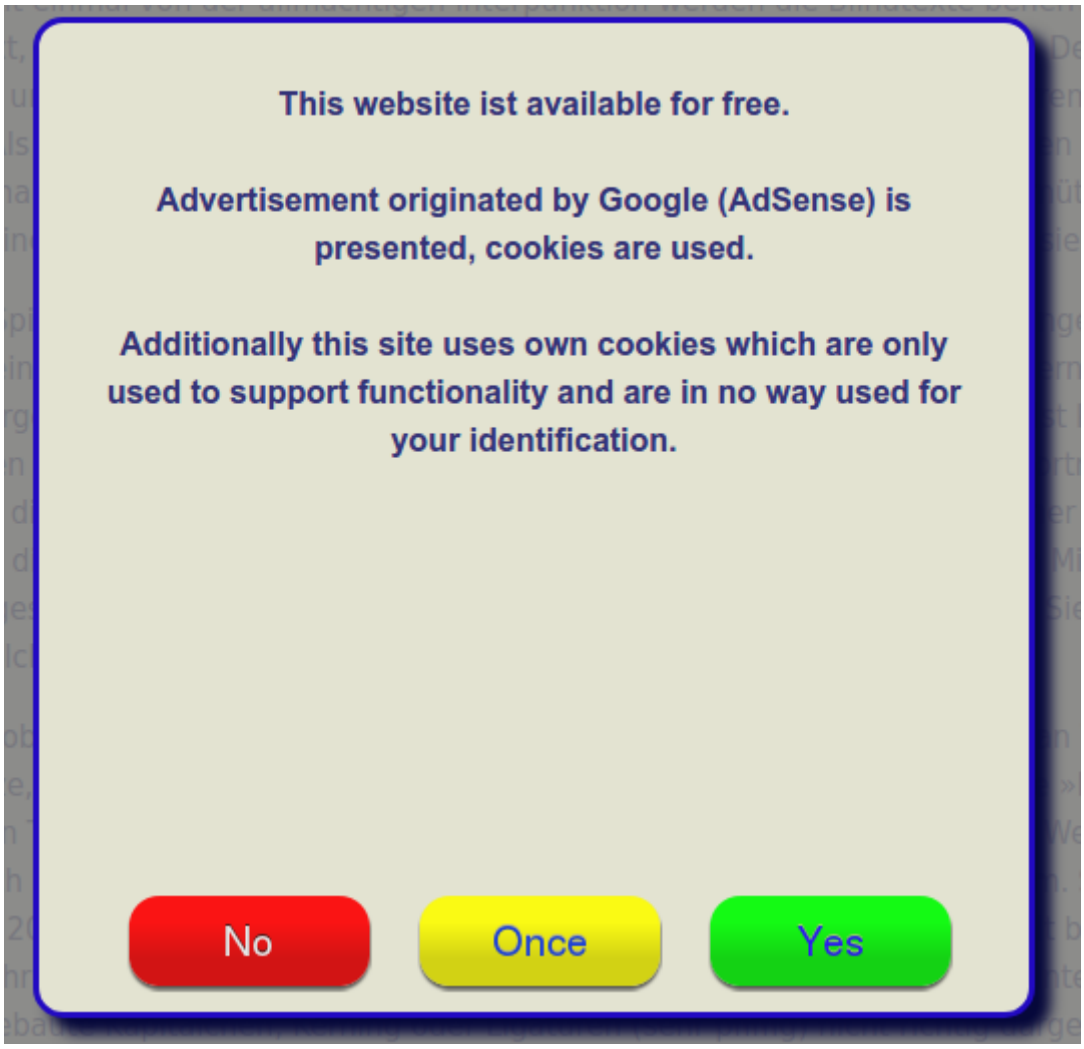
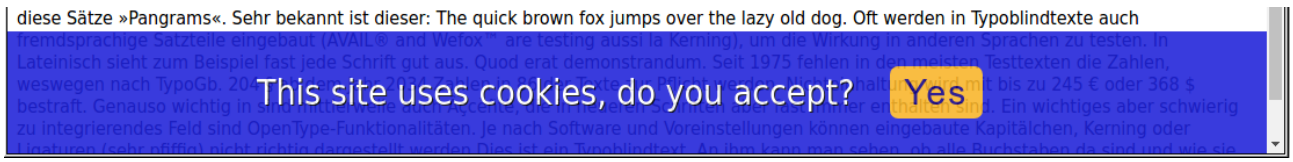
CookieQuestion

