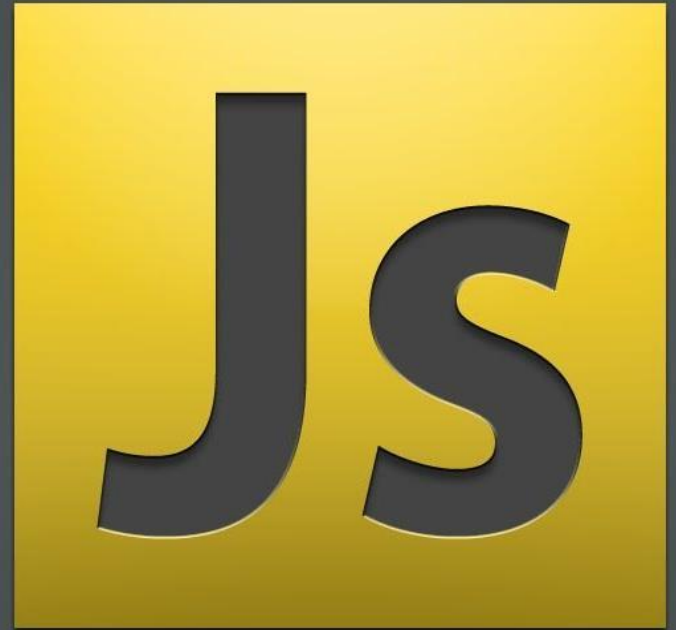
Use composer to manage your dependencies and download PHP-JWT:

```
composer require firebase/php-jwt
```

Optionally, install the `paragonie/sodium_compat` package from composer if your php is < 7.2 or does not have libsodium installed:

2

LOCAL STORAGE



Los objetos **localStorage** y **sessionStorage** permiten guardar pares clave/valor.

Lo más relevante es que los datos sobreviven a una recarga de la página (**sessionStorage**) y hasta al reinicio del ordenador (**localStorage**).

Diferencias de ambos respecto a las cookies son:

- No se envían automáticamente al servidor
- El servidor no puede manipularlos vía cabeceras HTTP, todo se hace con JavaScript
- El almacenaje está vinculado al origen (tripleto dominio/protocolo/puerto)



Los métodos disponibles son:

`setItem(clave, valor)` – almacenar un par clave/valor.

`getItem(clave)` – obtener el valor por medio de la clave.

`removeItem(clave)` – eliminar la clave y su valor.

`clear()` – borrar todo.

`key(índice)` – obtener la clave de una posición dada.

`length` – el número de ítems almacenados.

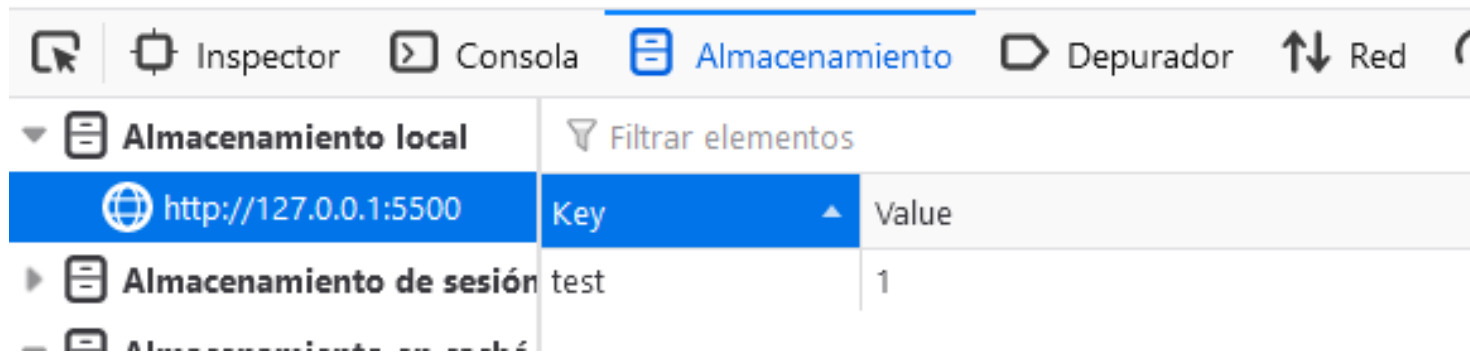
## Local Storage

Las principales funcionalidades del **localStorage** son:

