Project 11 – New York Times API

Objective: Use Ajax to load a JSON data set from a third party feed.

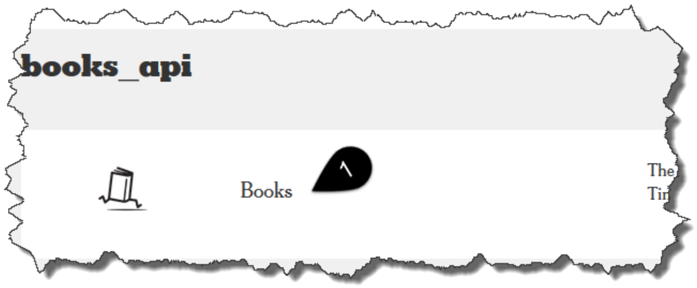
Part 1 – Follow-Along Intro

Step 1 – Request an API Key

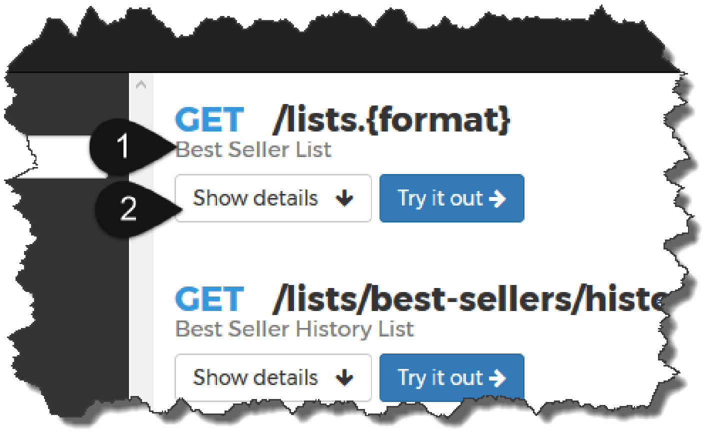
1. Go to https://developer.nytimes.com and read the Getting Started section (three short bullet points).
2. Click their link to request an API key.
    a. Fill in their form to request a key. For the *web site* field, type **localhost**. From the *API* select menu, choose **Books API**.
3. Check the email you provided to retrieve your API key. You should receive the email within one minute of submitting the form.
4. Write down or copy/paste this key as you will need it later.

Step 2 – View the Documentation

1. Return to the https://developer.nytimes.com page and click the Getting Started > FAQ link.
2. Scroll through the page and *read* each FAQ.
3. Return to the https://developer.nytimes.com page and scroll down to the list of individual APIs.
4. Click the **Books API** link.



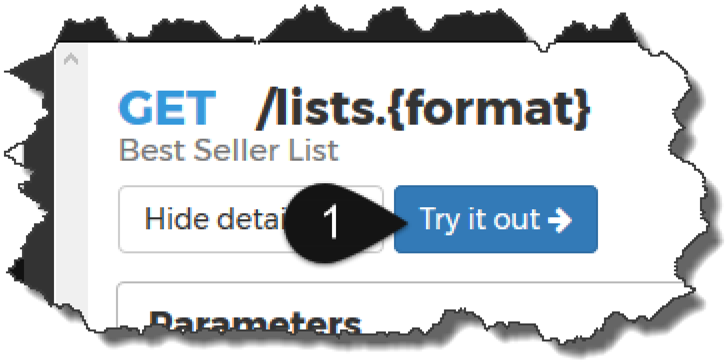
5. Notice there are six different 'things' you can search using the books API, including best sellers, names, and overview. For this tutorial, we will focus on the best sellers. Click Show Details for the Best Seller List.



6. Take note of the available parameters, such as *list*, *weeks-on-list, bestsellers-date, …* as those indicate the names of the form fields you will eventually need to create for your search form.

