

Google+ JavaScript API

The following methods and classes are useful when working with the [Google+ plugins](#), [Google+ Sign-In button](#) and the [HTTP APIs](#). The full [Google JavaScript Client Library reference](#) contains additional methods and classes.

Loading the JavaScript API

For most of the [plugins](#), your site only needs to include the Google+ JavaScript API ([platform.js](#)). The Google+ JavaScript API can be loaded with the following script tag:

```
<script src="https://apis.google.com/js/platform.js" async defer>
</script>
```

If you are building an app that uses the [sign-in button](#), [interactive posts](#) or uses REST web services, you should also load the client in your script tag:

```
<script src="https://apis.google.com/js/client:platform.js" async defer>
</script>
```

Deferred execution with onLoad and script tag parameters

Use the [onload](#) callback to execute widget code after all dependencies have loaded.

To specify [script tag parameters](#), use the following syntax:

```
<script >
  window.__gcf = {
    lang: 'zh-CN',
    parsetags: 'onload'
  };
</script>
<script src="https://apis.google.com/js/platform.js" async defer></script>
```

Specifying an onload callback

You can define an [onload](#) callback that will execute when the script has successfully loaded. This callback can be helpful to ensure that your scripts run at the appropriate time.

```
<script src="https://apis.google.com/js/platform.js?onload=onLoadCallback" async defer>
</script>
```

When you load the script synchronously, your [onload](#) callback function must be defined prior to loading the script above.

Specifying the publisher ID

When you use the [mobile web recommendations](#), you must pass your Google+ Page ID to the JavaScript API by using the [publisherid](#) query parameter:

```
<script src="https://apis.google.com/js/platform.js?publisherid={{PAGE_ID}}" async defer>
</script>
```

You would replace `{{PAGE_ID}}` with the numeric ID that can be found in the URL to your Google+ Page. For example:

```
https://apis.google.com/js/platform.js?publisherid=110967630299632321627
```

Loading the JavaScript API

Deferred execution with [onLoad](#) and script tag parameters

Specifying an [onload](#) callback

Specifying the publisher ID

Configuration options

Methods

`gapi.auth.authorize(parameters, callback)`
`gapi.auth.checkSessionState(sessionParams, callback)`
`gapi.auth.getToken(key, returnError)`
`gapi.auth.setToken(key, token)`
`gapi.auth.signIn(parameters)`
`gapi.auth.signOut()`
`gapi.client.load(name, version, callback)`
`gapi.client.request(args)`
`gapi.client.setApiKey(apiKey)`

Classes

`OAuth 2.0 Token Object`
`gapi.client.HttpRequest`

Plugin methods

`gapi.community.*` (Community badge)
`gapi.follow.*` (Follow button)
`gapi.interactivepost.*` (Interactive posts)
`gapi.page.*` (Page badge)
`gapi.person.*` (Profile badge)
`gapi.plusone.*` (+1 button)
`gapi.plus.*` (Share button)
`gapi.post.*` (Embedded posts)
`gapi.signin.*` (Sign in)

If your Google+ Page uses a custom URL, you can find the Google+ Page ID by inspecting one of the Google+ Page's posts. You can copy the link from the time stamp on the post to extract the numeric ID:

Copy the URL from a post and extract the ID shown below



Configuration options

You can specify options for how to handle the loading of plugins and the language to use for the plugins. These options must be defined within a `<script />` element prior to loading the JavaScript API or be defined within the same `<script />` element that loads the JavaScript API, for example:

```
<script>
  // Defines the configuration options, these must be set before you load
  // the platform.js file.
  window.__gcfg = {
    lang: 'en-US',
    parsetags: 'explicit' // DS?: can the onload param be removed from the line below and added here?
                          // E.g., 'explicit','onload'
  }
</script>
<script src="https://apis.google.com/js/platform.js?onload=onLoadCallback" async defer></script>
```

lang - type: string

Sets the language to use for all of the plugins on the page. See the full list of [supported languages](#).

parsetags - type: string

Sets the loading mechanism to use.

- **onload:** All plugins on the page are automatically rendered after the page loads.
- **explicit:** Plugins are rendered only with explicit calls to their respective `go` or `render` methods. When you use the explicit load in conjunction with `go` and `render` calls that point to specific containers in your page, you prevent the script from traversing the entire DOM, which can improve button rendering time.

Other Google JavaScript APIs also use this object to define configuration options, such as [Google Analytics's social analytics](#).

Methods

These methods listed below are a subset of the [Google JavaScript Client Library](#) that are useful when working with Google+ features. For example, getting and setting tokens that are returned by the [Google+ Sign-In button](#) and then performing subsequent client-side API calls by using the `gapi.client.request()` method.

gapi.auth.authorize(parameters, callback)

Initiates the OAuth 2.0 authorization process. The user will receive a popup window asking them to authenticate and authorize (users need to have popup blockers disabled). Once they complete the authorization process, the popup will close and the callback function will fire.

Warning: The [over-the-air installs](#) feature does not work with the `gapi.auth.authorize()` method. If you want to take advantage of this feature, wherever possible, use the [sign-in button markup](#) or `gapi.signin.render()` instead.