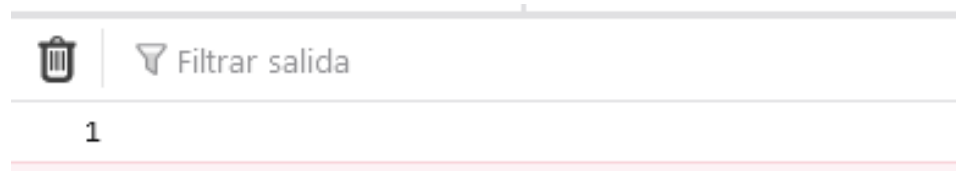
- Es compartido entre todas las pestañas y ventanas del mismo origen.
- Los datos no expiran. Persisten a los reinicios del navegador y del sistema operativo

**setItem** y **getItem** permiten escribir y recuperar valores.

```
localStorage.setItem('test', 1);
```

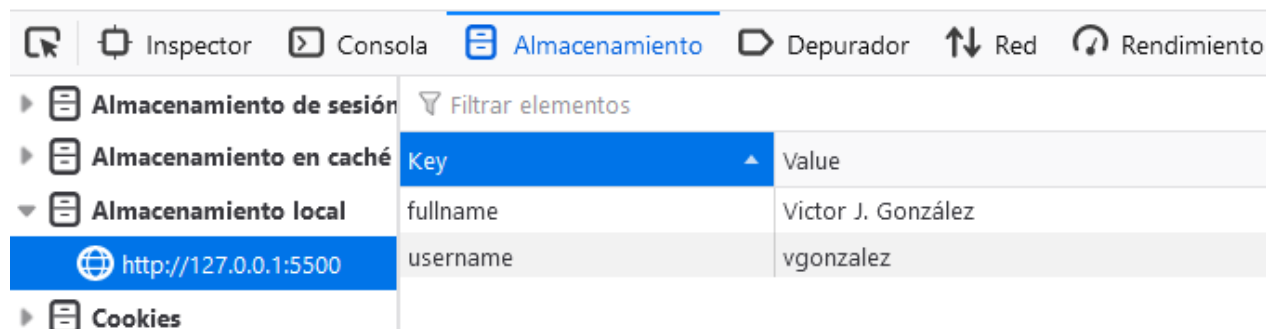


```
console.log(localStorage.getItem('test'));
```

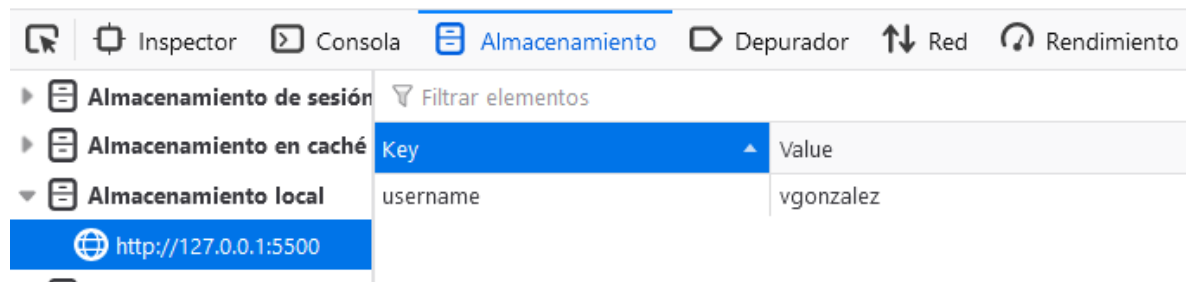


**removeItem()** elimina el valor que se indique y **clear()** borra todos los valores.

```
localStorage.setItem('fullname', 'Victor');  
localStorage.setItem('username', 'vgonzalez');
```



```
console.log(localStorage.removeItem('fullname'));
```



Los objetos de almacenaje **no son iterables**, por lo que la única forma de iterar sobre todos sus elementos es utilizar iteración usando el índice mediante la propiedad **length** y la función **key()**.

```
localStorage.setItem('fullname', 'Victor J. González');
localStorage.setItem('username', 'vgonzalez');

for(let i=0; i<localStorage.length; i++) {
  let key = localStorage.key(i);
  console.log(`${key}: ${localStorage.getItem(key)}`);
}
```



