**Newsworthy**

[UPDATE: As of July, 2018, Twitter has added an approval process to use their API. You can still complete the project, but you will first have to [apply for a Twitter developer account](https://apps.twitter.com/), and the process can take several days to weeks to complete. We're working on a solution!]

Using fetch, async, and await, you'll request information from [NewsAPI.org](https://newsapi.org/) and [the Twitter API](https://dev.twitter.com/) to create a news website that will allow you to retrieve news from three different sources and post any of the articles you wish on your personal twitter!

Before you begin, you'll need to register for developer accounts for both of the APIs above. They're both free. Once you make an account, News API.org will give you an API key. You'll need to save that in **main.js**. The Twitter API will give you several different credentials which you will need to save in **Twitter.js**. You can view a live version of this project [here](https://s3.amazonaws.com/codecademy-content/courses/intermediate-javascript-requests/newsworthy/index.html).

**Note:** In this project, we'll include a call to cors-anywhere.herokuapp.com. The Twitter API requires requests to come from another backend with CORS (Cross-Origin Resource Sharing) headers set, but we can't do that since we're sending requests on the client-side from our browser. To fix this, we add https://cors-anywhere.herokuapp.com/ to the front of our request URL. This will send our request to the cors-anywhere app, which exists purely to take the request, add the necessary CORS headers, and then send it to the requested service.

Mark the tasks as complete by checking them off

**Add information from NewsAPI.org**

**1.**

Go to [News API](https://newsapi.org/register) to register for an account. As soon as you make your account, you will be provided with an API key.

**2.**

Copy and paste the API key you received from News API to const apiKey in **main.js**.

**3.**

In the live version of the project, we used Engadget, Recode, and The Next Web as the sources for our news. At [this link](https://newsapi.org/sources), you can see a list of all the news sources News API provides. Click on the three sources above to get their URLs.

In **main.js**, there are three consts: engadgetUrl, recodeUrl, and nextWebUrl. Copy and paste the URL for each of these sources to these consts.

**Retrieve data from NewsAPI.org**

**4.**

Create an asynchronous function called getNews. This function should take a single parameter, url.

Stuck? Get a hint

**5.**

Use fetch() to make a request to the url passed to the getNews() function.

If you were logged into NewsAPI.org when you retrieved the URLs to call, your API Key should already be appended to them. If not, append your apiKey to the url in the request.

Await the response of this request and save it to a response variable that you create using the keyword let.

Stuck? Get a hint

**6.**

Create a new variable called jsonResponse using the keyword let. await the response of calling .json()on the response variable you created in the previous step and save that to jsonResponse.

Stuck? Get a hint

**7.**

Console.log jsonResponse.

**8.**

At the bottom of **main.js**, there are three event listeners attached to the variables that access the three buttons at the top of the web page.

Inside each of those event handlers, call getNews() and pass it the URL for that news source.

Stuck? Get a hint

**9.**

Open your browser's JavaScript console and click any of the news source buttons in the Codecademy Browser — "Engadget," "Recode," or "The Next Web."

What is logged to the console?

**10.**

In the object logged to the console, there is an array of articles. Below the line where you created the jsonResponse variable inside of getNews(), create a new variable called articlesArray and save the first five articles inside of it.

Stuck? Get a hint

**11.**

Log articlesArray to the console inside of getNews(). Run the code, click any of the news sources, and see what is logged to the console.

Explore the structure of that object because you'll be working with it quite a bit in the next section.

**12.**

getNews() should return articlesArray. You can also delete all of the console.log() statements at this time, but they're useful to have so that you can explore the objects at any time.

**Render data from NewsAPI.org**

**13.**

In **main.js**, there is a function called renderNews(). This function iterates over articlesArray to create the HTML that will be appended to **index.html** so that the information we receive from NewsAPI.org can be displayed to our users.

In the middle of the third line of the articleRow variable inside of the renderNews() function, replace 'article title' with the property name from articlesArray that contains this information.

Stuck? Get a hint

**14.**

In the following line of articleRow, replace 'article author' with the property name from articlesArray that contains this information.

Stuck? Get a hint

**15.**

In the following line of articleRow, replace 'article description' with the property name from articlesArray that contains this information.

**16.**

In the following line of articleRow, replace 'article url' with the property name from articlesArray that contains this information.

