

---

## App Store Optimization (aso)

This Node.js library provides a set of functions to aid App Store Optimization of applications in iTunes and Google Play.

The functions use either google-play-scraper or app-store-scraper to gather data, so bear in mind a lot of requests are performed under the hood and you may hit throttling limits when making too many calls in a short period of time.

- Installation
- API reference
  - Keyword Scores
    - ★ Difficulty
    - ★ Traffic
  - Keyword suggestions
    - ★ Suggestions by category
    - ★ Suggestions by similarity
    - ★ Suggestions by competition
    - ★ Suggestions by an arbitrary list of apps
    - ★ Suggestions based on seed keywords
    - ★ Suggestions based on search hints
  - App visibility score
  - App Keywords
    - ★ A note on keyword relevancy for iTunes
  - Store backend configuration

### Installation

```
1 npm install aso
```

### API Reference

The module exports a function to build a client that will query either iTunes ('[itunes](#)') or Google Play ('[gplay](#)');

```
1 const gplay = require('aso')('gplay');
2 const itunes = require('aso')('itunes');
3
```

---

```
4 // do stuff with google play
5 gplay.scores('panda').then(console.log);
6
7 // do stuff with itunes
8 itunes.scores('panda').then(console.log);
```

The behaviour of the algorithms is the same for both stores, except where noted.

## Keyword scores

The `scores` function gathers several statistics about a keyword and builds `difficulty` and `traffic` scores that can be used to evaluate the convenience of targeting that keyword.

The only argument is the keyword itself:

```
1 const aso = require('aso')('gplay');
2
3 aso.scores('panda').then(console.log)
```

Returns:

```
1 { difficulty:
2   { titleMatches: { exact: 10, broad: 0, partial: 0, none: 0, score:
3     10 },
4     competitors: { count: 33, score: 5.95 },
5     installs: { avg: 2470000, score: 10 },
6     rating: { avg: 4.04, score: 8.08 },
7     age: { avgDaysSinceUpdated: 81.4, score: 8.53 },
8     score: 8.84 },
9   traffic:
10    { suggest: { length: 3, index: 3, score: 8.7 },
11      ranked: { count: 5, avgRank: 52.2, score: 5.48 },
12      installs: { avg: 2470000, score: 10 },
13      length: { length: 5, score: 8.5 },
14      score: 8.18 } }
```

Scores are calculated as linear functions and aggregated with somewhat arbitrary weights. All statistics are included in the response to allow custom scoring functions to be used.

Any suggestions on how to tune or improve the score calculations are welcome :)

**Difficulty** The difficulty of a keyword measures how hard it is to rank high on searches for that keyword. This is usually the most important aspect to consider when picking a keyword (after relevance of the keyword for the given app). The lower this score, the better the candidate keyword.

The properties considered for this score are:

- 
- **titleMatches**: classifies the titles of the top 10 apps for the keyword according to how well they match the words that make it: exact (contains all the words, in the same order), broad (contains all the words in a different order), partial (contains some of the words), none (does not contain any of the words).
  - **competitors**: counts how many of the top 100 apps for the keyword actually target that keyword in their title and description.
  - **installs**: measures the average amount of installs of the top 10 apps. Since iTunes does not expose the amount of installs, the reviews count is used instead.
  - **rating**: measures the average rating of the top 10 apps.
  - **age**: measures the average time since the apps in the top 10 have been updated.

**Traffic** The traffic score estimates how much traffic that keyword gets. Note this factor is better considered after picking keywords with high relevance and low difficulty. A high score means high traffic and therefore a better keyword candidate.

The properties considered for this score are:

- **suggest**: For Google Play the amount of characters needed for the keyword to come up as a suggestion in the search box, and the position in the suggestions list. iTunes already scores their suggest results, so that number is used instead.
- **ranked**: the amount of apps in the top 10 of the keyword that appear in their category rankings, and the average ranking of those that do.
- **installs**: same metric as in difficulty, but with a lower weight in the overall score.
- **length**: length of the keyword (less traffic is assumed for longer keywords).

## Keyword suggestions

The **suggest** function returns a list of suggestions consisting of the most commonly used keywords among a given set of apps. There are several strategies to select that set of apps.

This function takes an options object with the following properties: \* **strategy**: the strategy used to get suggestions. Defaults to **CATEGORY**. \* **num**: the amount of suggestions to get in the results. Defaults to 30. \* **appId**: store app ID (for iTunes both numerical and bundle IDs are supported). Required for the **CATEGORY**, **SIMILAR** and **COMPETITION** strategies. \* **apps**: array of store app IDs. Required for the **ARBITRARY** strategy. \* **keywords**: array of seed keywords. Required for the **KEYWORDS** and **SEARCH** strategies.

A common flow of work would be to try all the strategies for a given app, hand pick the most interesting keywords and then run the **scores** function on them to analyze their quality.

---

**Suggestions by category** Looks at apps in the same category as the one given.

```
1 const aso = require('aso')('gplay');
2
3 aso.suggest({
4   strategy: aso.CATEGORY,
5   appId: 'com.dxco.pandavszombies',
6   num: 5})
7 .then(console.log);
```

Returns:

```
1 [ 'game', 'world', 'features', 'weapons', 'action' ]
```

**Suggestions by similarity** Looks at apps marked by Google Play as “similar”. For iTunes the “customers also bought” apps are used instead (which may not necessarily be similar to the given app).

```
1 const aso = require('aso')('gplay');
2
3 aso.suggest({
4   strategy: aso.SIMILAR,
5   appId: 'com.dxco.pandavszombies',
6   num: 5})
7 .then(console.log);
```