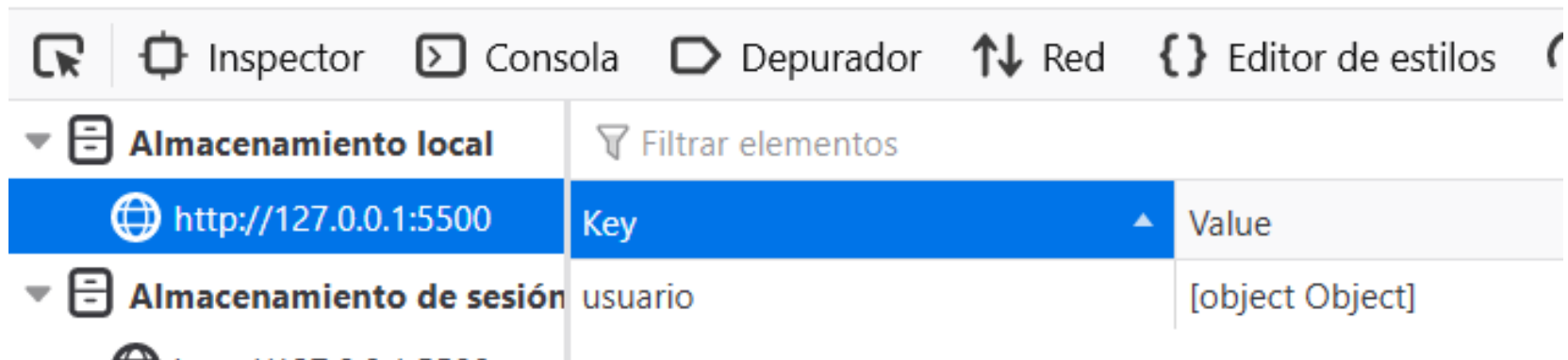
Filtrar salida

username: vgonzalez

fullname: Victor J. González

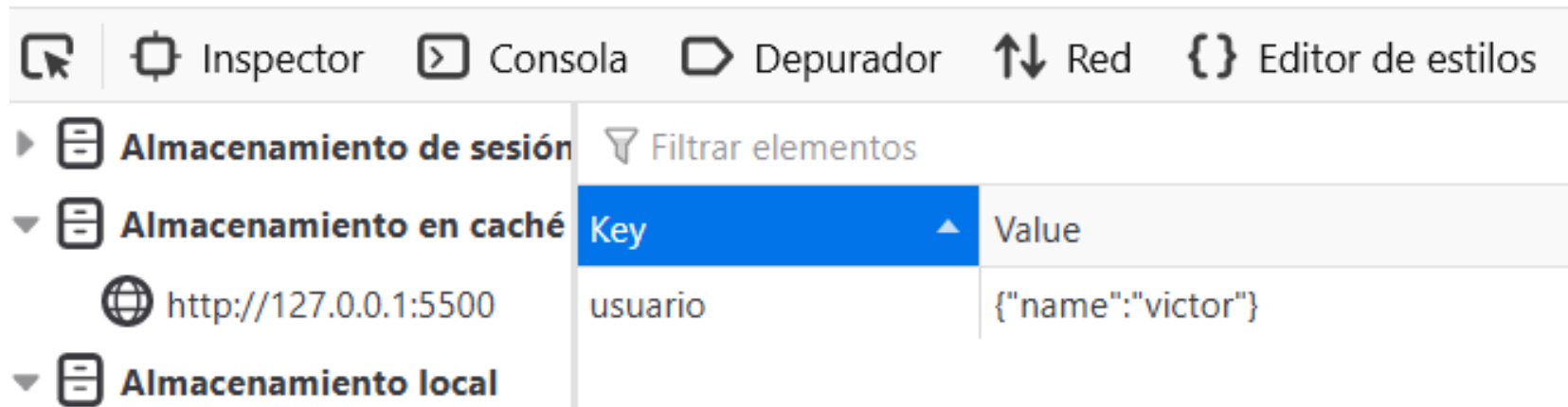
Algo importante es que en el localStorage **solo se pueden guardar *strings***, por lo que cualquier otro tipo de datos se convertirá automáticamente a una cadena.

```
localStorage.usuario = { name: "victor"};
```



Si queremos guardar JSON deberemos utilizar la función **JSON.stringify()** para convertirlo adecuadamente a una cadena.

```
localStorage.usuario = JSON.stringify(  
    { name: "victor"});
```



## Session Storage

### Características:

- Mismos métodos y características que localStorage
- Solo existe dentro de la pestaña del navegador
  - Otra pestaña con la misma página tendrá un contenido distinto
  - Sí se comparte entre iframes del mismo origen de la misma pestaña
- Los datos sobreviven a un refresco de página, pero no a cerrar/abrir la pestaña



## Evento storage

Hay un evento asociado al *localStorage* y *sessionStorage* denominado **storage** que se dispara cada vez que hay cambios en el *localStorage*.