A free (CC BY-SA 4.0), ready-to-use, lightweight, adjustable JavaScript framework to fulfill legal requirements regarding user information and acknowledge





Contents

1. Motivation

Google requires web page providers that use Google AdSense and/or DoubleClick to explicitly inform their web page visitors in the European Union (EU) and acknowledge their agreement.

The EU Commission has released a so called „Cookie-Guideline“ that also requires to inform users about Cookies used on web pages.

It may lead to inconveniences not to follow these rules.

The CookieQuestion framework keeps the task to fulfill these rules simple – just by adding a single line to the web page source. It features

- free use
- easy to setup functionality (wait for acknowledgement, require acknowledgement, redirect to alternative web page w/o cookies, etc.)
- design and reactions like blending adjustable
- 10 predefined, ready-to-use themes included
- tested with most modern browsers (see Tests)
- support for old-style frameset-frames
- requires few changes to existing web pages, only a single line has to be included
- multi-language support
- no use of 3rd party frameworks

2. Quickstart

a) Extract the archive

b) Copy the directory *cookieQuestion.min* to any file location accessible by the web pages and rename to *cookieQuestion*.

c) Include the line
<script src="[absolute or relative url-path to …/cookieQuestion/cookieQuestionStarter.js]"></script>
into all web pages that use cookies, anywhere inside the head or body section of the page (but not inside of other HTML tags than head or body).

For a live demo you are welcome to visit https://ContemporaryCodes.com/.

3. How it works

CookieQuestion stores the acceptance of cookies in a cookie itself, i.e. the absence or presence of the cookieQuestion internal cookie is relevant for displaying a cookie question. The visual parts of the cookie question are dynamically inserted and removed from the HTML document DOM by means of JavaScript. The visual appearance and behavior (like blending etc.) are all capsuled inside the theme classes (like CookieQuestionTheme_Massive etc.). They can be styled by direct element styles or (dynamically created) sytle sheets. Since the visual cookie Elements are given an element id it is possible to use external style sheets too. All pages that are subject to cookie acceptance must include the cookieQuestion framework using a line like e.g.
<script src="/cookieQuestion/cookieQuestionStarter.js"></script>
or
<script src="../cookieQuestion/cookieQuestionStarter.js" data-theme="Minimal" data-yesUrl="contentPage.html" data-noUrl="noCookiesPage.html" data-cookieName="cookiesAcceptedFlag"></script>

4. Detailed setup

a) Cookie theme
The cookie theme encapsulates the visual appearance of the cookie question. All themes are located in the directory *cookieQuestion/themes* and must follow the naming
QookieQuestionTheme_[name of theme].js. The *name of theme*, case sensitive, is set globally in the file *cookieQuestion/cookieQuestionStarter.js* in the line
var themeToUse = "name of theme";
The theme to use can also be specified on a per page base inside the including script tag using
data-theme=”[pure theme name to use]”
where *[pure theme name to use]* may be one of the predefined theme names *Massive, Massive2, Minimal, Minimal2, MinimalSmart, MinimalSmartG, Shy, ShyBut, ShySmart, Simple* or the name of a self-defined theme.

b) Behavior if cookies not accepted
If the used theme supports to reject the use of cookies (answer *no*), the default behavior is to go back to the previous page in browser history or, if such is not available, to open the *about:blank* page. This behavior can be changed to display a specific page that does not use cookies. Such no-cookie-page can be specified on a per theme base in the file *cookieQuestion/themes/CookieQuestionTheme_[name of theme].js* in the line
this.locationNoCookies = “[absolute or relative path to the no-cookies page]”;
That location can also be specified on a per page base inside the including script tag using
data-noUrl=”[absolute or relative path to the no-cookies page]”
Specifying an empty String forces the default behavior, but an empty String inside the including script tag does not override a different specified location in the theme file.