Arguments:

parameters - type: object

A key/value map of parameters for the request. If the key is not one of the expected OAuth 2.0 parameters (see below), it is added to the URI as a query parameter. Valid keys for OAuth 2.0 parameters include:

client_id - type: string

The application's client ID. Visit the [Google Developers Console](#) to get an OAuth 2.0 client ID.

immediate - type: boolean

If `true`, then the sign-in process uses "immediate mode", which means that the token is refreshed behind the scenes, and no UI is shown to the user.

response_type - type: string

The OAuth 2.0 response type property. Default: `token`.

scope - type: array

The auth scope or scopes to authorize. Auth scopes for individual APIs can be found in their documentation or in the [OAuth 2.0 playground](#).

callback - type: function

The function to call once the sign-in process is complete. The function takes an [OAuth 2.0 token object](#) as its only parameter.

`gapi.auth.checkSessionState(sessionParams, callback)`

Checks if the user has a valid Google session by checking that the client ID and the session data that was created when the user first authorized is still valid. This method can help identify if a user signed out of Google, signed in to a different Google account, or reauthenticates with Google. If you pass `null` to `sessionParams.session_state`, you can check if the user is signed in to Google, whether or not they have previously authorized your app.

Arguments:

`sessionParams` - type: object

An object, which must include the following parameters.

`client_id` - type: string

The client ID of the app. Required.

`session_state` - type: string

The `session_state` from the OAuth 2.0 token or null if you want to only determine if the user is signed in to Google.

`callback` - type: function

The function to call when check is complete. The function is passed an argument to represent the user's session state. If the argument is true, the supplied `session_state` is still valid. If the argument is false, it is no longer valid and the application should perform a new [re-authorization flow](#) with `immediate` set to `true` to get an up-to-date access token or detect a non-authorized or nonexistent Google sign-in session.

`gapi.auth.getToken(key, returnError)`

Retrieves the OAuth 2.0 token for the application.

Arguments:

`key` - type: string

Optional identifier that is used to retrieve the `token` object. Default: `token`.

`returnError` - type: boolean

Optional flag. If `true`, an object is returned that contains an error property, which describes the error state. Default: `false`.

Returns: type: object. The OAuth 2.0 token. See [OAuth 2.0 token](#) below.

`gapi.auth.setToken(key, token)`

Sets the OAuth 2.0 token for the application. This function can also be called using a single argument without the key,

`gapi.auth.setToken(token)`.

Arguments:

`key` - type: string

Optional identifier used to store and later retrieve the `token` object. Use this argument when you need to be able to store multiple tokens. Default: `token`.

`token` - type: object

The token to set. See [OAuth 2.0 token](#) below.

`gapi.auth.signIn(parameters)`

Initiates the client-side Google+ Sign-In OAuth 2.0 flow. Similar to `gapi.auth.authorize()` except that this method supports the advanced Google+ Sign-In features, including [over-the-air installs](#) of Android apps. This method is a JavaScript alternative to using the [Google+ Sign-In button](#) widget.

Arguments:

`parameters` - type: object

An object, which must include the sign-in parameters. For the full list of parameters, see the [sign-in button attributes](#). When the method is called, the OAuth 2.0 authorization dialog is displayed to the user and when they accept, the `callback` function is called.

For example:

Example:

```
function initiateSignIn() {
  var myParams = {
    'clientid' : 'xxxxxxxxxxxxxxxx..apps.googleusercontent.com',
    'cookiepolicy' : 'single_host_origin',
    'callback' : 'mySignInCallback',
    'scope' : 'https://www.googleapis.com/auth/plus.login',
    'requestvisibleactions' : 'http://schema.org/AddAction'
    // Additional parameters
  }
```

```
};  
gapi.auth.signIn(myParams);  
}
```

`gapi.auth.signOut()`

Signs a user out of your app without logging the user out of Google. This method will only work when the user is signed in with Google+ Sign-In.

For more information, see [Sign-out](#).

`gapi.client.load(name, version, callback)`

Loads the client library interface to a particular API. The new API interface will be in the form `gapi.client.api.collection.method`. For example, the Moderator API would create methods like `gapi.client.moderator.series.list`.

Arguments:

name - **type:** **string**

The name of the API to load.

version - **type:** **string**

The version of the API to load.

callback - **type:** **function**

Optional function that is called once the API interface is loaded.

`gapi.client.request(args)`

Creates a HTTP request for making RESTful requests.

Arguments:

args - **type:** **object**

An object encapsulating the various arguments for this method. The path is required, the rest are optional. The values are described in detail below.

path - **type:** **string**

The URL to handle the request.

method - **type:** **string**

The HTTP request method to use. Default is `GET`.

params - **type:** **Object**

URL params in key-value pair form.

headers - **type:** **Object**

Additional HTTP request headers.

body - **type:** **string**

The HTTP request body (applies to PUT or POST).

callback - **type:** **Function**

If supplied, the request is executed immediately and no `gapi.client.HttpRequest` object is returned.

Returns: **type:** `gapi.client.HttpRequest|undefined`. If no callback is supplied, a request object is returned. The request can then be executed via `gapi.client.HttpRequest.execute(callback)`.

`gapi.client.setApiKey(apiKey)`

Sets the API key for the app, which can be found in the Google Developers Console. Some APIs require this key to be set in order to work.

Arguments:

apiKey - **type:** **string**

The API key to set.

Classes

OAuth 2.0 Token Object

The OAuth 2.0 token object represents the OAuth 2.0 token and any associated data. The properties of this object include:

access_token - **type:** **string**

The OAuth 2.0 token. Only present in successful responses.