Sus propiedades son:

`key` – la clave que ha cambiado, (`null` si se llama `.clear()`).

`oldValue` – el anterior valor (`null` si se añade una clave).

`newValue` – el nuevo valor (`null` si se borra una clave).

`url` – la url del documento donde ha pasado la actualización.

`storageArea` – bien el objeto `localStorage` o `sessionStorage`,  
donde se ha producido la actualización.

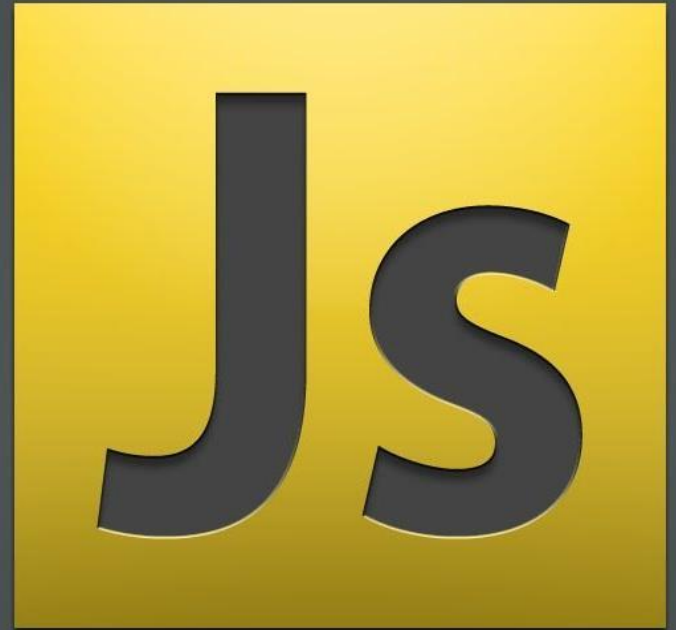
Este evento se dispara en todos los objetos window donde el almacenaje es accesible, **excepto en el que lo ha causado.**

```
window.onstorage = e => {  
  if (e.key !== 'now') return;  
  console.log(`${e.key}: ${e.newValue} en ${e.url}`)  
};  
  
localStorage.setItem('now', Date.now());
```

Este mecanismo es útil para que ventanas del mismo origen **puedan intercambiar mensajes.**

3

FECHA Y HORA



El objeto **Date** almacena la fecha y hora, así como proporciona diversos métodos para administrarlas.

Para crear un nuevo objeto Date se instancia con **new Date()**. Se pueden pasar los siguientes parámetros:

- **new Date()**: almacenará la fecha y hora actual
- **new Date(*timestamp*)**: milisegundos desde 01/01/1970 (Unix Time). Admite valores negativos para fechas anteriores
- **new Date(datestring)**: cadena con la fecha que será analizada y convertida a fecha automáticamente
- **new Date(año, mes, día, hora, mins, segs, ms)**

Para acceder a los componentes de fecha tenemos las siguientes funciones.

- **getFullYear():** devuelve el año
- **getMonth():** devuelve el mes
- **getDate():** devuelve el día del mes
- **getHours(), getMinutes(), getSeconds(), getMilliseconds()**
- **getDay():** devuelve el día de la semana partiendo de 0 (domingo) hasta 6 (sábado)
- **getTime():** devuelve el timestamp (Unix time) en msecs

Análogamente, hay una serie de funciones para establecer los componentes:

- **setFullYear(year, [month], [date])**
- **setMonth(month, [date])**
- **setDate(date)**
- **setHours(hour, [min], [sec], [ms])**
- **setMinutes(min, [sec], [ms])**
- **setSeconds(sec, [ms])**
- **setMilliseconds(ms)**
- **setTime(timestamp)**

Hay que tener en cuenta que se aplica la autocorrección para los objetos Date cuando fijamos valores fuera de rango.

```
let date = new Date(2013, 1, 32); // ¿32 de Enero 2013?  
console.log(date); // 1 de Febrero de 2013
```

También se pueden establecer valores en 0 o negativos

```
date=new Date(2020, 0, 2); // 2 de enero de 2020  
date=new Date(2020, -4, 2); // 2 de septiembre de 2019
```

