**JavaScript Cookie**

A simple, lightweight JavaScript API for handling cookies

* Works in [all](https://saucelabs.com/u/js-cookie) browsers
* Accepts [any](https://github.com/js-cookie/js-cookie#encoding) character
* [Heavily](https://github.com/js-cookie/js-cookie/blob/master/test) tested
* No dependency
* [Unobtrusive](https://github.com/js-cookie/js-cookie#json) JSON support
* Supports AMD/CommonJS
* [RFC 6265](https://tools.ietf.org/html/rfc6265) compliant
* Useful [Wiki](https://github.com/js-cookie/js-cookie/wiki)
* Enable [custom encoding/decoding](https://github.com/js-cookie/js-cookie#converters)
* **~900 bytes** gzipped!

**If you're viewing this at**[**https://github.com/js-cookie/js-cookie**](https://github.com/js-cookie/js-cookie)**, you're reading the documentation for the master branch.**[**View documentation for the latest release.**](https://github.com/js-cookie/js-cookie/tree/latest#readme)

**Build Status Matrix**

**Installation**

**Direct download**

Download the script [here](https://github.com/js-cookie/js-cookie/blob/latest/src/js.cookie.js) and include it (unless you are packaging scripts somehow else):

<script src="/path/to/js.cookie.js"></script>

**Do not include the script directly from GitHub (**[**http://raw.github.com/**](http://raw.github.com/)**...).** The file is being served as text/plain and as such being blocked in Internet Explorer on Windows 7 for instance (because of the wrong MIME type). Bottom line: GitHub is not a CDN.

**Package Managers**

JavaScript Cookie supports [npm](https://www.npmjs.com/package/js-cookie) and [Bower](http://bower.io/search/?q=js-cookie) under the name js-cookie.

**Module Loaders**

JavaScript Cookie can also be loaded as an AMD, CommonJS or [ES6](https://github.com/js-cookie/js-cookie/issues/233#issuecomment-233187386) module.

**Basic Usage**

Create a cookie, valid across the entire site:

Cookies.set('name', 'value');

Create a cookie that expires 7 days from now, valid across the entire site:

Cookies.set('name', 'value', { expires: 7 });

Create an expiring cookie, valid to the path of the current page:

Cookies.set('name', 'value', { expires: 7, path: '' });

Read cookie:

Cookies.get('name'); // => 'value'

Cookies.get('nothing'); // => undefined

Read all visible cookies:

Cookies.get(); // => { name: 'value' }

Delete cookie:

Cookies.remove('name');

Delete a cookie valid to the path of the current page:

Cookies.set('name', 'value', { path: '' });

Cookies.remove('name'); // fail!

Cookies.remove('name', { path: '' }); // removed!

*IMPORTANT! when deleting a cookie, you must pass the exact same path and domain attributes that was used to set the cookie, unless you're relying on the*[*default attributes*](https://github.com/js-cookie/js-cookie#cookie-attributes)*.*

*Note: Removing unexisting cookie does not raise any exception nor return any value*

**Namespace conflicts**

If there is any danger of a conflict with the namespace Cookies, the noConflict method will allow you to define a new namespace and preserve the original one. This is especially useful when running the script on third party sites e.g. as part of a widget or SDK.

// Assign the js-cookie api to a different variable and restore the original "window.Cookies"

var Cookies2 = Cookies.noConflict();

Cookies2.set('name', 'value');

*Note: The .noConflict method is not necessary when using AMD or CommonJS, thus it is not exposed in those environments.*

**JSON**

js-cookie provides unobtrusive JSON storage for cookies.

When creating a cookie you can pass an Array or Object Literal instead of a string in the value. If you do so, js-cookie will store the string representation of the object according to JSON.stringify:

Cookies.set('name', { foo: 'bar' });

When reading a cookie with the default Cookies.get api, you receive the string representation stored in the cookie:

Cookies.get('name'); // => '{"foo":"bar"}'

Cookies.get(); // => { name: '{"foo":"bar"}' }

When reading a cookie with the Cookies.getJSON api, you receive the parsed representation of the string stored in the cookie according to JSON.parse:

Cookies.getJSON('name'); // => { foo: 'bar' }

Cookies.getJSON(); // => { name: { foo: 'bar' } }

*Note: To support IE6-7 (*[*and IE 8 compatibility mode*](http://stackoverflow.com/questions/4715373/json-object-undefined-in-internet-explorer-8)*) you need to include the JSON-js polyfill:*[*https://github.com/douglascrockford/JSON-js*](https://github.com/douglascrockford/JSON-js)

**Encoding**

This project is [RFC 6265](http://tools.ietf.org/html/rfc6265#section-4.1.1) compliant. All special characters that are not allowed in the cookie-name or cookie-value are encoded with each one's UTF-8 Hex equivalent using [percent-encoding](http://en.wikipedia.org/wiki/Percent-encoding).  
The only character in cookie-name or cookie-value that is allowed and still encoded is the percent % character, it is escaped in order to interpret percent input as literal.  
Please note that the default encoding/decoding strategy is meant to be interoperable [only between cookies that are read/written by js-cookie](https://github.com/js-cookie/js-cookie/pull/200#discussion_r63270778). To override the default encoding/decoding strategy you need to use a [converter](https://github.com/js-cookie/js-cookie#converters).

