APIs

An API (**A**pplication **P**rogramming **I**nterface), at its simplest, refers to the set of functions/methods, instructions and tools you use to communicate with an application or program. In other words, an API is simply a way for you to communicate with an application. In web development, your own API is the set of functions and methods that are used to communicate with your own website. External APIs are APIs that *you* can use to communicate with an external web service or web application.

Through the use of APIs, you only need to know which functions you should use without having access to or understanding the underlying code. This is important because, if you think about someone using your web application, you don't want that user modifying or having access to your code. The only thing the end user needs to know is that he or she can use this function to do something or get some data in return. This gives you access to a much wider set of functionality that you don't have to code yourself. For example, if you want Google Maps functionality on your site, simply connect to Google's Maps API and use whichever functions you need. Typically, there would also be some form of authentication via an API key or some other method in order for you to make your API request, though this is not always the case.

General steps

There are no hard and fast rules for using an API because how you connect is dependent on which web application or service you want to use, but there are a few general things you need to know or look for:

- ***Developer resources and accounts***: Most sites which have APIs have some sort of Developer page or site with API information. Once you're on the Developer page or sub-site, you should be able to find an option to create a developer account. In some cases, if you already have a non-developer account, you can also log in to the developer site with your regular log in and it will create a developer account for you. (For example, Paypal allows you to log in to the developer account with your regular account and will automatically create a developer account upon login.)
- ***Sandbox accounts***: Some applications allow you to have sandbox accounts. If they are available for free, this is great because Sandbox accounts allow you to mimic actual site processes without sending or processing data for real. This is essential for testing any sort of payment API. For example, you can test a Paypal API integration without completing actual financial transactions. It will mimic the processses without actual money being exchanged.
- ***Documentation***: Every API you use should have documentation which details how you can connect to and use the API. Sometimes this may be a little harder to find, but look around for a Developer page or API page, then look for a link for Documentation. It is important that you get used to looking for documentation and reading it because being able to use documentation is an extremely important skill for you to have as a web developer.
- ***API key***: Usually you will need an API key to use for authentication in order to allow you to use the API. This can be found somewhere in your developer account. Look for something called API or API key within your developer account. In the case of REST APIs using OAuth, you'll need a client ID and secret.

Note that although the examples in this document are for the Trakt API, the general steps of checking the documentation cannot be stressed enough because **every API is different** so you can't really escape documentation. Don't let yourself get intimidated by documentation. If you can read documentation, this is a huge advantage.