Si convertimos un objeto Date a número toma el valor del *timestamp* actual.

```
let now = new Date();  
console.log(+now);           //1675147090001
```

Esto implica que se pueden restar fechas y nos dará su diferencia en milisegundos.

```
let start = new Date(); // comienza a medir  
  
// la función hace su trabajo  
for (let i = 0; i < 100000; i++) {  
    let doSomething = i * i * i;  
}  
let end = new Date(); // termina de medir el tiempo  
console.log(`El tiempo transcurrido es de ${end - start} ms`);
```



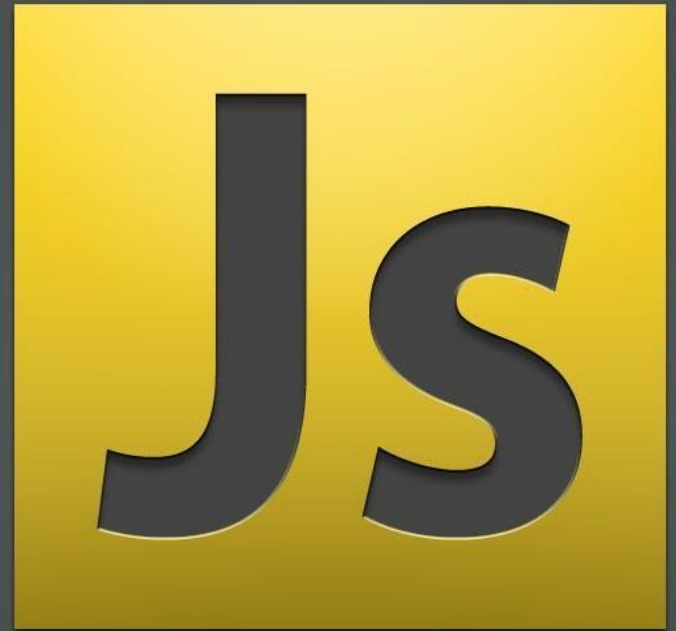
Si solo queremos obtener el *timestamp* actual (como en el ejemplo anterior) podemos utilizar el método **Date.now()**.

Este método es equivalente a **new Date().getTime()** pero sin necesidad de crear una instancia del objeto, siendo por tanto más eficiente al no afectar a la recolección de basura.

```
console.log(Date.now()); // 1675147546300
```

# 4

LIBRERÍAS:  
LEAFLET



Algo muy importante en cualquier lenguaje de programación es **no volver a programar aquello que ya ha programado otro.**

Este objetivo se consigue mediante las **librerías**, archivos de código que ya implementan funcionalidades que podemos utilizar en nuestros desarrollos web sin necesidad de volver a codificarlos.

Aunque pueden ser conceptos similares, no hay que confundir las **librerías** con los **frameworks**.

Una **librería** JavaScript es una pieza de código reutilizable que agrupa múltiples funciones/métodos/objetos generalmente con un fin común.

Ejemplo:

- **Leaflet**: manipulación de mapas
- **Lodash**: facilita el trabajo con arrays, objetos, strings, ...
- **jQuery**: manipulación del DOM, eventos, consultas AJAX, ...

Un **framework** es un conjunto de librerías que proporcionan código para realizar las tareas habituales de programación, pero proporcionando una estructura en torno al cual realizar el proyecto.

Un framework fuerza a realizar la aplicación siguiendo una determinada arquitectura.

Ejemplos:

- **Angular**
- **EmberJS**
- **VueJS**

Hay dos formas de incluir una librería en un documento HTML:

- Descargando el archivo js (disponible en la web de la propia librería) e insertándolo de la forma habitual mediante la etiqueta **<script>**
- Utilizando una **CDN** (*Content Delivery Network*), redes que realizan copias caché de archivos en servidores repartidos por todo el mundo para reducir tiempos de latencia.

Independientemente del método utilizado, deberá insertarse **antes** de nuestro script para tener disponibles las funciones cuando éste se ejecute.

La librería **Leaflet** proporciona una serie de funciones para insertar mapas interactivos en una página web.



an open-source JavaScript library  
for mobile-friendly interactive maps

[Overview](#) [Tutorials](#) [Docs](#) [Download](#) [Plugins](#) [Blog](#)

Si vamos al apartado de descargas podemos obtener el código fuente para guardarla directamente en nuestro servidor.