Step 3 – Test the API

1. Click the Try It Out button.

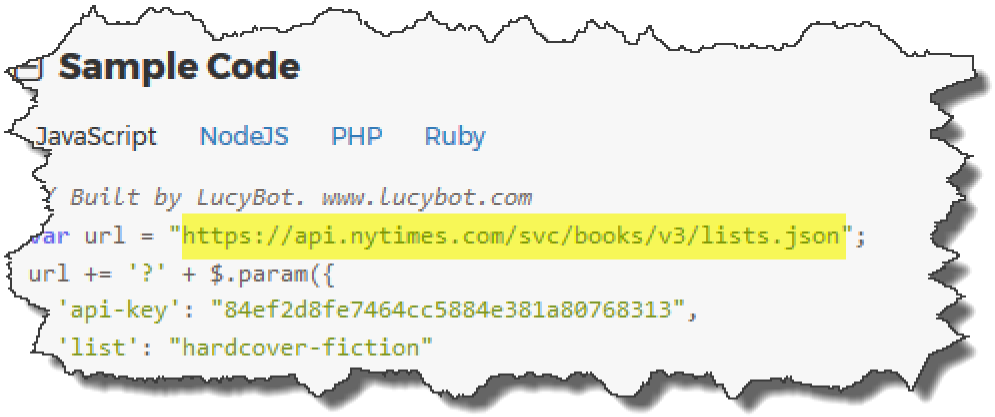


2. The left pane allows you to enter information for each parameter. The right pane shows the JSON results. Currently the right pane shows an error message because you have not provided a valid API key.

3. In the left pane, type your API key in the api-key field.
   Now you get a JSON result, but there is still an error because you have not specified any other parameters.

4. Click in the *list* field and read the message that appears under it. This describes what this parameter does. Type **hardcover-fiction** and notice the right pane now shows JSON results.

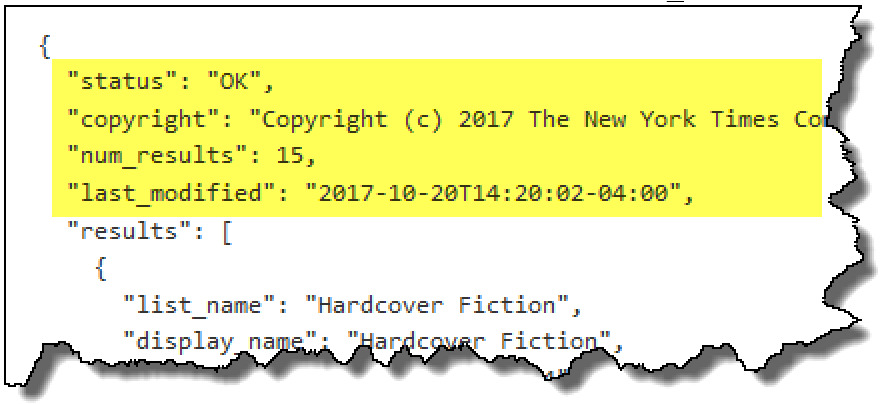
5. Lastly – look at the top of the left pane under the Sample Code heading and note the URL string. This is the URL you must use in your own code.



```
/ Built by LucyBot. www.lucybot.com
var url = "https://api.nytimes.com/svc/books/v3/lists.json";
url += '?' + $.param({
  'api-key': "84ef2d8fe7464cc5884e381a80768313",
  'list': "hardcover-fiction"
```

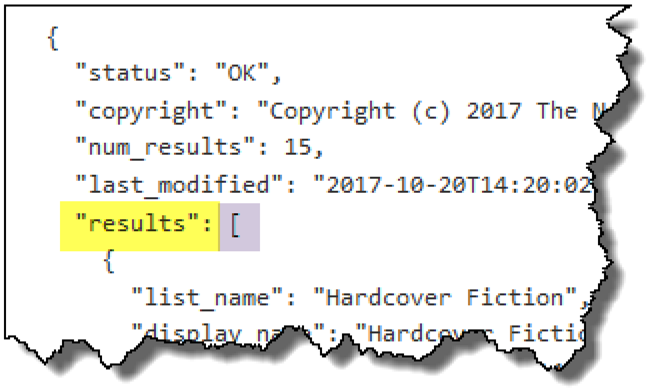
Step 4 – Examine the JSON Data

1. Look at the JSON data in the right pane. This is the file/data the server sends back to you. The first four items are similar to a variable name and its value. Notice the `num_results` line indicates 15 books have been returned.



```
{
  "status": "OK",
  "copyright": "Copyright (c) 2017 The New York Times Co
  "num_results": 15,
  "last_modified": "2017-10-20T14:20:02-04:00",
  "results": [
    {
      "list_name": "Hardcover Fiction",
      "display_name": "Hardcover Fiction",
```