`error` - **type: string**
Details about the error. Only present in error responses.

`expires_in` - **type: string**
The duration, in seconds, the token is valid for. Only present in successful responses.

`state` - **type: string**
The Google API scopes related to this token.

gapi.client.HttpRequest

An object encapsulating an HTTP request. This object is not instantiated directly, rather it is returned by `gapi.client.request`. It defines one method:

gapi.client.HttpRequest.execute(callback)

Executes the request and runs the supplied callback on response.

Arguments:

`callback(jsonResp, rawResp)` - **type: Function**
The callback function which will execute when the request succeeds or fails. `jsonResp` contains the response parsed as JSON. If the response is not JSON, this field will be `false`. `rawResp` is the HTTP response. It is JSON, and can be parsed to an object which includes `body`, `headers`, `status`, and `statusText` fields.

The full [Google JavaScript Client Library reference](#) contains additional methods and classes.

Plugin methods

The following methods are used to control the rendering and loading of the Google+ plugins, including the [+1 button](#), [badges](#), [follow button](#), [interactive share](#), [share button](#), and [sign-in button](#).

gapi.community.* (Community badge)

gapi.community.go(opt_container)

Renders all community badge tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container
The container containing the community badge tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all community badge tags on the page are rendered.

gapi.community.render(container, parameters)

Renders the specified container as a community badge.

Arguments:

container
The container to render as the community badge. Specify either the ID of the container (string) or the DOM element itself.

parameters
An object containing `tag attributes` as key=value pairs, for example, `{ "width": "300", "theme": "light" }`.

The following table lists the parameters that are applicable to the `gapi.community.render()` method:

Attribute	Value	Default	Description
<code>href</code>	URL to the Google+ page or user profile		The URL of the Google+ community that the user might choose to visit and join.
<code>layout</code>	<code>landscape</code> <code>portrait</code>	<code>portrait</code>	Sets the orientation of the badge
<code>showphoto</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the community profile photo if set to true and the photo exists.

<code>showowners</code>	<code>true</code> <code>false</code>	<code>false</code>	Displays a list of community owners if set to true.
<code>showtagline</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the community's tag line if set to true.
<code>theme</code>	<code>light</code> <code>dark</code>	<code>light</code>	The color theme of the badge. Use <code>dark</code> when placing the badge on a page with a dark background.
<code>width</code>	<code>int</code>	300	<p>The pixel width of the badge to render.</p> <p>The following ranges are valid:</p> <p>Portrait layout 180-450 pixels</p> <p>Landscape layout 273-450 pixels</p>

`gapi.follow.*` (Follow button)

`gapi.follow.go(opt_container)`

Renders all follow button tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container

The container containing the follow button tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all follow button tags on the page are rendered.

`gapi.follow.render(container, parameters)`

Renders the specified container as a follow button.

Arguments:

container

The container to render as the follow button. Specify either the ID of the container (string) or the DOM element itself.

parameters

An object containing `tag attributes` as key=value pairs, for example, `{"width": "300", "theme": "light"}`.

The following table lists the parameters that are applicable to the `gapi.follow.render()` method:

Attribute	Value	Default	Description
<code>href</code>	URL to the Google+ page or user profile		<p>The URL of the Google+ page or user profile that the user might choose to follow. The following URL formats are supported:</p> <ul style="list-style-type: none"> <code>https://plus.google.com/110967630299632321627</code> <code>https://plus.google.com/+LarryPage</code>
<code>annotation</code>	<code>none</code> <code>bubble</code> <code>vertical-bubble</code>	<code>bubble</code>	<p>Sets the annotation to display next to the button.</p> <hr/> <p>none Do not render additional annotations.</p> <p>bubble Display the number of users who are following this page or person in a graphic next to the button.</p> <p>vertical-bubble Display the number of users who are following this page or person in a graphic above to the button.</p>

<code>height</code>	15 20 24	20	The pixel height of the button to render. This height controls only the height of the button. If you use the vertical annotation, the actual height of the full widget will be larger.
<code>rel</code>	<code>author</code> <code>publisher</code>	—	Describes the relationship of the entity defined at the href location to the page the badge is embedded.

`gapi.interactivepost.*` (Interactive posts)

`gapi.interactivepost.go(opt_container)`

Renders all interactive post tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

`opt_container`

The container containing the interactive post tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all interactive post tags on the page are rendered.

`gapi.interactivepost.render(container, parameters)`

Renders the specified container as a interactive post.

Arguments:

`container`

The container to render as the interactive post. Specify either the ID of the container (string) or the DOM element itself.

`parameters`

An object containing [tag attributes](#) as key=value pairs, for example, `{ "width": "300", "theme": "light" }`.

