


# API:Search and discovery

 This page is part of the [MediaWiki action API](#) documentation.

MediaWiki, its extensions, and its sibling projects hold tremendous potential for knowledge discovery through search. The [Discovery](#) department maintains the mechanisms, tools, and services for doing so.

Users can find information in MediaWiki by [looking it up directly](#), and in Wikidata by reading [Help:Navigating Wikidata](#).

## Contents

1	<b>MediaWiki</b>
1.1	Search modules
1.2	Query list submodules
1.3	Clients
1.3.1	Command line
1.3.2	JavaScript
2	<b>CirrusSearch</b>
2.1	Additional CirrusSearch API modules
3	<b>Wikidata</b>
4	<b>Wikidata query service</b>
5	<b>Interactive examples</b>
5.1	Wikipedia
5.2	Wikidata
5.3	Wiktionary
6	<b>External examples</b>
6.1	Wikipedia
6.2	Wikidata
6.3	Wiktionary

## MediaWiki

The [MediaWiki API](#) has several search-related modules. You can make requests and view generated help at any wiki's `/w/api.php` entry point, or fill in API request parameters at [Special:ApiSandbox](#).

### Search modules

### MediaWiki APIs

Search API pages

- [Web APIs showcase](#)
- [REST content API \(https://www.mediawiki.org/api/\)](https://www.mediawiki.org/api/)

### MediaWiki action API

- [Introduction and quick start](#)
- [FAQ](#)
  - [Etiquette and usage limits](#)
- [Formats](#)
- [Error reporting](#)
- [Restricting usage](#)
- [Cross-site requests](#)
- [Authentication](#)
  - [Login](#)
  - [Logout](#)
  - [Account creation](#)
- [Queries](#)
  - [Meta information](#)
  - [Properties](#)
  - [Lists](#)
- [Searching \(by title, content, coordinates...\)](#)
- [Parsing wikitext and expanding templates](#)
- [Purging pages' caches](#)
- [Parameter information](#)
- [Changing wiki content](#)
  - [Create and edit pages](#)
  - [Move pages](#)
  - [Merge pages](#)
  - [Rollback](#)
  - [Delete pages](#)
  - [Restore deleted revisions](#)
  - [\(Un\)protect pages](#)
  - [\(Un\)block users](#)
  - [\(Un\)watch pages](#)
  - [Mark revisions of watched pages as visited](#)
  - [Send email](#)

**action=opensearch**

See [API:Opensearch](#). Returns search results in OpenSearch format, each with text extract on Wikimedia projects. [View generated API help](#)

**action=languagesearch**

Search for language names in any script. [View generated API help](#)

**Query list submodules**

These [Query](#) submodules return a list of wiki pages matching the search criteria, and some return additional information about each page. Furthermore, you can use each as a [generator](#) to provide many other [Properties](#) of the set of returned pages, such as a lead image, snippet, and/or page description.

**action=query list=prefixsearch**

Retrieves wiki page titles with the given prefix. See the showcase article [Page info in search results](#). See module documentation for [API:Prefixsearch](#) and [View generated API help](#).

**action=query list=search**

Uses the wiki search engine to find matching pages. On Wikimedia wikis it provides search results from [CirrusSearch](#), returning typical search result information such as text snippets and page size. See module documentation for [API:Search](#) and [View generated API help](#)

**action=query list=geosearch**

If the [GeoData](#) extension is installed on the wiki, then this returns wiki pages near a location, with their geographical information. See the showcase article [Showing nearby wiki information](#), module documentation for [geosearch](#), and [View generated API help](#).

- [Patrol changes](#)
- [Import pages](#)
- [Change user group membership](#)
- [Upload files](#)
- [User options](#)
- [Tokens](#)
- [Page language](#)
- [More...](#)
- [Watchlist feed](#)
- [Wikidata](#)
- [Extensions](#)
- [Using the API in MediaWiki and extensions](#)
- [Miscellaneous](#)
- [Implementation](#)
  - [Localisation](#)
- [Client code](#)
- [Asserting](#)

[v](#) · [d](#) · [e](#) (<https://www.mediawiki.org/w/index.php?title=Template:API&action=edit>)

**Clients**

**Command line**

From the command line you can query the API using [cURL](#) to make the API request, then use [jq](#) (<http://stedolan.github.io/jq/>) to parse the JSON response.