Nombre	Tamaño	Comprimido	Tipo	Modificado	CRC32
..			Carpeta de archivos		
images	6.503	6.503	Carpeta de archivos	18/04/2022 12:42	
leaflet.css	14.661	3.438	Documento de hoj...	18/04/2022 12:42	4936552F
leaflet.js	143.908	41.408	Archivo JavaScript	18/04/2022 12:43	D774C897
leaflet.js.map	196.318	71.620	Linker Address Map	18/04/2022 12:43	2C082D69
leaflet-src.esm.js	412.215	107.202	Archivo JavaScript	18/04/2022 12:43	9CEAFF6F
leaflet-src.esm.js.map	841.092	162.583	Linker Address Map	18/04/2022 12:43	AA1592B1
leaflet-src.js	437.177	109.082	Archivo JavaScript	18/04/2022 12:43	413C0D45
leaflet-src.js.map	841.184	163.763	Linker Address Map	18/04/2022 12:43	C03CF4BD

## Download Leaflet

Version	Description
<a href="#">Leaflet 1.9.3</a>	Stable version, released on November 18, 2022.
<a href="#">Leaflet 1.8.0</a>	Previous stable version, released on April 18, 2022.
<a href="#">Leaflet 2.0-dev</a>	In-progress version, developed on the main branch.



## leaflet.js

```

/* @preserve
 * Leaflet 1.8.0, a JS library for interactive maps. https://leafletjs.com
 * (c) 2010-2022 Vladimir Agafonkin, (c) 2010-2011 CloudMade
 */
!function(t,i){"object"==typeof exports&&"undefined"!=typeof module?exports=i:(("undefined"!=typeof globalThis?globalThis:t||self).leaf
length;n<o;n++)for(i in e=arguments[n])t[i]=e[i];return t}var R=Object.create||function(t){return N.prototype=t,new N};function N(){function a
(t,i){var e=Array.prototype.slice;if(t.bind)return t.bind.apply(t,e.call(arguments,1));var n=e.call(arguments,2);return function(){return t.
apply(i,n.length?n.concat(e.call(arguments)):arguments)}}var D=0;function h(t){return"_leaflet_id"in t||(t._leaflet_id=++D),t._leaflet_id}
function j(t,i,e){var n,o,s=function(){n=!1,o&&(r.apply(e,o),o=!1)},r=function(){n?o=arguments:(t.apply(e,arguments),setTimeout(s,i),n=!0)};
return r}function H(t,i,e){var n=i[1],i=i[0],o=n-i;return t===n&&e?((t-i)%o+o)%i:function u(){return!1}function e(t,i){if(!1===i)return t;
i=Math.pow(10,void 0===i?6:i);return Math.round(t*i)/i}function W(t){return t.trim?t.trim():t.replace(/^\s+|\s+$/g,"")}function F(t){return W
(t).split(/\s+/)}function c(t,i){for(var e in Object.prototype.hasOwnProperty.call(t,"options"))|(t.options=t.options?R(t.options):{}),i)t.
options[e]=i[e];return t.options}function U(t,i,e){var n,o=[];for(n in t)o.push(encodeURIComponent(e?n.toUpperCase():n)+"="+encodeURIComponent(t
[n]));return i&&-1!==i.indexOf("?")?"&":"?"+"o.join("&");var V=/\{ *([\w_ -]+) *\}/g;function q(t,e){return t.replace(V,function(t,i){i=e[i];if
(void 0===i)throw new Error("No value provided for variable "+t);return i="function"==typeof i?i(e):i})}var d=Array.isArray||function(t){return"
[object Array]"===Object.prototype.toString.call(t)};function G(t,i){for(var e=0;e<t.length;e++)if(t[e]===i)return e;return-1}var K="data:image/
gif;base64,R0lGODlhAQABAAD/ACwAAAAAAQABAAACADs=";function Y(t){return window["webkit"+t]||window["moz"+t]||window["ms"+t]}var X=0;function J(t)

```

## leaflet-src.js

```

/* @preserve
 * Leaflet 1.8.0, a JS library for interactive maps. https://leafletjs.com
 * (c) 2010-2022 Vladimir Agafonkin, (c) 2010-2011 CloudMade
 */

(function (global, factory) {
  typeof exports === 'object' && typeof module !== 'undefined' ? factory(exports) :
  typeof define === 'function' && define.amd ? define(['exports'], factory) :
  (global = typeof globalThis !== 'undefined' ? globalThis : global || self, factory(global.leaflet = {}));
})(this, (function (exports) { 'use strict';

  var version = "1.8.0";

  /*
   * @namespace Util
   *
   * Various utility functions, used by Leaflet internally.
   */

  // @function extend(dest: Object, src?: Object): Object
  // Merges the properties of the `src` object (or multiple objects) into `dest` object and returns the latter

```

La otra opción es utilizar un **CDN** para liberar de carga nuestro servidor y además obtener una menor latencia cuando el usuario se conecta desde una ubicación lejana.