The following table lists the parameters that are applicable to the `gapi.interactivepost.render()` method:

Key	Value	Default	Required	Description
<code>clientid</code>	<i>string</i>	—	Required	Your OAuth 2.0 client ID that you obtained from the Google Developers Console .
<code>contenturl</code>	<i>url</i>	—	Required	<p>The content URL points to the page that you want to render in the preview snippet in the share post. This URL can differ from the call-to-action button's URL, for example, you might link to an overview with the content URL and to a form with the call-to-action button.</p> <p>If you specify the <code>contentdeeplinkid</code> attribute and the link is clicked on a mobile device, the Google+ app launches your app and passes the <code>contentdeeplinkid</code> for your app to use to determine the location within your app to take the user.</p> <p>The protocol (http/https), host name, and port (if specified) for the content URL and the call-to-action URL must match. Because these must match, you should avoid using URL shorteners and redirects for either URL.</p>
<code>contentdeeplinkid</code>	<i>string</i>	—	—	

				<p>Specify a URI path as a deep link ID for the content link in the shared post. The value must be 512 characters or fewer in length.</p> <p>To take advantage of deep linking, your mobile app should be configured to handle incoming deep links. For more information, see handling incoming deep links for Android or iOS.</p>
<code>cookiepolicy</code>	<code>uri</code> <code>single_host_origin</code> <code>none</code>	—	Required	<p>Directs the interactive share button to store user and session information in a session cookie and HTML5 session storage on the user's client for the purpose of minimizing HTTP traffic and distinguishing between multiple Google accounts a user might be signed into. We call these two means of storage the <i>client storage</i>.</p> <p>When a page containing a interactive share button is loaded, the user's sign-in state can quickly be accessed from this client storage rather than from Google servers, reducing the latency for button loading. In addition, the cookie can allow a user who is signed in to multiple Google accounts (say, work and personal) to select which account to use with your website. The cookie and session storage are deleted once the user ends their browsing session.</p> <p>The value of the <code>cookiepolicy</code> attribute determines the scope of URIs that can access the cookie. Choose this value based on the website's domain name to set the cookie's scope. Then the cookie can be accessed by all interactive share buttons within that scope. You should use a policy that is as broad as necessary for your site because that reduces the number of cookies that your site writes to the user's client. There should ideally be only one cookie for each distinct domain suffix you are allowed to write cookies for (for example, one cookie each for <code>example.com</code> and <code>example.co.uk</code>).</p> <p>Available values - Three values of <code>cookiepolicy</code> are allowed: a <code>uri</code>, <code>single_host_origin</code>, and <code>none</code>. The most versatile setting you can pass is the <code>uri</code> value that most broadly matches the structure of your website.</p> <p>For help with choosing a value, see Determining a value for cookie policy.</p> <ul style="list-style-type: none"> • <code>uri</code> <p>The <code>uri</code> includes the <i>scheme</i>, <i>site-domain</i> and possibly <i>port</i> to be set for your site. The value you provide determines the scope of the cookie. The more general the <code>uri</code>, the broader the scope of the cookie. The first five use cases are examples. Specify <code>uri</code> in one of the following three URI formats:</p>

- **http://site_domain** - An http scheme in *uri* provides the broadest scope—it sets a cookie whose scope includes both http and https domains and all of the site's subdomains. A more specific URI, such as `http://mail.example.com`, would set a more narrowly scoped cookie that further restricts its scope to only that domain and its subdomains.
- **https://site_domain** - An https scheme in *uri* is similar to but narrower than http—use it for SSL-only sites. This ensures that the cookies will have the "secure" attribute set and hence will not be sent over an un-encrypted connection.
- **scheme://site_domain:port** - Using a value that includes a port number. This value restricts the cookie's scope to only that one URI.

Note: Sites with distinct domain suffixes (such as `http://example.com` and `http://example.co.uk`) must have different *site_domain* values, and therefore must be specified with different `cookiepolicy` markup.

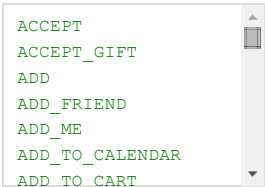
- `single_host_origin`

Use this value if your website has only a single host name and no subdomains (for example, host `http://example.com` but not `http://www.example.com`). This string is shorthand for the scheme and host name (and port number if it's not the default for the scheme) of the current page, which then is equivalent to the *uri* value above.

- `none`

The `none` value does not set cookies or session storage for the interactive share button and uses a less efficient fallback mechanism for determining user and session information. Setting this value to `none` also prevents `gapi.auth.signout` from working for the user and requires you to implement signout appropriately. This value also can prevent a user who is signed in to multiple Google accounts (say, work and personal) from being able to select which account to use with your website.

Note: You can identify client storage

				<p>entries written by the interactive share button by checking their prefixes: <code>GCS</code> prefix is used for session cookies and HTML5 session storage, and <code>G_AUTHUSER_</code> prefix is used for session cookies only.</p>
<code>calltoactionlabel</code>		Open	—	<p>The label that renders in the stream post that prompts the reader to act. This field should be a verb that accurately describe the action that will be taken.</p> <p>The <code>calltoactionurl</code> must also be specified to render the call-to-action button.</p> <p>The call-to-action label is automatically translated in the Google+ stream to the viewer's language.</p>
<code>calltoactionurl</code>	<i>url</i>	—	Required	<p>The URL that the user will be taken to when the call-to-action button is clicked from a web browser. If the button is clicked on a mobile device, the Google+ app launches your app and passes the <code>calltoactiondeeplinkid</code> for your app to use to determine the location within your app to take the user.</p> <p>The protocol (<code>http/https</code>), host name, and port (if specified) for the call-to-action URL and the <code>contenturl</code> must match. Because these must match, you should avoid using URL shorteners and redirects for either URL.</p> <p>Important: Your app or site must validate all input and be careful about performing actions when a deep link is clicked.</p>
<code>calltoactiondeeplinkid</code>	<i>string</i>	—	—	<p>Specify a URI path as a deep link ID for the call to action button. The value must be 512 characters or fewer in length.</p> <p>To take advantage of deep linking, your mobile app should be configured to handle incoming deep links. For more information, see handling incoming deep links for Android or iOS.</p>
<code>prefilltext</code>	<i>string</i>	—	—	<p>Text that is prefilled in the comment area of the share box on behalf of the user. The user can edit or remove the prefilled text.</p> <p>The maximum length of the prefilled text is 1042 characters.</p>
<code>recipients</code>	<i>string</i>	—	—	<p>A comma-separated list of user IDs and email addresses that you want to prefill the share settings with. You can get these</p>