For example, let's try looking up item [wikidata:Q39246](#) on Wikidata, and request its English-language label:

```
$ URL='https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q39246&format=json'
$ curl -s $URL | jq '.entities[].labels.en.value'
"Richard Feynman"

$ curl -s $URL | jq '.entities[].claims|length'
55
```

We find that [Q39246](#) is the Wikidata identifier for the item with English label "Richard Feynman", and that there are 55 claims made about him.

**JavaScript**

To write a MediaWiki API client in JavaScript, all that's needed is a JSONP handler. Many libraries (e.g. [jQuery](#)) include JSONP clients, or one can be written independently.

Within the MediaWiki ecosystem, jQuery can be used directly:

```
$.ajax({
  url: ' '//www.wikidata.org/w/api.php',
  data: { action: 'wbgetentities', ids: mw.config.get('wgWikibaseItemId'), format: 'json' },
  dataType: 'jsonp',
  success: function (x) {
    console.log('wb label', x.entities.Q39246.labels.en.value);
    console.log('wb description', x.entities.Q39246.descriptions.en.value);
  }
});
```

This uses jQuery's `$.ajax()` which is available in many interactive JavaScript coding environments and makes sense if your eventual goal is a separate standalone project.

If your eventual goal is code running on a wiki, e.g. as a Gadget, then you should use the higher-level `mw.api()` function provided by the 'mediawiki.api' ResourceLoader module.

In other environments, a simple JSONP handler can be written:

```
var mw;
(function (mw) {

  /**
   * Query a MediaWiki api.php instance with the given options
   */
  function mwQuery(endpoint, options) {

    /**
     * Create a uniquely-named callback that will process the JSONP results
     */
    var createCallback = function (k) {
      var i = 1;
      var callbackName;
      do {
        callbackName = 'callback' + i;
        i = i + 1;
      } while (window[callbackName])
      window[callbackName] = k;
      return callbackName;
    }

    /**
     * Flatten an object into a URL query string.
     * For example: { foo: 'bar', baz: 42 } becomes 'foo=bar&baz=42'
     */
    var queryStr = function (options) {
      var query = [];
      for (var i in options) {
        if (options.hasOwnProperty(i)) {
          query.push(encodeURIComponent(i) + '=' + encodeURIComponent(options[i]));
        }
      }
      return query.join('&');
    }

    /**
     * Build a function that can be applied to a callback. The callback processes
     * the JSON results of the API call.
     */
    return function (k) {
      options.format = 'json';
      options.callback = createCallback(k);
      var script = document.createElement('script');
      script.src = endpoint + '?' + queryStr(options);
      var head = document.getElementsByTagName('head')[0];
      head.appendChild(script);
    };
  }

});
```

```
mw.api = {
  query: mwQuery,
};

})(mw || (mw = {}));
```

# CirrusSearch

CirrusSearch is a MediaWiki extension to enable Elastic-based search of MediaWiki content. It acts as a search back-end, so `action=query&list=search` is the main interface to this.

You can use the same Cirrus features in API queries that users can enter in the search box. For example, you can use the `morelike:` special prefix to find related pages. Search for pages related to Marie Curie and radium.