Returns:

```
1 [ 'game', 'zombies', 'zombie', 'weapons', 'action' ]
```

**Suggestions by competition** Looks at apps that target the same keywords as the one given.

```
1 const aso = require('aso')('gplay');
2
3 aso.suggest({
4   strategy: aso.COMPETITION,
5   appId: 'com.dxco.pandavszombies',
6   num: 5})
7 .then(console.log);
```

Returns:

```
1 [ 'game', 'zombies', 'features', 'app', 'zombie' ]
```

---

#### **Suggestions by an arbitrary list of apps**

```
1 const aso = require('aso')('gplay');
2
3 aso.suggest({
```

---

```
4 strategy: aso.ARBITRARY,  
5 apps: ['com.dxco.pandavszombies'],  
6 num: 5})  
7 .then(console.log);
```

Returns:

```
1 [ 'game', 'zombies', 'features', 'app', 'zombie' ]
```

**Suggestions based on seed keywords** Look at apps that target one of the given seed keywords.

```
1 const aso = require('aso')('gplay');  
2  
3 aso.suggest({  
4   strategy: aso.KEYWORDS,  
5   keywords: ['panda', 'zombies', 'hordes'],  
6   num: 5})  
7 .then(console.log);
```

Returns:

```
1 [ 'features', 'game', 'zombies', 'panda', 'zombie' ]
```

**Suggestions based on search hints** Given a set of seed keywords, infer a new set from the search completion suggestions of each one. Then look at apps that target the resulting keywords. This is expected to work better for iTunes, where the search completion yields more results.

```
1 const aso = require('aso')('gplay');  
2  
3 aso.suggest({  
4   strategy: aso.SEARCH,  
5   keywords: ['panda', 'zombies', 'hordes'],  
6   num: 5})  
7 .then(console.log);
```

Returns:

```
1 [ 'game', 'features', 'zombie', 'zombies', 'way' ]
```

## App visibility score

The `visibility` function gives an estimation of the app's discoverability within the store. The scores are built aggregating how well the app ranks for its target keywords, the traffic score for those keywords and how the app ranks in the top global and category rankings.

---

The only argument to the function is the App ID (package id for Google Play and either numerical or bundle ID for iTunes).

Google Play example:

```
1 const aso = require('aso')('gplay');
2
3 aso.visibility('com.dxco.pandavszombies').then(console.log);
```

Returns:

```
1 { keywords:
2   { 'panda vs zombies': { traffic: 2.94, rank: 1, score: 29.4 },
3     rocky: { traffic: 7.81, rank: 74, score: 57.48 },
4     'panda vs zombie': { traffic: 3.49, rank: 8, score: 34.03 },
5     'panda warrior': { traffic: 1.47, rank: 5, score: 14.49 },
6     'zombie elvis': { traffic: 3.3, rank: 1, score: 33 },
7     meatloaf: { traffic: 5.79, rank: 16, score: 54.77 },
8     ftw: { traffic: 2.88, rank: 58, score: 22.87 } },
9   collections:
10    { global: { rank: undefined, score: 0 },
11      category: { rank: undefined, score: 0 } },
12   score: 246.04 }
```

iTunes example:

```
1 const aso = require('aso')('gplay');
2
3 aso.visibility(284882215) // ID for the facebook app
4   .then(console.log);
```

Returns:

```
1 { keywords:
2   { facebook: { traffic: 9.55, rank: 1, score: 95.5 },
3     friends: { traffic: 7.21, rank: 2, score: 71.74 } },
4   collections:
5     { global: { rank: 3, score: 991 },
6       category: { rank: 2, score: 99.5 } },
7   score: 1257.74 }
```

## App keywords

The `app` function returns an array of keywords extracted from title and description of the app. The only argument is the Google Play ID of the application (the `?id=` parameter on the url).

```
1 const aso = require('aso')('gplay');
2
```

---

```
3 aso.app('com.dxco.pandavszombies').then(console.log)
```

Returns:

```
1 [
2   'panda',
3   'rocky',
4   'zombie',
5   'panda vs zombie',
6   'elvis',
7   'undead',
8   'time',
9   'game',
10  'vs',
11  (...),
12 ]
```

retext-keywords is used to extract the keywords from the app title and description.

**A note on keyword relevancy for iTunes** As said, the algorithm used by the `app` function extracts the keywords from title and description. This algorithm is also used internally by the `scores` and `suggest` functions.

While in all cases the most important place to look at for keywords is the title, the app description is usually less relevant in the iTunes app store, since there's a specific keywords list field when submitting the app. Unfortunately the contents of that field are not (that I know of) reachable from any public page or API. So keywords based on description may not have a big weight on iTunes searches.

Google Play, on the other hand, doesn't have a keywords field and so the description is expected to contain most of the app's targeted keywords.

### Store backend configuration

An object can be passed as a second argument to the client builder function, with options to override the behavior of google-play-scraper and app-store-scraper. The given options will be included in every method call to the stores. This can be used, for example, to target a different country than the default `'us'`:

```
1 const itunesRussia = require('aso')('itunes', { country: 'ru' });
2
3 // do stuff with itunes
4 itunesRussia.scores('panda').then(console.log);
```

Other options that may be useful are `cache` and `throttle`. See the reference of each scraper for all the available options.

---

### **Note about Google Play performance**

While iTunes provides an API to search apps with all their details, getting data from Google Play usually requires making a request for the search and then additional requests to get the details for each resulting app, then parsing the HTML. This means that most of the functions of this module (specially scores) will be much slower for Google Play than for iTunes (taking even minutes). This is expected given that data is scraped from Google Play in real time on every call. This can be partially mitigated using memoization, at the expense of memory usage, but a better approach (outside the scope of this project) to get faster results would be to periodically scan Google Play, save the data to a database and query that for score calculations.