				IDs by calling the people.list method on behalf of an authenticated user . A maximum of ten recipients are allowed.
Inherited attributes from sign in				
NOTE: These attributes apply only to web, not to mobile web.				
<code>accesstype</code>	<code>online</code> <code>offline</code>	<code>online</code>	—	If using the server-side sign-in flow and you want to get a refresh token that you can use to make offline requests, specify this attribute and set the value to <code>offline</code> .
<code>approvalprompt</code>	<code>auto</code> <code>force</code>	<code>auto</code>	—	Allows control over when the user is re-prompted for consent. When set to <code>auto</code> , the user only sees the OAuth consent dialog if they have not authorized your application. When set to <code>force</code> , the user is shown the OAuth consent dialog each time that they click the sign-in button.
<code>callback</code>	<code>function(authResult)</code>	—	—	<p>A function, in the global namespace, which is called after a successful sign-in occurs. This function must hide the sign-in button.</p> <p>This function is passed a single parameter: a JSON object with the following structure:</p> <pre>{ "id_token": A JSON web token (JWT) that contains identity information about the user that is digitally signed by Google, "access_token": the access token, "expires_in": the validity of the tokens, in seconds, "code" : a one-time code that can be sent to your server and exchanged for an access token, "error": The OAuth2 error type if problems occurred, "error_description": an error message if problems occurred }</pre>
<code>requestvisibleactions</code>	A space delimited list of moment type URIs .	—	—	If your app will write moments, list the full URI of the types of moments that you intend to write. For example: <code>http://schema.org/AddAction</code> .
<code>scope</code>	A space-delimited list of scope URIs	<code>https://www.googleapis.com/auth/plus.login</code>	—	The OAuth 2.0 scopes for the APIs that you would like to use as a space-delimited list. You can list Google+ scopes and other Google OAuth 2.0 scopes that your application might require. Find more scopes in the OAuth 2.0 Playground .

`gapi.page.*` (Page badge)

`gapi.page.go(opt_container)`

Renders all badge tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container
The container containing the badge tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all badge tags on the page are rendered.

gapi.page.render(container, parameters)

Renders the specified container as a badge.

Arguments:

container
The container to render as the badge. Specify either the ID of the container (string) or the DOM element itself.

parameters
An object containing `tag attributes` as key=value pairs, for example, `{ "width": "300", "theme": "light" }`.

The following table lists the parameters that are applicable to the `gapi.page.render()` method:

Attribute	Value	Default	Description
<code>href</code>	URL to the Google+ page		The URL of the Google+ page that is associated with this brand.
<code>layout</code>	<code>landscape</code> <code>portrait</code>	<code>portrait</code>	Sets the orientation of the badge
<code>theme</code>	<code>light</code> <code>dark</code>	<code>light</code>	The color theme of the badge. Use <code>dark</code> when placing the badge on a page with a dark background.
<code>showcoverphoto</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the cover photo in the badge if set to true and the photo exists.
<code>showtagline</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the page's tag line if set to true.
<code>width</code>	<code>int</code>	300	The pixel width of the badge to render. The following ranges are valid: Portrait layout 180-450 pixels Landscape layout 273-450 pixels

gapi.person.* (Profile badge)

gapi.person.go(opt_container)

Renders all badge tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container
The container containing the badge tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all badge tags on the page are rendered.

gapi.person.render(container, parameters)

Renders the specified container as a badge.

Arguments:

container

The container to render as the badge. Specify either the ID of the container (string) or the DOM element itself.

parameters

An object containing [tag attributes](#) as key=value pairs, for example, `{"width": "300", "theme": "light"}`.

The following table lists the parameters that are applicable to the `gapi.person.render()` method:

Attribute	Value	Default	Description
<code>href</code>	URL to the Google+ page		The URL of the Google+ profile.
<code>layout</code>	<code>landscape</code> <code>portrait</code>	<code>portrait</code>	Sets the orientation of the badge
<code>showcoverphoto</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the cover photo in the badge if set to true and the photo exists.
<code>showtagline</code>	<code>true</code> <code>false</code>	<code>true</code>	Displays the user's tag line if set to true.
<code>theme</code>	<code>light</code> <code>dark</code>	<code>light</code>	The color theme of the badge. Use <code>dark</code> when placing the badge on a page with a dark background.
<code>width</code>	<code>int</code>	300	The pixel width of the badge to render. The following ranges are valid: Portrait layout 180-450 pixels Landscape layout 273-450 pixels

gapi.plusone.* (+1 button)

gapi.plusone.go(opt_container)

Renders all +1 button tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container

The container containing the +1 button tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all +1 button tags on the page are rendered.

gapi.plusone.render(container, parameters)

Renders the specified container as a +1 button.