**`api.php ? action=query & list=search & srsearch=morelike:Marie_Curie|radium & srlimit=10 & srprop=size & formatversion=2` ([https://en.wikipedia.org/w/api.php?action=query&list=search&srsearch=morelike:Marie\\_Curie%7Cradium&srlimit=10&srprop=size&formatversion=2](https://en.wikipedia.org/w/api.php?action=query&list=search&srsearch=morelike:Marie_Curie%7Cradium&srlimit=10&srprop=size&formatversion=2))** [try in ApiSandbox] ([https://en.wikipedia.org/wiki/Special:ApiSandbox#action=query&list=search&srsearch=morelike:Marie\\_Curie%7Cradium&srlimit=10&srprop=size&formatversion=2](https://en.wikipedia.org/wiki/Special:ApiSandbox#action=query&list=search&srsearch=morelike:Marie_Curie%7Cradium&srlimit=10&srprop=size&formatversion=2))

## Additional CirrusSearch API modules

In addition, CirrusSearch can report its configuration and internal information. These APIs are probably only useful if you're familiar with Elasticsearch and want to see how CirrusSearch uses it. These are all considered internal debugging API's and no guarantees are made with regards to backwards compatability of changes to their output.

### **`?action=cirrusdump` page parameter**

For example, <https://en.wikipedia.org/wiki/2014?action=cirrusdump>

### **`?cirrusDumpQuery` parameter to Special:Search queries**

This is an action parameter to `index.php`, for example

<https://en.wikipedia.org/wiki/Special:Search/cat%20dog%20chicken?cirrusDumpQuery>

### **`?cirrusDumpResult` parameter to Special:Search queries**

This is an action parameter to `index.php`, for example

<https://en.wikipedia.org/wiki/Special:Search/cat%20dog%20chicken?cirrusDumpResult>

An additional parameter, `cirrusExplain`, can be passed with `cirrusDumpResult` to have the lucene explanation of the the score included with the result dump. For example

<https://en.wikipedia.org/wiki/Special:Search/cat%20dog%20chicken?cirrusDumpResult&cirrusExplain>

### **API modules `cirrus-config-dump`, `cirrus-settings-dump`, `cirrus-mapping-dump`**

These dump the CirrusSearch setup. Here's a sample query

**`api.php ? action=cirrus-config-dump & formatversion=2` (<https://en.wikipedia.org/w/api.php?action=cirrus-config-dump&formatversion=2>)** [try in ApiSandbox] (<https://en.wikipedia.org/wiki/Special:ApiSandbox#action=cirrus-config-dump&formatversion=2>)

# Wikidata

Wikidata's API (<https://www.wikidata.org/w/api.php>) includes a few actions (*`wbgetentities`* (<https://www.wikidata.org/w/api.php?action=help&modules=wbgetentities>), *`wbgetclaims`* (<https://www.wikidata.org/w/api.php?action=help&modules=wbgetclaims>), *`wbsearchentities`* (<https://www.wikidata.org/w/api.php?action=help&modules=wbsearchentities>)) that can be used to search for information about entities, properties, statements, and claims.

# Wikidata query service

Wikidata query service performs graph-based searching of Wikidata via a SPARQL API. It's available at <https://query.wikidata.org/>

WDQS Explorer (demo) ([http://earldouglas.github.io/wdqs-explorer/Richard\\_Feynman/](http://earldouglas.github.io/wdqs-explorer/Richard_Feynman/)) (source code) (<https://github.com/earldouglas/wdqs-explorer>) provides in-browser graph exploration using SPARQL queries against the Wikidata query service.

# Interactive examples

## Wikipedia

Browse to [https://en.wikipedia.org/wiki/Main\\_Page](https://en.wikipedia.org/wiki/Main_Page), open up the JavaScript console, and run the following:

```
$.ajax({
  url: '//en.wikipedia.org/w/api.php',
  data: { action: 'query', list: 'search', srsearch: 'Richard Feynman', format: 'json' },
  dataType: 'jsonp',
  success: function (x) {
    console.log('title', x.query.search[0].title);
  }
});
```

This logs the string Richard Feynman to the JavaScript console.

If the MediaWiki libraries and environment are unavailable, this can be done using the `wmQuery()` function above:

```
var queryWikipedia = mw.api.query('//en.wikipedia.org/w/api.php',
  { action: 'query', list: 'search', srsearch: 'Richard Feynman' });

queryWikipedia(function (x) {
  console.log('title', x.query.search[0].title);
});
```

## Wikidata

Using JSONP, we can perform the above steps right from the Web browser's JavaScript console. On Wikipedia, the Wikidata item identifier is available via the MediaWiki configuration value `wgWikibaseItemId`.