c) Behaviour if cookies accepted
If the used theme supports to redirect to a page once cookies have been accepted (answer *yes*) – the default behavior is to just remove the cookie question and stay on the page on which the cookie question was displayed – that page can be specified on a per theme base in the file *cookieQuestion/themes/CookieQuestionTheme_[name of theme].js* in the line
this.locationCookieContinue = “[absolute or relative path to the cookies accepted page]”;
That location can also be specified on a per page base inside the including script tag using
data-yesUrl=”[absolute or relative path to the cookies accepted page]”
Specifying an empty String forces the default behavior, but an empty String inside the including script tag does not override a different specified location in the theme file.

d) Flag cookie name
The flag cookie that flags if cookies have been accepted is by default named ContemporaryCodes.com_CookieQuestion. This default name can be changed on global scope in the file *cookieQuestion/themes/CookieQuestionTheme.js* in the line
this.cookieName = cookieNameTmp ? cookieNameTmp : "ContemporaryCodes.com_CookieQuestion";
That name can also be specified on a per page base inside the including script tag using
data-cookieName="[name of the cookie]"
All pages that use the same cookie permission must use the same cookie name.

e) Flag cookie lifetime
The flag cookie that flags if cookies have been accepted may be a session cookie (answer *once*) that is valid until the browser session ends or a cookie with a dedicated life time (answer *yes*). Please note that some browsers may have problems to store non-session cookies, see e.g. https://answers.microsoft.com/en-us/windows/forum/apps_windows_10-msedge/microsoft-edge-blocking-cookies-windows-10/3d865aca-a21e-445d-9804-d45a5d02efa2. Browsers running in private or incognito mode usually take all cookies as session cookies. The lifetime of a regular cookie (vs. session cookie) is defined on a global base in the file *cookieQuestion/themes/CookieQuestionTheme.js* in the line
this.yesCookiesDays = 100;
CookieQuestion can be setup to store all cookies as session cookies by changing *false* to *true* in the file *cookieQuestion/themes/CookieQuestionTheme.js* in the line
this.yesMeansOnce = false;

f) Language
The language to use is detected by the *language* property value of the *navigator* variable and can therefor be selected by the browser‘s preferred language setting.
Inside the theme files languages can be added by extending the JavaScript variable **this.langData**. Not existing entries default to the english version.
Please be aware that all JavaScript (*.js*) files use UTF-8 charset encoding (Unicode charset). Especially Windows© users should take care to use an UTF-8 capable editor when modifying or adding files.
Predefined languages are English (*en*) as default and German (*de*).

g) Frameset/frame support
Such frames are not anymore part of the standards, but are still (limited) supported by cookieQuestion: To show a cookie question on such framed web page the frame-page must be included like
<frame id="cookieQuestionFrame" …>

5. Creating own themes

One is, according to the license, free to extend or modify the given existing source code to own likes. This chapter describes the implementation used in the predefined themes.

To create a new theme it is recommended to copy an existing theme, giving it it's own, unique theme name, and modify that.

First modify the class definition line
`class CookieQuestionTheme_[name of theme] extends CookieQuestionTheme {`
to contain the new *name of theme*, identical to it's filename w/o filename extension.

The HTML code that displays the cookie question is assembled dynamically in the theme file, *CookieQuestionTheme_[theme name].js*, in the method named **`displayQuestion`**.

That method uses the style definitions **`this.css`** and/or **`this.styleRules`** located in the upper part of the theme file. If using pseudo classes like e.g. *button.cookieQuestionYes:focus* the use of *styleRules* is a must. *styleRules* do not work on obsolescent frameset frames. All style definitions that could be noted either in *css* or in *style rules* may be specified inside the *css* object.

A second duty of the **`displayQuestion`** method is to bind the methods **`cookiesYes`** and optionally **`cookiesOnce`** and **`cookiesNo`** to the HTML buttons or other elements that trigger permanent acceptance, single acceptance and rejected acceptance.

The method **`removeQuestion`** is responsible for removing the cookie question visually and should also delete the, afterwards invisible, cookie question HTML element(s) and optionally used event listeners. Special effects like blending out etc. should be triggered from within **`removeQuestion`**.