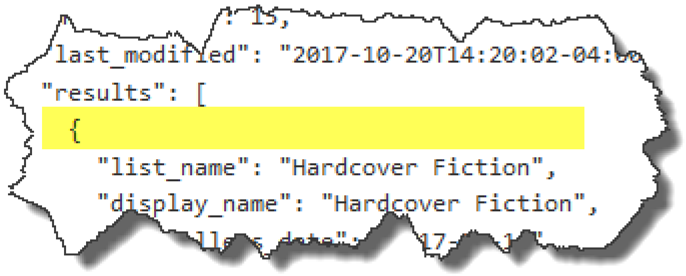
2. The next item, *results*, is more substantial. It contains an array of values as indicated by the opening square bracket.



```
{
  "status": "OK",
  "copyright": "Copyright (c) 2017 The N
  "num_results": 15,
  "last_modified": "2017-10-20T14:20:02
  "results": [
    {
      "list_name": "Hardcover Fiction",
      "display_name": "Hardcover Fictio
```

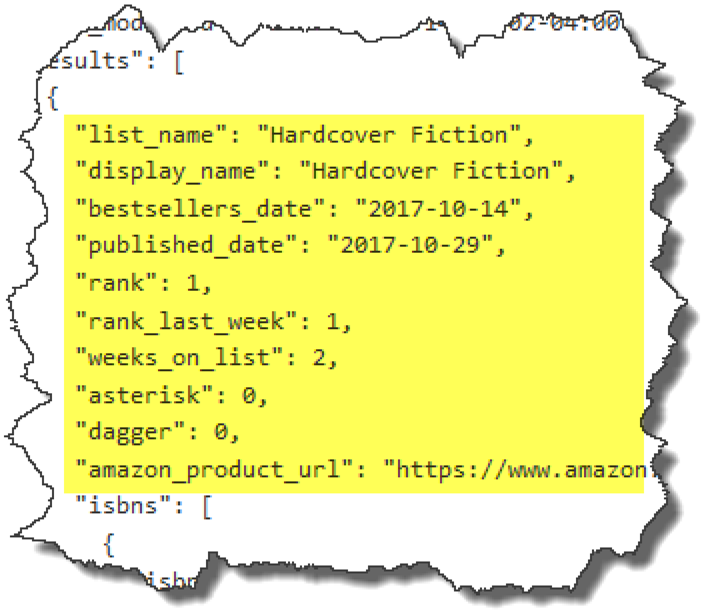
3. Let's examine this a line or chunks at a time.

4. Immediately after `"results": [`, there is an opening curly brace that starts the first book.



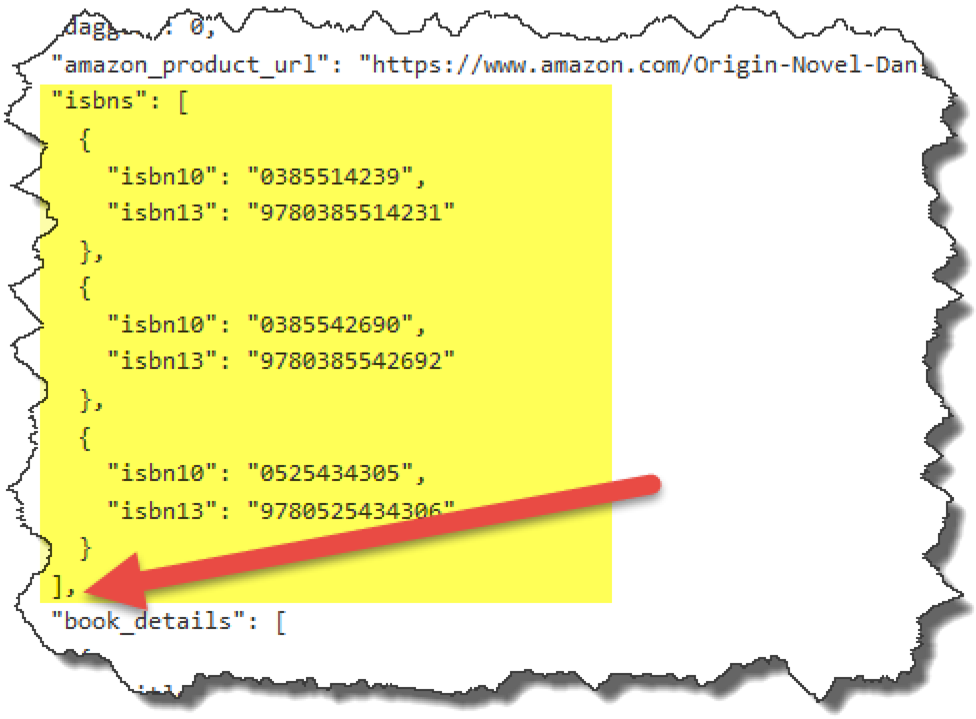
```
  ": 15,
 last_modified": "2017-10-20T14:20:02-04:00
"results": [
  {
    "list_name": "Hardcover Fiction",
    "display_name": "Hardcover Fiction",
```