Arguments:

container

The container to render as the +1 button. Specify either the ID of the container (string) or the DOM element itself.

parameters

An object containing [tag attributes](#) as key=value pairs, for example, `{"width": "300", "theme": "light"}`.

The following table lists the parameters that are applicable to the `gapi.plusone.render()` method:

Attribute	Value	Default	Description
<code>href</code>	<i>URL to +1</i>	<i>current page URL</i>	Sets the URL to +1. Set this attribute when you have a +1 button next to an item description for another page and want the button to +1 the referenced page and not the current page. If you set this attribute by using <code>gapi.plusone.render</code> , you should not escape the URL.
<code>size</code>	<code>small</code> <code>medium</code> <code>standard</code> <code>tall</code>	<code>standard</code>	Sets the +1 button size to render. See button sizes for more information.
<code>annotation</code>	<code>none</code> <code>bubble</code> <code>inline</code>	<code>bubble</code>	<p>Sets the annotation to display next to the button.</p> <hr/> <p>none Do not render additional annotations.</p> <p>bubble Display the number of users who have +1'd the page in a graphic next to the button.</p> <p>inline Display profile pictures of connected users who have +1'd the page and a count of users who have +1'd the page.</p>
<code>width</code>	<i>int</i>	—	<p>If <code>annotation</code> is set to <code>"inline"</code>, this parameter sets the width in pixels to use for the button and its inline annotation. If the width is omitted, a button and its inline annotation use <code>450px</code>.</p> <p>See Inline annotation widths for examples of how the annotation is displayed for various width settings.</p>
<code>align</code>	<code>left</code> <code>right</code>	<code>left</code>	Sets the horizontal alignment of the button assets within its frame.
<code>expandTo</code>	<i>comma-separated list of</i> <code>top</code> <code>right</code> <code>bottom</code> <code>left</code>	<i>empty list</i>	<p>Sets the preferred positions to display hover and confirmation bubbles, which are relative to the button. Set this parameter when your page contains certain elements, such as Flash objects, that might interfere with rendering the bubbles.</p> <p>For example, <code>top</code> will display the hover and confirmation bubbles above the button.</p> <p>If omitted, the rendering logic will guess the best position, usually defaulting to below the button by using the <code>bottom</code> value.</p>
<code>callback</code>	<i>function(jsonParam)</i>	—	<p>If specified, this function is called after the user clicks the +1 button. The callback function must be in the global namespace and accept a single parameter, which is a JSON object with the following structure:</p> <pre>{ "href": target URL, "state": "on" "off" }</pre> <p>The <code>state</code> property is set to <code>"on"</code> for a +1, and <code>"off"</code> for the removal of a +1.</p>
<code>onstartinteraction</code>	<i>function(jsonParam)</i>	—	<p>If specified, this function is called either when a hover bubble displays, which is caused by the user hovering the mouse over the +1 button, or when a confirmation bubble displays, which is caused by the user +1'ing the page. You can use this callback function to modify your page, such as pausing a video when the bubble appears.</p> <p>This function must be in the global namespace and accept a single parameter, which is a JSON object with the following structure:</p>

			<pre>{ "id": target URL, "type": hover confirm }</pre>
<code>onendinteraction</code>	<code>function(jsonParam)</code>	—	<p>If specified, this function is called when either a hover or confirmation bubble disappears. You can use this callback function to modify your page, such as resuming a video when the bubble closes.</p> <p>This function accepts a single parameter, which is identical in structure to the parameter passed to <code>onstartinteraction</code>.</p>
<code>recommendations</code>	<code>true, false</code>	<code>true</code>	To disable showing recommendations within the +1 hover bubble, set <code>recommendations</code> to <code>false</code> . This feature is currently in platform preview .
<code>count</code>	<code>true, false</code>	<code>true</code>	Deprecated: To disable the count display, use <code>annotation="none"</code> .

`gapi.plus.*` (Share button)

`gapi.plus.go(opt_container)`

Renders all share button tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

opt_container

The container containing the share button tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all share button tags on the page are rendered.

`gapi.plus.render(container, parameters)`

Renders the specified container as a share button.

Arguments:

container

The container to render as the share button. Specify either the ID of the container (string) or the DOM element itself.

parameters

An object containing [tag attributes](#) as key=value pairs, for example, `{ "width": "300", "theme": "light" }`.

The following table lists the parameters that are applicable to the `gapi.plus.render()` method::

Attribute	Value	Default	Description
<code>href</code>	<i>URL to share</i>	<i>current page URL</i>	The URL to share. If you set this attribute by using <code>gapi.plus.render</code> , you should not escape the URL.
<code>annotation</code>	<code>inline</code> <code>bubble</code> <code>vertical-bubble</code> <code>none</code>	<code>bubble</code>	<p>The annotation to display next to the button.</p> <hr/> <p>inline Display profile pictures of connected users who have shared the page and a count of users who have shared the page.</p> <p>bubble Display the number of users who have shared the page in a graphic next to the button.</p> <p>vertical-bubble Display the number of users who have shared the page in a graphic above the button.</p> <p>none Do not render any additional annotations.</p>