**17.**

A few lines below, replace 'article image url' with the property name from articlesArray that contains this information.

**18.**

Below the line that adds the image to article row, we turn the "Tweet This" button into a link that will open your Twitter page in a new tab after you tweet something. If you want this functionality, replace <your user name>with your Twitter username.

If you do not want this functionality, delete the entire <a> tag around the <button> tag.

**Note:** The new page may open before the Tweet request is complete. Refresh the page if you don't see your Tweet showing up immediately!

**19.**

At the bottom of **main.js**, in each of the event listeners, chain a .then() method to the end of getNews().

.then() should take a single parameter, articlesArray, and call renderNews() — passing articlesArray to renderNews()as an argument.

Remember to do this in each of the three event listeners.

Stuck? Get a hint

**Add information from Twitter API**

**20.**

If you have not already done so, make an account on Twitter. Then, go to [Apps.Twitter](https://apps.twitter.com/) and click "Create New App."

**21.**

Give your app a name and description. In the URL field, if you have a personal website you can use that URL. Otherwise, just use http://www.codecademy.com/.

Click create.

**22.**

Open **Twitter.js**. In const oauthConsumerKey, paste the Consumer Key (API Key) given to you when you made your app in the previous step.

**23.**

Click "manage keys and access tokens" next to where you located the API Key in the previous step. Copy your "Consumer Secret" into const oauthConsumerSecret in **Twitter.js**.

**24.**

At the bottom of the page where you located your Consumer Secret, there is a button that says "Generate my access token." Click that button.

Copy your Access Token and paste it into oauthToken in **Twitter.js**.

Copy your Access Token Secret and paste it into oauthTokenSecret in **Twitter.js**.

**Retrieve data from Twitter API**

**25.**

Below the consts from the previous step, create an asynchronous function that takes a single paramenter, status, called Twitter.postStatus. It should look like this:

Twitter.postStatus = async (status) => { };

status is the Tweet that you'll want to post.

**26.**

Inside of the function you just made, we're going to use fetch() to make a POST request to Twitter.

Call fetch(), pass it this url:

`https://cors-anywhere.herokuapp.com/${baseUrl}?status=${encodeData(status)}`

This URL does not require quotations marks in addition to the backticks. Copy and paste it directly as it appears above.

The template literal above contains a few important elements:

* cors-anywhere.herokuapp.com, an application that adds the necessary CORS headers to our request (you can read more in the introduction to this project)
* the baseUrl variable, which is the API endpoint for posting tweets
* the status parameter with a value of encodeData(status)
* encodeData() is a helper function defined lower in **Twitter.js**

**27.**

In the same call to fetch() from the previous step, pass fetch() an empty settings object. await the response of calling fetch() and save it a responsevariable that you create using let.

Stuck? Get a hint

**28.**

In the settings object you passed to fetch(), add the method property with the value 'POST'.

**29.**

Add a headers property. The value of this property will be an object that has a single property, Authorization, whose value is

Twitter.generateAuthorizationHeader(status)

Stuck? Get a hint

**30.**

Below the settings object, create the jsonResponse variable using let and save await response.json() to it.

**31.**

Log jsonResponse to the console and also return jsonResponse.

**Send Tweets**

**32.**

In **main.js**, there is a sendTweets()function. This function:

* retrieves the "Tweet This" buttons from the HTML and saves them in an array-like object
* loops through the array-like object to add event listeners to each of the tweet buttons
* changes "Tweet This" to say "Tweeted"

What it doesn't do yet is actually post the Tweet.

Replace the comment in the middle of sendTweets() with a call to Twitter.postStatus(). We also have to pass an argument to .postStatus().

The argument passed to sendTweets(), newsObjects, contains the articles returned by the renderNews() function. We want to pass the url of the article that corresponds with the button that we're clicking. Use what you know about loops and iterator variables to try to figure this one out!

Stuck? Get a hint

**33.**

In the event listeners at the bottom of **main.js**, we want to chain a .then()method to the .then() method that renders the articles and images.

This .then() method should take a single parameter, articles, and call the sendTweets() method with articlespassed to it as an argument.

Do this in each of the three event handlers.

Stuck? Get a hint

**34.**

Congratulations! You made a working app with both GET and POST requests!