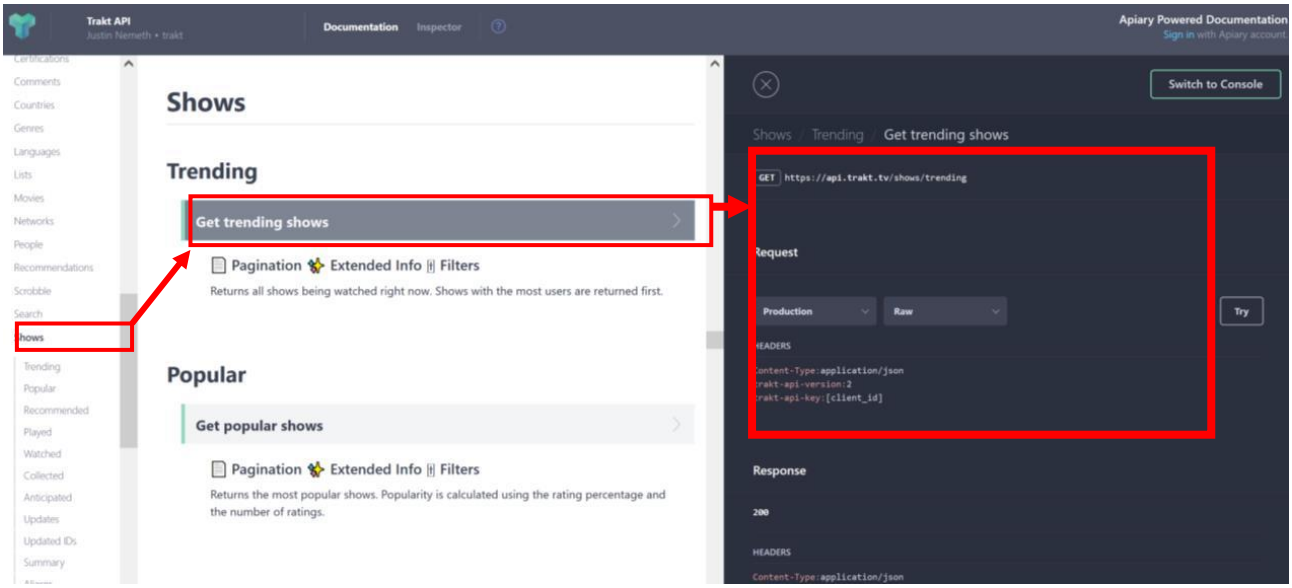
REST API documentation

Typical REST API documentation lists the required:

- ***request endpoint***: The endpoint is the URL you'll need to submit the request. Sometimes you'll only be given a URL stub (e.g. "/movies/list"). The full request URL is the API URL (find in documentation) plus the endpoint stub.
- ***request method***: The method (GET, POST, etc) is specified for the request.
- ***headers***: If the request requires specific headers, it will be specified. If the documentation only displays a request example using cURL, lines with `-H` or `--header` refer to header lines.
- ***body***: If the request requires data to be sent in the body, it will be specified. If the request in the documentation is displayed using cURL, the body data is referred to by `-d`.

Example of documentation for a request: The request below is to retrieve a list of trending shows from the Trakt API. It should be a GET request to `https://api.trakt.tv/shows/trending`. The request only requires some headers as specified in the box below.

The **Response** part below the request displays an example of how the response data will look like.



For Node.js, you can use Fetch (native to JS) or use Axios (`npm install --save axios`). The nice thing about Fetch is you don't need to install anything extra.

Trakt (*https://trakt.tv*)

Trakt is a service for tracking TV shows and movies. The API uses OAuth 2.0 for authorization for requests which are requesting specific user data. Note that not all requests need OAuth. But first, you'll need to create a Trakt account.

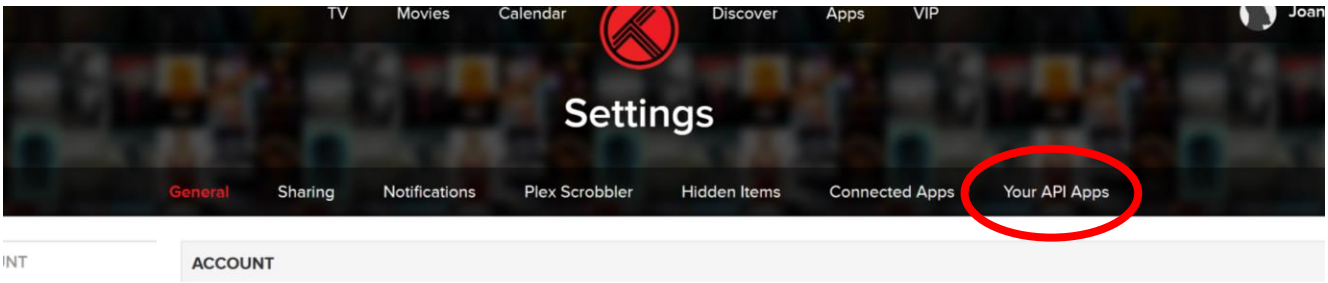
The following example is a very basic example showing how to get set up. It is not complete, as it needs to be cleaned up and more error-checking and validation need to be implemented. **To follow best practices, you should write more modular code**, but for example purposes it's easier to see everything in one file.

Before getting into the code, the first step is to create or register your API app. When creating an "app" which uses OAuth (in this case, OAuth is only required for user-based requests because this requires user authorization), you need to register the app with the service in order to receive a client ID and secret. In the case of Trakt, there are some requests which do *not* need OAuth, but you still need to register your app in order to receive a client ID (this is like your basic API key).

To do this, while logged in, go to your profile icon in the top-right corner of the page and click on **Settings**. To view the API documentation, go to the footer menu and click on **API** (https://trakt.docs.apiary.io/).

Register your site app with Trakt

1. Log in to Trakt.
2. Go to the user profile icon in the top-right corner of the page and click on **Settings** from the drop-down.
3. Within the settings page, click on **Your API Apps.**



4. Click on "**+ NEW APPLICATION**" and fill in the fields (at least the app name, redirect URI, and CORS origins).
5. Click **Save App**.
6. After your app is registered, you should see your client ID and secret required for OAuth authentication.