<code>width</code>	<code>int</code>	—	The maximum width to allocate to the entire share plugin. See Button Sizes for more information.
<code>height</code>	<code>int</code>	20	The height to assign the button. This may be 15, 20, 24 or 60. See Button Sizes for more information.
<code>align</code>	<code>left</code> <code>right</code>	<code>left</code>	Sets the alignment of the button assets within its frame.
<code>expandTo</code>	<i>comma-separated list of</i> <code>top</code> <code>right</code> <code>bottom</code> <code>left</code>	<i>empty list</i>	<p>The preferred positions in which to display hover and confirmation bubbles, relative to the button. Set this parameter when your page contains certain elements, such as Flash objects, that might interfere with rendering the bubbles.</p> <p>If omitted, the rendering logic will guess the best position, usually defaulting to below the button (bottom).</p>
<code>onstartinteraction</code>	<i>function(jsonParam)</i>	—	<p>If specified, this function is called when an interaction bubble appears (such as when the user hovers over the button or starts the sharing flow).</p> <p>This function must be in the global namespace and may accept a single parameter, which will be a JSON object with the following structure:</p> <pre>{ "id": target URL, "type": hover confirm }</pre>
<code>onendinteraction</code>	<i>function(jsonParam)</i>	—	<p>If specified, this function is called when the interaction bubble disappears. You can use this callback function to modify your page, such as resuming a video, when the bubble closes.</p> <p>This function accepts a single parameter, which is identical in structure to the parameter passed to <code>onstartinteraction</code>.</p>

`gapi.post.*` (Embedded posts)

`gapi.post.go(opt_container)`

Renders all embedded post tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

`opt_container`

The container containing the embedded post tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all embedded post tags on the page are rendered.

`gapi.post.render(container, parameters)`

Renders the specified container as a embedded post.

Arguments:

`container`

The container to render as the embedded post. Specify either the ID of the container (string) or the DOM element itself.

`parameters`

An object containing [tag attributes](#) as key=value pairs, for example, `{ 'href' : 'https://plus.google.com/109813896768294978296/posts/hdbPtrsqMXQ' }`.

The following table lists the parameters that are applicable to the `gapi.post.render()` method:

--	--	--	--

Attribute	Value	Default	Description
<code>href</code>	URL to the post		Sets the URL to the specific post that you want to embed in your page. The URL must point to a publically shared post. If you set this attribute by using <code>gapi.post.render</code> , you should not escape the URL.

`gapi.signin.*` (Sign in)

`gapi.signin.go(opt_container)`

Renders all Google+ Sign-In tags and classes in the specified container. This function should be used only if `parsetags` is set to `explicit`, which you might do for performance reasons.

Arguments:

`opt_container`

The container containing the Google+ Sign-In tags to render. Specify either the ID of the container (string) or the DOM element itself. If the `opt_container` parameter is omitted, all Google+ Sign-In tags on the page are rendered.

`gapi.signin.render(container, parameters)`

Renders the specified container as a Google+ Sign-In.

Arguments:

`container`

The container to render as the Google+ Sign-In. Specify either the ID of the container (string) or the DOM element itself.

`parameters`

An object containing [tag attributes](#) as key=value pairs, for example, `{'clientid' : 'xxxxxxx.apps.googleusercontent.com', 'cookiepolicy' : 'single_host_origin', 'callback' : 'myCallback', 'requestvisibleactions' : 'http://schema.org/AddAction http://schemas.google.com/CommentAction'}`.

The following table lists the parameters that are applicable to the `gapi.signin.render()` method:

Key	Value	Default	Required	Description
<code>clientid</code>	<i>string</i>	—	Required	Your OAuth 2.0 client ID that you obtained from the Google Developers Console .
<code>cookiepolicy</code>	<i>uri</i> <code>single_host_origin</code> <code>none</code>	—	Required	<p>Directs the sign-in button to store user and session information in a session cookie and HTML5 session storage on the user's client for the purpose of minimizing HTTP traffic and distinguishing between multiple Google accounts a user might be signed into. We call these two means of storage the <i>client storage</i>.</p> <p>When a page containing a sign-in button is loaded, the user's sign-in state can quickly be accessed from this client storage rather than from Google servers, reducing the latency for button loading. In addition, the cookie can allow a user who is signed in to multiple Google accounts (say, work and personal) to select which account to use with your website. The cookie and session storage are deleted once the user ends their browsing session.</p> <p>The value of the <code>cookiepolicy</code> attribute determines the scope of URIs that can access the cookie. Choose this value based on the website's domain name to set the cookie's scope. Then the cookie can be accessed by all sign-in buttons within</p>

that scope. You should use a policy that is as broad as necessary for your site because that reduces the number of cookies that your site writes to the user's client. There should ideally be only one cookie for each distinct domain suffix you are allowed to write cookies for (for example, one cookie each for example.com and example.co.uk).

Available values - Three values of `cookiepolicy` are allowed: a `uri`, `single_host_origin`, and `none`. The most versatile setting you can pass is the `uri` value that most broadly matches the structure of your website.

For help with choosing a value, see [Determining a value for cookie policy](#).

- `uri`

The `uri` includes the `scheme`, `site-domain` and possibly `port` to be set for your site. The value you provide determines the scope of the cookie. The more general the `uri`, the broader the scope of the cookie. The [first five use cases](#) are examples. Specify `uri` in one of the following three URI formats:

- `http://site_domain` - An http scheme in `uri` provides the broadest scope—it sets a cookie whose scope includes both http and https domains and all of the site's subdomains. A more specific URI, such as `http://mail.example.com`, would set a more narrowly scoped cookie that further restricts its scope to only that domain and its subdomains.
- `https://site_domain` - An https scheme in `uri` is similar to but narrower than http—use it for SSL-only sites. This ensures that the cookies will have the "secure" attribute set and hence will not be sent over an un-encrypted connection.
- `scheme://site_domain:port` - Using a value that includes a port number. This value restricts the cookie's scope to only that one URI.