Available localizations, i.e. texts in different languages, should replace or extend the **`langData`** object, which must contain 2-letter lower case language codes, each of them is an object of message id strings in the denoted language. The message id string values are arrays of text, each array element is rendered as a single paragraph.

In case a theme file uses resources like images, as e.g. the predefined *Massive* theme does, they should all be placed in the *cookieQuestion/themes/resources* directory. The path to that directory is inside a theme file available through the **`this.resourcesUrlS`** variable, which includes a postfix-slash.

The final version of a cookie theme should be minimized. A useful tool for minimizing is the *babel* program. Minimizing not only reduces resources but also enlarges the number of browsers that can handle the code. If the minimized version of a file *filename.js* is named *filename.min.js* the rewrite definition inside the *.htaccess* file, for apache web server, in the *cookieQuestion.themes* directory is a sample of one way to automatically deliver the *.min.js* file if it is present and otherwise deliver the *.js* file. Minimizing can be done on https://ContemporaryCodes.com/jsmin/.

6. Hints

- For testing do not use browsers running in private or incognito mode. Several browsers support quick cookie deletion via *[Ctrl]+[Shift]+[Del]*.
- Changing the boolean value of the variable **`this.yesMeansOnce`** inside *cookieQuestion/themes/CookieQuestionTheme* will force creation of session cookies and therefor make the cookie question appear during every new session revisit of the web page. This may be helpful for testing.
- Some browsers, as seen with Microsoft edge instances, may not accept cookies permanently, but only for the browsing session. About problems regarding the edge browser please see https://answers.microsoft.com/en-us/windows/forum/apps_windows_10-msedge/microsoft-edge-blocking-cookies-windows-10/3d865aca-a21e-445d-9804-d45a5d02efa2.
- For the purpose to use self provided stylesheets one needs the id's of the visual components of cookie Question. Those are provided as 3rd argument to the *StyleELement.applyCss* method inside the *displayQuestion* method in the theme files. Since *applyCss* applies direct element styles external style definitions must use an *!important* postfix to get applied.
- Links in cookie question always use the target *_blank* (new page/tab) and follow the syntax **`>>>[link destination]<<<[link text]`**, for an example see CookieQuestionThemeMinimalSmartG.js.
- To not reduce the set of supported browsers always minimized JavaScript should be delivered. A free JavaScript minimizer is available at https://ContemporaryCodes.com/jsmin/
- The fall back, default language to use is defined in the **`setLang`** method inside the theme files.
- Using frameset/frames themes should direct format the styles using the **`css`** variable, style sheets may not be injectable.
- JavaScript file paths are relative to the location of the including HTML page.
- To create data URL inline images as used with the *Massive2* theme the data URL generator at https://ContemporaryCodes.com/dataUrl/ may be used.

7. Tests

The CookieQuestion framework was successfully tested with following browsers:

Chrome (mobile)	version 58	Android 5.0.1
Chrome	version 59	Linux
Chrome	version 59	Windows 10
Firefox (mobile)	version 53	Android 5.0.1
Firefox	version 53	Linux
Firefox	version 53	Windows 10
Internet Explorer	version 11	Windows 10
MS edge	version 40	Windows 10
Opera	version 45	Linux
Opera	version 45	Windows 10
Safari (mobile)	version 6.0	iPad 3/6.1
Safari	version 6.2	Mac OSX 10.8

8. Contact

Email: swdev@web.de

Please feel free to contact if you like individual support for any task regarding cookieQuestion or if you need a proprietary license, optionally including support.

9. License

The JavaScript framework cookieQuestion is licensed under the creative commons license *CC BY-SA 4.0:*

You are free to:

- **Share:** copy and redistribute the material in any medium or format
- **Adapt:** remix, transform, and build upon the material
- **For any purpose**, even commercially.

Under the following terms:

- **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike:** If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions:** You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

See the included file *Creative Commons — Attribution-ShareAlike 4.0 International — CC BY-SA 4.pdf* for details.

Aside of the *CC BY-SA 4.0* license the author keeps the right to license the JavaScript framework *cookieQuestion* under different conditions.