Browse to [https://en.wikipedia.org/wiki/Richard\\_Feynman](https://en.wikipedia.org/wiki/Richard_Feynman), open up the JavaScript console, and run the following:

```
$.ajax({
  url: '//www.wikidata.org/w/api.php',
  data: { action: 'wbgetentities', ids: mw.config.get('wgWikibaseItemId'), format: 'json' },
  dataType: 'jsonp',
  success: function (x) {
    console.log('wb label', x.entities.Q39246.labels.en.value);
    console.log('wb description', x.entities.Q39246.descriptions.en.value);
  }
});
```

This logs the string Richard Feynman and the Wikidata entry description string "American quantum physicist" to the JavaScript console.

If the MediaWiki libraries and environment are unavailable, this can be done using the `wmQuery()` function above:

```
var queryWikidata = mw.api.query('//www.wikidata.org/w/api.php',
  { action: 'wbgetentities', ids: 'Q39246' });

queryWikidata(function (x) {
  console.log('wb label', x.entities.Q39246.labels.en.value);
  console.log('wb description', x.entities.Q39246.descriptions.en.value);
});
```

# Wiktionary

Browse to [https://en.wikipedia.org/wiki/Main\\_Page](https://en.wikipedia.org/wiki/Main_Page), open up the JavaScript console, and run the following:

```
$.ajax({
  url: '//en.wiktionary.org/w/api.php',
  data: { action: 'query', prop: 'revisions', rvprop: 'content', titles: 'Richard Feynman', format: 'json' },
  dataType: 'jsonp',
  success: function (x) {
    console.log('wiktionary title', x.query.pages['-1'].title);
  }
});
```

This logs the string Richard Feynman to the JavaScript console.

If the MediaWiki libraries and environment are unavailable, this can be done using the `wmQuery ( )` function above:

```
var queryWiktionary = mw.api.query('//en.wiktionary.org/w/api.php',
  { action: 'query', prop: 'revisions', rvprop: 'content', titles: 'Richard Feynman' });

queryWiktionary(function (x) {
  console.log('wiktionary title', x.query.pages['-1'].title);
});
```

# External examples

These approaches can be used to build external Web applications that can search MediaWiki instances via the API.

## Wikipedia

Wikipedia Query [demo](http://earldouglas.github.io/wikipedia-query/selection-search/) (<http://earldouglas.github.io/wikipedia-query/selection-search/>) ([source code](https://github.com/earldouglas/wikipedia-query)) (<https://github.com/earldouglas/wikipedia-query>) provides in-browser full-text searching using the `list` search.

## Wikidata

Wikidata Explorer [demo](http://earldouglas.github.io/wikidata-explorer/Richard_Feynman/) ([http://earldouglas.github.io/wikidata-explorer/Richard\\_Feynman/](http://earldouglas.github.io/wikidata-explorer/Richard_Feynman/)) ([source code](https://github.com/earldouglas/wikidata-explorer)) (<https://github.com/earldouglas/wikidata-explorer>) provides in-browser graph exploration using the `wbgetentities` action.

## Wiktionary

Wiktionary Query [demo](http://earldouglas.github.io/wiktionary-query/selection-search/) (<http://earldouglas.github.io/wiktionary-query/selection-search/>) ([source code](https://github.com/earldouglas/wiktionary-query)) (<https://github.com/earldouglas/wiktionary-query>) provides in-browser word searching using the `revisions` property.

Retrieved from "[https://www.mediawiki.org/w/index.php?title=API:Search\\_and\\_discovery&oldid=2539382](https://www.mediawiki.org/w/index.php?title=API:Search_and_discovery&oldid=2539382)"

**This page was last edited on 18 August 2017, at 05:58.**

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. See [Terms of Use](#) for details.