A basic example without OAuth

Many APIs now use JSON to build request data and to receive response data because JSON is a well-supported plain-text format that's easy to use.

You can use any library you want for making HTTP requests. A popular one is Axios, but now that Node.js supports Fetch (a JS-native way of making HTTP requests), we'll test using Fetch.

Note: Assume that the client ID, secret, and redirect URI are stored as environment variables (TRAKT_CLIENT_ID, TRAKT_CLIENT_SECRET, and TRAKT_REDIRECT_URI respectively).

To use Fetch for a request:

```
let response = await fetch(<req_url>, <options>);

//retrieve data as JSON with: response.json()
```

The options JSON object allows you to specify:

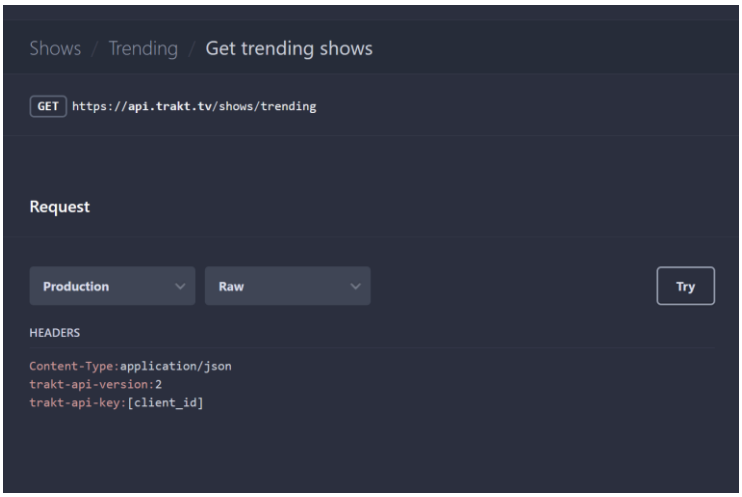
- `method: <get> | <post>`
- `headers: {`
  `"<header1>": "<header1_value>",`
  `"<header2>": "<header2_value>",`
  `…`
  `}`
- `body: <post_data_goes_here>` (data format depends on "Content-Type" header)

For more of the options you can add, see: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

You can also use a `catch()` block to catch possible errors during the request.

```
let dataToReturn;
await fetch(<req_url>, <options>)
  .then((response) => response.json())
  .then((data) => {
    //do stuff with data
    dataToReturn = data;
  }
  .catch((error) => {
    //print error message
  });
return dataToReturn;
```

We'll try a Trakt API request to get a list of trending movies (screenshot of full documentation page on page 2). Here's a close up of the sample request in the documentation.



Based on the documentation, this is a GET request and we'll need to send headers named `Content-Type`, `trakt-api-version` along with the `trakt-api-key`, where the value for the `trakt-api-key` is your client ID.
Let's write a function for this API request. But first, let's create a module file for this so that the overall site structure is neater.

Add a folder to your project for Trakt and in that folder, add a file named api.js (in red):

- <base_project_folder>
  - ***/modules/***
    - ***/trakt/***
      - *api.js*
  - ***/node_modules/***
  - ***/public/***
  - *...*
  - *index.js*

Add the following code within ***/modules/trakt/api.js***. Notice the const for the base Trakt API URL. I like to have a variable for this so that, if the URL changes in the future, you can just change base URL in one spot.

```
const trakt = "https://api.trakt.tv";

async function getTrendingMovies() {
  let reqUrl = `${trakt}/movies/trending`;
  console.log(reqUrl);
  var response = await fetch(
    reqUrl,
    {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
        "trakt-api-version": 2,
        "trakt-api-key": process.env.TRAKT_CLIENT_ID
      }
    }
  );
  return await response.json();
}

module.exports = {
  getTrendingMovies
};
```

In *index.js* add the import for the *api.js* file, making sure you place this after the lines to load environment variables (if using the dotenv module).

```
const dotenv = require("dotenv");
dotenv.config();

const trakt = require("./modules/trakt/api");
```

Finally, add the code to retrieve a list of trending movies for the home page route.

```
app.get("/", async (request, response) => {
  //the "trakt" below corresponds to the "trakt" from the require line
  let moviesList = await trakt.getTrendingMovies();

  response.render("index", { title: "Movies", movies: moviesList });
});
```