### Using a Hosted Version of Leaflet

The latest stable Leaflet release is available on several CDN's — to start using it straight away, place this in the head of your HTML code:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" integrity="sha256-kLaT2GOSpl  
<script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js" integrity="sha256-WBkoXOWTeyKc10HuWtc+i2uENF|
```

Note that the [integrity hashes](#) are included for security when using Leaflet from CDN.

La otra opción es utilizar un **CDN** para liberar de carga nuestro servidor y además obtener una menor latencia cuando el usuario se conecta desde una ubicación lejana.

### Using a Hosted Version of Leaflet

The latest stable Leaflet release is available on several CDN's — to start using it straight away, place this in the head of your HTML code:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" integrity="sha256-kLaT2GOSpl  
<script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js" integrity="sha256-WBkoXOWTeyKc10HuWtc+i2uENF|
```

Note that the [integrity hashes](#) are included for security when using Leaflet from CDN.

El primer paso para insertar un mapa en nuestra página es **crear el espacio que lo va a contener.**

Este será un elemento `<div>` que deberá tener el **identificador map.**

Este `<div>` deberá tener definido un tamaño mediante CSS lo que determinará el tamaño del mapa.

```
<style>
  #map {
    width: 1280px;
    height: 640px;
  }
</style>

<div id="map"></div>
```

Ahora hay que crear el mapa.

Leaflet proporciona un objeto llamado **L** que tiene una serie de funciones para manipular el mapa. La mayoría de estas funciones devuelven el propio objeto mapa, **lo que permite encadenarlas**.

Ahora usaremos:

- **L.map('map')**: crea el objeto mapa y lo inserta en el elemento HTML con la etiqueta que se le pase como parámetro.
- **L.setView([lat, lon], zoom)**: establece la vista del mapa en las coordenadas indicadas con el nivel de zoom.

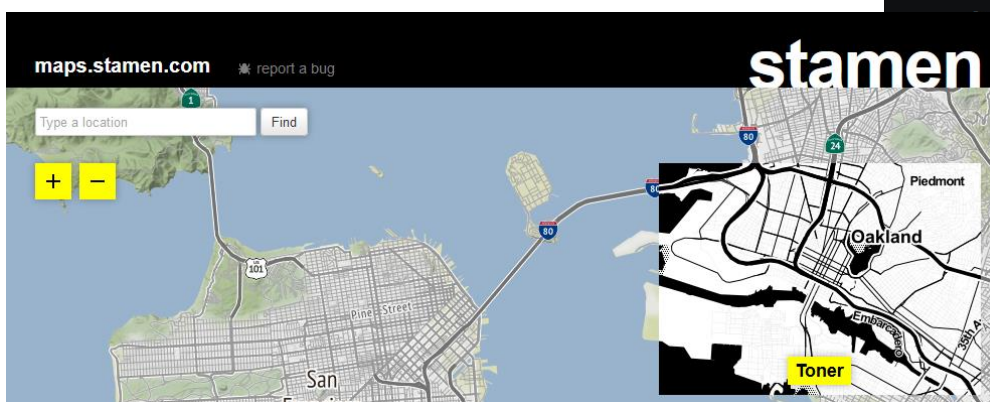
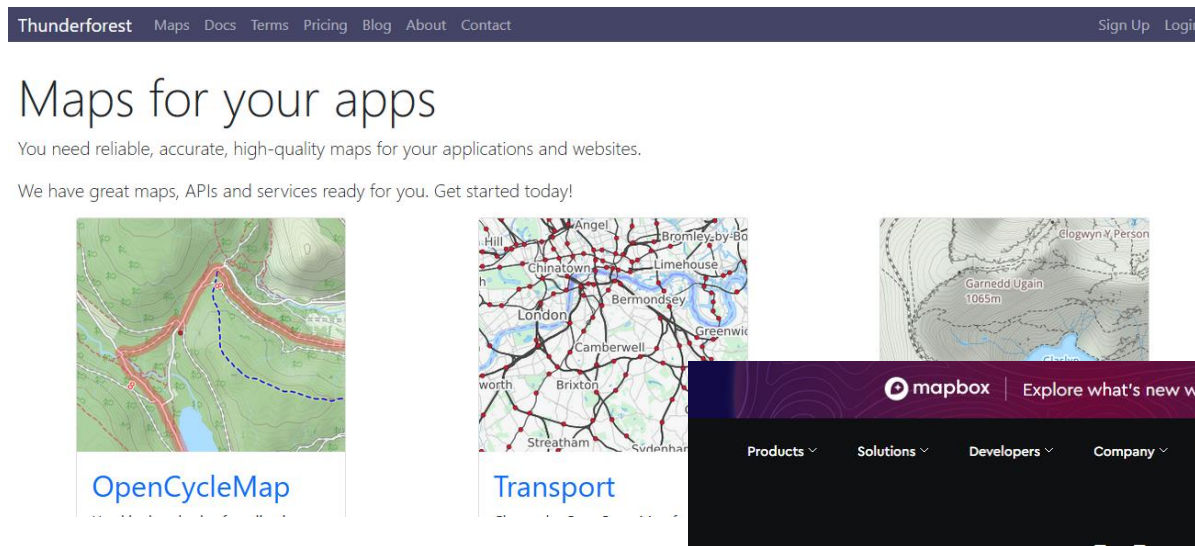
- **L.tileLayer(url, options).addTo(map)**: esta función indica de dónde se van a obtener los *tiles* del mapa. Por ejemplo, de OpenStreetMap. A continuación, lo añade al mapa.

```
let map = L.map('map')
    .setView([42.6000300, -5.570320], 13);
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png',
{
    maxZoom: 19,
    attribution: '&copy; <a
href="http://www.openstreetmap.orgcopyright">OpenStreetMap</a
>'
}).addTo(map);
```