5. Details about the book are indicated on the next few lines – similar to a simple variable name and its value. Notice that each one ends with a comma.



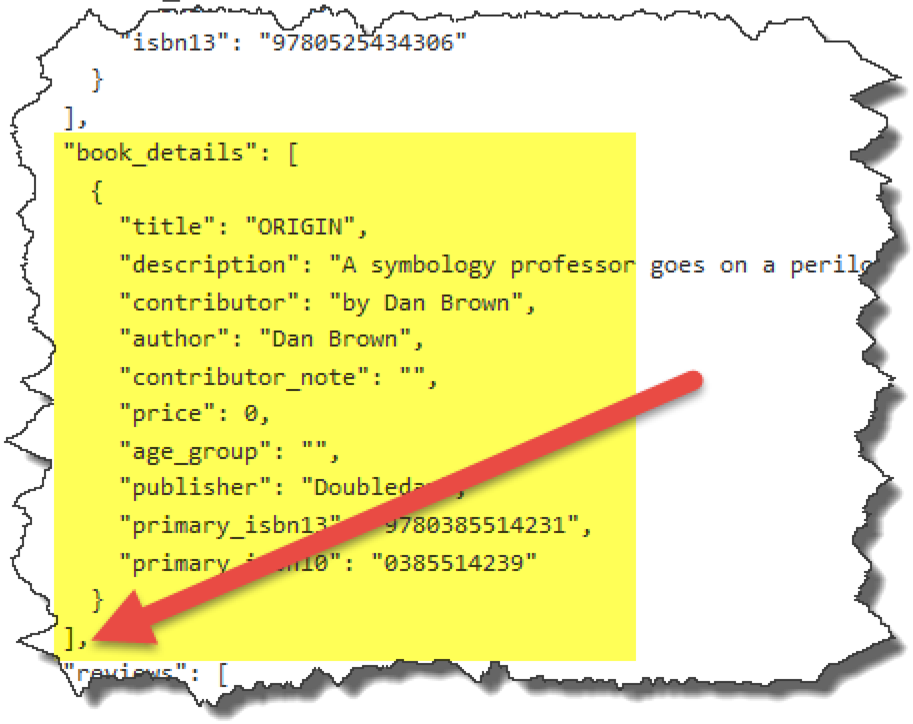
```
_mod                          1    02-04:00
esults": [
{
  "list_name": "Hardcover Fiction",
  "display_name": "Hardcover Fiction",
  "bestsellers_date": "2017-10-14",
  "published_date": "2017-10-29",
  "rank": 1,
  "rank_last_week": 1,
  "weeks_on_list": 2,
  "asterisk": 0,
  "dagger": 0,
  "amazon_product_url": "https://www.amazon
  "isbns": [
    {
      "isbn
```

6. The next item is `isbns`. Rather than storing a single value like the other items, this one stores its own nested array. Notice that when this array of values finally ends, it ends with a final comma to indicate `isbns` is finished.