Note: Sites with distinct domain suffixes (such as `http://example.com` and `http://example.co.uk`) must have different `site_domain` values, and therefore must be specified with different `cookiepolicy` markup.

- `single_host_origin`

				<p>Use this value if your website has only a single host name and no subdomains (for example, host <code>http://example.com</code> but not <code>http://www.example.com</code>). This string is shorthand for the scheme and host name (and port number if it's not the default for the scheme) of the current page, which then is equivalent to the <i>uri</i> value above.</p> <ul style="list-style-type: none"> <code>none</code> <p>The <code>none</code> value does not set cookies or session storage for the sign-in button and uses a less efficient fallback mechanism for determining user and session information. Setting this value to <code>none</code> also prevents <code>gapi.auth.signout</code> from working for the user and requires you to implement signout appropriately. This value also can prevent a user who is signed in to multiple Google accounts (say, work and personal) from being able to select which account to use with your website.</p> <p>Note: You can identify client storage entries written by the sign-in button by checking their prefixes: <code>GCSC</code> prefix is used for session cookies and HTML5 session storage, and <code>G_AUTHUSER_</code> prefix is used for session cookies only.</p>
<code>accesstype</code>	<code>online</code> <code>offline</code>	<code>online</code>	—	<p>If using the server-side sign-in flow and you want to get a refresh token that you can use to make offline requests, specify this attribute and set the value to <code>offline</code>.</p>
<code>apppackage</code>	<code>string</code>	—	—	<p>If you have an Android app, you can drive automatic Android downloads from your web sign-in flow. Set this parameter to the package name that you listed in your Google Developers Console project. For example:</p> <pre>com.google.android.apps.plus.</pre> <p>Automatic Android app installs are limited to free apps in the Google Play store that exceed a quality threshold.</p>
<code>approvalprompt</code>	<code>auto</code> <code>force</code>	<code>auto</code>	—	<p>Allows control over when the user is re-prompted for consent. When set to <code>auto</code>, the user only sees the OAuth consent dialog if they have not authorized your application. When set to <code>force</code>, the user is shown the OAuth consent dialog each time that they click the sign-in button.</p>

				<p>You should use this parameter when your app needs to acquire a new one-time code because you have lost or discarded your one-time code or refresh tokens. Do not use this parameter to simulate a sign-in and sign-out solution or to disable the immediate authorization check that occurs when the button is rendered or page-level configuration parameters are present. See signing out the user.</p>
<code>callback</code>	<code>function(authResult)</code>	—	—	<p>A function in the global namespace, which is called when the sign-in button is rendered and also called after a sign-in flow completes. When the button is rendered the callback occurs to check whether or not the user previously authorized the app and should be automatically signed in.</p> <p>This function should hide or remove the sign-in button when the sign in is successful as indicated by the <code>authResult</code> object containing an access token.</p> <p>This function is passed a single parameter: a JSON object that contains a number of properties. The following example shows the structure of the object and highlights a subset of the properties:</p> <pre> { "id_token": A JSON web token (JWT) that contains identity information about the user that is digitally signed by Google, "access_token": the access token, "expires_in": the validity of the token, in seconds, "error": The OAuth2 error type if problems occurred }</pre>
<code>height</code>	<code>short</code> <code>standard</code> <code>tall</code>	<code>standard</code>	—	<p>Sets the vertical height of the button.</p> <p>See also width.</p>
<code>includegrantedscopes</code>	<code>true</code> <code>false</code>	<code>true</code>	—	<p>If <code>true</code>, all previously granted scopes remain granted in each incremental request, for incremental authorization. The default value <code>true</code> is correct for most use cases; use <code>false</code> only if employing delegated auth, where you pass the bearer token to a less-trusted component with lower programmatic authority.</p>
<code>requestvisibleactions</code>	<i>A space-delimited list of moment type URIs.</i>	—	—	<p>If your app will write moments, list the full URI of the types of moments that you intend to write. For example: http://schema.org/AddAction.</p>
<code>scope</code>	<i>A space-delimited list of scope URIs</i>	https://www.googleapis.com/	—	

		<code>auth/plus.login</code>		The OAuth 2.0 scopes for the APIs that you would like to use as a space-delimited list. You can list Google+ scopes and other Google OAuth 2.0 scopes that your application might require. Find more scopes in the OAuth 2.0 Playground .
<code>theme</code>	<code>light</code> <code>dark</code>	<code>dark</code>	—	The color theme of the badge. Light renders a white button with red text and icon. Dark renders a red button with white text and icon.
<code>width</code>	<code>iconOnly</code> <code>standard</code> <code>wide</code>	<code>standard</code>	—	Controls the width of the button and the button text that appears. The approximate width is calculated for the user's language to fit the corresponding text: <ul style="list-style-type: none"> <code>iconOnly</code> - Displays the Google+ icon only. <code>standard</code> - Displays the Google+ icon followed by "Sign in" <code>wide</code> - Displays the Google+ icon followed by "Sign in with Google" See also height .

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#).

Last updated October 8, 2014.