- **L.tileLayer(url, options).addTo(map)**: esta función indica de dónde se van a obtener los *tiles* del mapa. Por ejemplo, de OpenStreetMap. A continuación, lo añade al mapa.

```
let map = L.map('map').setView([42.6000300, -5.570320], 13);
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png',
  {
    maxZoom: 19,
    attribution: '&copy; <a
href="http://www.openstreetmap.orgcopyright">OpenStreetMap</a
>'
  }).addTo(map);
```

Aparte de OpenStreetMap se pueden utilizar otros servicios de mapas online, como **Mapbox**, **Stamen** o **Thunderforest**



# Stamen Maps and location for developers

Precise location data and powerful developer tools to change the way we navigate the world.



## `L.marker([lat, lon], {options}).addTo(map)`

Añade un marcador al mapa.

Algunas de las propiedades de las opciones:

- **title**: *tooltip* que se mostrará al situar el cursor sobre él
- **opacity**: valor entre 0 y 1.0 que indica la opacidad del marcador.
- **riseOnHover**: animación al colocar el cursor sobre él
- **draggable**: si puede ser arrastrado

## `L.circle([lat, lon], {options}).addTo(map)`

Añade un círculo al mapa.

Algunas de las propiedades de las opciones:

- **color** y **fillColor**: cadena en formato CSS
- **fillOpacity**: opacidad
- **radius**: radio del círculo en metros
- **weight**: grosor del borde en píxeles

## Popups

Se puede añadir un popup a cualquiera de los elementos anteriores mediante la función **bindPopup()**. Este se mostrará al hacer click sobre el elemento o bien se mostrará abierto si añadimos la función **openPopup()**.

El parámetro que hay que pasarle a bindPopup es una cadena con código HTML que determinará el contenido del mismo.

```
var marker = L.marker([42.6000300, -5.570320], {  
  title: 'León',  
  opacity: 0.7,  
}).addTo(map);  
  
marker.bindPopup("<img src=./leon.jpg>").openPopup();
```



## **L.polyline(coords, options).addTo(map)**

Dibuja una polilínea que une las diferentes coordenadas que se le pasan como parámetro.

Las coordenadas se pasan como un array donde cada elemento es a su vez otro array con la latitud y la longitud del punto correspondiente.

Algunas opciones:

- **smoothFactor**: un mayor valor significa mejor rendimiento y una apariencia más suavizada, pero menor precisión.
- **color**
- **weight**

```
L.polyline([[42.70, -5.60], [42.800, -5.80], [42.750, -5.55], [42.7350, -5.5532] ], {  
  color: 'blue',  
  weight: 10,  
}).addTo(map);
```

