Google Loader Developer's Guide

In order to use the Google APIs, you must import them using the Google API loader in conjunction with the API key. The loader allows you to easily import one or more APIs, and specify additional settings (such as language, location, API version, etc.) applicable to your needs.

In addition to the basic loader functionality, savvy developers can also use dynamic loading or auto-loading to enhance the performance of your application.

Table of Contents

Introduction to Loading Google APIs

Detailed Documentation

- `google.load`
- Versioning
- Dynamic Loading
- Auto-Loading

Available APIs

Introduction to Loading Google APIs

To begin using the Google APIs, first you need to sign up for an API key. The API key costs nothing, and allows us to contact you directly if we detect an issue with your site.

To load the APIs, include the following script in the header of your web page. Enter your Google API key where it says `INSERT-YOUR-KEY` in the snippet below.

Warning: You need your own API key in order to use the Google Loader. In the example below, replace "`INSERT-YOUR-KEY`" with your own key. Without your own key, these examples won't work.

```
<script type="text/javascript" src="https://www.google.com/jsapi?key=INSERT-YOUR-KEY"></script>
```

Next, load the Google API with `google.load(module, version)`, where

- `module` calls the specific API module you wish to use on your page.
- `version` is the version number of the module you wish to load. The example below loads the latest stable versior of the search API, and the specified versions of the JQuery and JQuery UI libraries.

```
<script type="text/javascript">
  google.load("search", "1");
  google.load("jquery", "1.4.2");
  google.load("jqueryui", "1.7.2");
</script>
```

After you call `google.load`, you can use all of the loaded modules in your web page. For specific examples of each API, visit the code playground or the documentation specific to the desired API(s) (see links in the left navigation).

The loader is cached in the user's browser for up to one hour.

Detailed Documentation

`google.load`

`google.load(moduleName, moduleVersion, optionalSettings)`, allows you to call individual APIs by version, where:

- `moduleName` is the name of the API (e.g., `"maps"` or `"search"`).
- `version` specifies the version of the API module, as described below. You must always specify the version of the API you are using. If you are unsure which version you want to use, use the version stated in the in the documentation for each API.
- `optionalSettings` specifies all optional configuration options for the API you are loading as a JavaScript object literal. Different APIs have different options as listed in available APIs. The possible properties are:
  - `callback`: The function to call once the script has loaded. If using the Auto-loading feature, this must specif a function name, not a function reference.
  - `language`: The language in which to localize the API's UI controls. This is specified as a ISO639 language code.
  - `nocss`: A boolean that tells the API whether to load any style sheets typically associated with its controls. If you don't intend to use the default CSS, you can reduce the load time by setting this to `true`. The default setting is `false`.
  - `packages`: An array of strings specifying related packages to be read in along with the core API. For example, you could load "piechart" and "table" along with the Visualization API.
  - `base_domain`: The base domain from which to load the API. For example, you could load from "ditu.google.cn" with the "maps" module to get the Chinese version of the Maps API.
  - `other_params`: Specific parameters supported by a particular API (and usually very specific to the API). A alternative to passing in a parameter via a `<script>` tag.

Versioning

The second parameter of `google.load` is the version of the API. Every API has a major version and revision number, of the form `VERSION.REVISION`. Every time an an API introduces new JavaScript, the revision number increases. So if an API is currently on version 2.2.3, and the team does an update, the next version becomes 2.2.4.

Our APIs are updated frequently, so to ensure stability, all of our APIs have an active *stable version* as well as a *test version*. Every time a team introduces a new API version, the previous version becomes the *stable version* of the API, and the most recent becomes the *test version*.

To always load the latest stable version of the API, request the version number without specifying a revision. So, using the above example, requesting version 2 loads the latest stable revision of the API, e.g., *2.2.3*.

To always load the test version of the API, you can use the wildcard *2.x*.

The usage model Google encourages is:

- Use the stable version of each API in the production HTML.
- Use the test version of each API (e.g., *2.x*) on your development machines, and report any issues you find in the developer forum for that API. If many users encounter serious issues with a particular API revision, Google will revert or hold back the revision.

While you can technically request any older version of an API at any time, old versions of APIs are not officially supported. In many cases, server-side changes will require that you stop using old versions of the API. However, Google tries try to keep old versions of each API for long periods of time, so you have ample time to upgrade.

Dynamic Loading

The standard `google.load` functionality loads the API(s) when your page loads; however, you can also load the AP dynamically. This is useful if you don't need the API to be available when the page loads, such as when a user performs a search or some other action.

Dynamic load is not possible for the following APIs, because they do not support callbacks:

- Friend Connect
- Earth
- gData
- Orkut

To load the API dynamically, pass a `callback` option in the third parameter, as follows:

```
function mapsLoaded() {
  var map = new google.maps.Map2(document.getElementById("map"));
  map.setCenter(new google.maps.LatLng(37.4419, -122.1419), 13);
}

function loadMaps() {
  google.load("maps", "2", {"callback" : mapsLoaded});
}
```

In this example, the Maps API loads and the `mapsLoaded` callback executes when `loadMaps()` is called. Make sure the DOM is ready when you call `google.load` with the `callback` option. You need to do this because the loader will try to append an element to the DOM. Subsequent calls to load the Maps API will immediately execute the provided callback, so you don't have to worry about loading the same API more than once.