**Cookie Attributes**

Cookie attributes defaults can be set globally by setting properties of the Cookies.defaults object or individually for each call to Cookies.set(...) by passing a plain object in the last argument. Per-call attributes override the default attributes.

**expires**

Define when the cookie will be removed. Value can be a [Number](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number) which will be interpreted as days from time of creation or a [Date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date) instance. If omitted, the cookie becomes a session cookie.

To create a cookie that expires in less than a day, you can check the [FAQ on the Wiki](https://github.com/js-cookie/js-cookie/wiki/Frequently-Asked-Questions#expire-cookies-in-less-than-a-day).

**Default:** Cookie is removed when the user closes the browser.

**Examples:**

Cookies.set('name', 'value', { expires: 365 });

Cookies.get('name'); // => 'value'

Cookies.remove('name');

**path**

A [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String) indicating the path where the cookie is visible.

**Default:** /

**Examples:**

Cookies.set('name', 'value', { path: '' });

Cookies.get('name'); // => 'value'

Cookies.remove('name', { path: '' });

**Note regarding Internet Explorer:**

Due to an obscure bug in the underlying WinINET InternetGetCookie implementation, IE’s document.cookie will not return a cookie if it was set with a path attribute containing a filename.

(From [Internet Explorer Cookie Internals (FAQ)](http://blogs.msdn.com/b/ieinternals/archive/2009/08/20/wininet-ie-cookie-internals-faq.aspx))

This means one cannot set a path using path: window.location.pathname in case such pathname contains a filename like so: /check.html (or at least, such cookie cannot be read correctly).

**domain**

A [String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String) indicating a valid domain where the cookie should be visible. The cookie will also be visible to all subdomains.

**Default:** Cookie is visible only to the domain or subdomain of the page where the cookie was created, except for Internet Explorer (see below).

**Examples:**

Assuming a cookie that is being created on site.com:

Cookies.set('name', 'value', { domain: 'subdomain.site.com' });

Cookies.get('name'); // => undefined (need to read at 'subdomain.site.com')

**Note regarding Internet Explorer default behavior:**

Q3: If I don’t specify a DOMAIN attribute (for) a cookie, IE sends it to all nested subdomains anyway?  
A: Yes, a cookie set on example.com will be sent to sub2.sub1.example.com.  
Internet Explorer differs from other browsers in this regard.

(From [Internet Explorer Cookie Internals (FAQ)](http://blogs.msdn.com/b/ieinternals/archive/2009/08/20/wininet-ie-cookie-internals-faq.aspx))

This means that if you omit the domain attribute, it will be visible for a subdomain in IE.

**secure**

Either true or false, indicating if the cookie transmission requires a secure protocol (https).

**Default:** No secure protocol requirement.

**Examples:**

Cookies.set('name', 'value', { secure: true });

Cookies.get('name'); // => 'value'

Cookies.remove('name', { secure: true });

**Converters**

**Read**

Create a new instance of the api that overrides the default decoding implementation.  
All get methods that rely in a proper decoding to work, such as Cookies.get() and Cookies.get('name'), will run the converter first for each cookie.  
The returning String will be used as the cookie value.

Example from reading one of the cookies that can only be decoded using the escape function:

document.cookie = 'escaped=%u5317';

document.cookie = 'default=%E5%8C%97';

var cookies = Cookies.withConverter(function (value, name) {

if ( name === 'escaped' ) {

return unescape(value);

}

});

cookies.get('escaped'); // 北

cookies.get('default'); // 北

cookies.get(); // { escaped: '北', default: '北' }

**Write**

Create a new instance of the api that overrides the default encoding implementation:

Cookies.withConverter({

read: function (value, name) {

// Read converter

},

write: function (value, name) {

// Write converter

}

});

**Server-side integration**

Check out the [Servers Docs](https://github.com/js-cookie/js-cookie/blob/master/SERVER_SIDE.md)

**Contributing**

Check out the [Contributing Guidelines](https://github.com/js-cookie/js-cookie/blob/master/CONTRIBUTING.md)

**Security**

For vulnerability reports, send an e-mail to jscookie at gmail dot com

**Manual release steps**

* Increment the "version" attribute of package.json
* Increment the version number in the src/js.cookie.js file
* Commit with the message "Release version x.x.x"
* Create version tag in git
* Create a github release and upload the minified file
* Change the latest tag pointer to the latest commit
  + git tag -f latest
  + git push <remote> :refs/tags/latest
  + git push origin master --tags
* Release on npm

**Authors**

* [Klaus Hartl](https://github.com/carhartl)
* [Fagner Brack](https://github.com/FagnerMartinsBrack)
* And awesome [contributors](https://github.com/js-cookie/js-cookie/graphs/contributors)