You can load the Google API loader dynamically by creating a `script` element and setting its source to the same `"https://www.google.com/jsapi?INSERT-YOUR-KEY"` URL with an additional query `callback` parameter. The callback will be executed when the loader is ready. See the snippet below:

```
function mapsLoaded() {
  var map = new google.maps.Map2(document.getElementById("map"));
  map.setCenter(new google.maps.LatLng(37.4419, -122.1419), 13);
}

function loadMaps() {
  google.load("maps", "2", {"callback" : mapsLoaded});
}

function initLoader() {
  var script = document.createElement("script");
  script.src = "https://www.google.com/jsapi?key=INSERT-YOUR-
KEY&callback=loadMaps";
  script.type = "text/javascript";
  document.getElementsByTagName("head")[0].appendChild(script);
}
```

Auto-Loading

It is possible to auto-load a list of APIs or Javascript libraries when including the loader script. This allows you to reduc the load time in many cases by reducing the number of JavaScript requests that run at load time.

Warning! This advanced feature can be difficult to implement, depending on the exact situation. Therefore, we recommend that you consider auto-loading only in specific instances where reducing latency is crucial.

To use this feature, start with the configuration wizard.

If you don't want to use the configuration wizard, you can also autoload APIs manually. To use it manually, you need t specify the list of APIs to load in the initial `<script>` tag, rather than in a separate `google.load` call for each API For instance, the object declaration to auto-load version 1.0 of the Search API (English language), version 2.x of the Maps API, and the local search element, would look like:

```
{
  "modules" : [
    {
      "name" : "search",
      "version" : "1.0",
      "language" : "en"
    },
    {
      "name" : "maps",
      "version" : "2.x"
    },
    {
      "name" : "elements",
      "version" : "1.0",
      "packages" : [
        "localsearch"
      ]
    }
  ]
}
```

This would be compressed to:

```
{"modules":[{"name":"search","version":"1.0","language":"en"},
{"name":"maps","version":"2.x"},{"name":"elements","version":"1.0","packages":
["localsearch"]}]}
```

And then URL encoded to:

```
%7B%22modules%22%3A%5B%7B%22name%22%3A%22search%22%2C%22version%22%3A%221.0%22%2C%
22language%22%3A%22en%22%7D%2C%7B%22name%22%3A%22maps%22%2C%22version%22%3A%222.X%
22%7D%2C%7B%22name%22%3A%22elements%22%2C%22version%22%3A%221.0%22%2C%22packages%22%
3A%5B%22localsearch%22%5D%7D%5D%7D
```

This whole string is then appended as the value of the `autoload` parameter in the initial `<script>` tag:

```
<script src="https://www.google.com/jsapi?autoload=%7B%22modules%22%3A%5B%7B%22name%
22%3A%22search%22%2C%22version%22%3A%221.0%22%2C%22language%22%3A%22en%22%7D%2C%7B%
22name%22%3A%22maps%22%2C%22version%22%3A%222.X%22%7D%2C%7B%22name%22%3A%22elements%
22%2C%22version%22%3A%221.0%22%2C%22packages%22%3A%5B%22localsearch%22%5D%7D%5D%
7D&key=INSERT-YOUR-KEY"></script>
```

Auto-loading supports all of the options that can be passed in using `google.load(module, version, options)`. See above for the available options and below for information on which options are supported by each API.

Note: The callback option is supported, but the value must supply the name of a function, rather than a function reference.

To automatically generate the necessary code or see an example to get you started, see our configuration wizard.

Available APIs

The following Google APIs are available:

Google Maps API

name: maps
versions: 2, 2.x
load request: `google.load("maps", "2");`
supported options: `callback, base_domain, language, other_params`
note: Version numbers between 2.92 and the latest version (2.x) can also be specified. See the Maps API documentation and API Changelog for more details.

Google Search API (see below for links to specific searchers)

name: search
versions: 1
load request: `google.load("search", "1");`
supported options: `callback, language, nocss`
note: The following APIs use the `google.load("search", "1");` load request: Blog Search, Book Search, Image Search, Local Search, News Search, Patent Search, and Video Search.

Google Feeds API

name: feeds
versions: 1
load request: `google.load("feeds", "1");`
supported options: `callback, language, nocss`

Google Language API

name: language
versions: 1
load request: `google.load("language", "1");`
supported options: `callback, language, nocss`

Google Data APIs

name: gdata
versions: 1, 1.x
load request: `google.load("gdata", "1");`
supported options: `packages`

Google Earth API

name: earth
versions: 1

load request: `google.load("earth", "1");`
supported options: none

Google Visualization API

name: visualization
versions: 1
load request: `google.load("visualization", "1");`
supported options: `packages`

Google Friend Connect API

name: friendconnect
versions: 1
load request: `google.load("friendconnect", "1");`
supported options: none

Orkut API

name: orkut
versions: 1
load request: `google.load("orkut", "1");`
supported options: `packages`

©2010 Google - Code Home - Terms of Service - Privacy Policy - Site Directory

Google Code offered in: English - Español - 日本語 - 한국어 - Português - Русский - 中文(简体) - 中文(繁體)