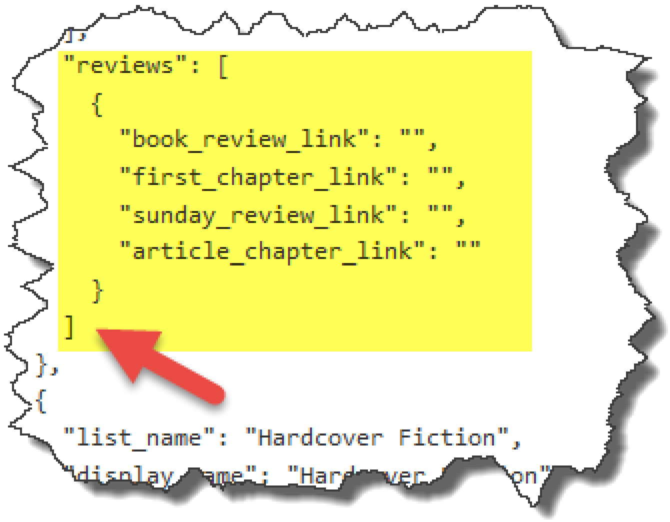
```
dagger": 0,
"amazon_product_url": "https://www.amazon.com/Origin-Novel-Dan
"isbns": [
  {
    "isbn10": "0385514239",
    "isbn13": "9780385514231"
  },
  {
    "isbn10": "0385542690",
    "isbn13": "9780385542692"
  },
  {
    "isbn10": "0525434305",
    "isbn13": "9780525434306"
  }
],
"book_details": [
```

7. Next is `book_details`. Like `isbns`, this contains an array of values. It ends with a comma.



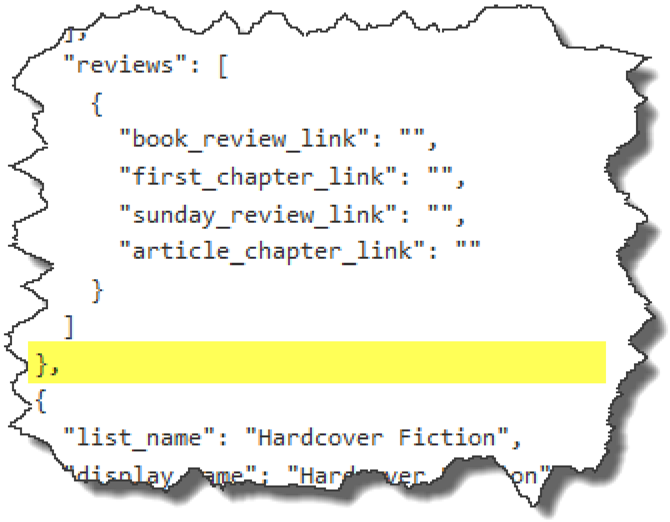
```
    "isbn13": "9780525434306"
  }
],
"book_details": [
  {
    "title": "ORIGIN",
    "description": "A symbology professor goes on a peril
    "contributor": "by Dan Brown",
    "author": "Dan Brown",
    "contributor_note": "",
    "price": 0,
    "age_group": "",
    "publisher": "Doubleday",
    "primary_isbn13": "9780385514231",
    "primary_isbn10": "0385514239"
  }
],
"reviews": [
```

8. Next is `reviews` – another nested array. As `reviews` is the last item about the first book, there is no comma at the end.



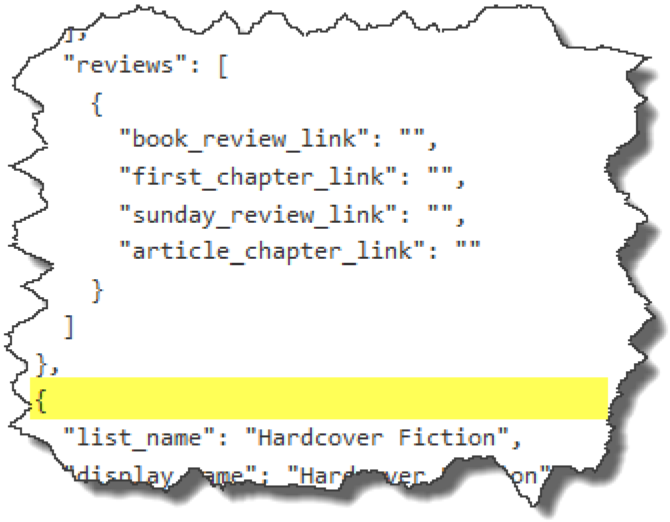
```
"reviews": [
  {
    "book_review_link": "",
    "first_chapter_link": "",
    "sunday_review_link": "",
    "article_chapter_link": ""
  }
]
},
{
"list_name": "Hardcover Fiction",
"display_name": "Hardcover Fiction"
```

9. Next is the closing curly brace and comma which marks the end of the first book.



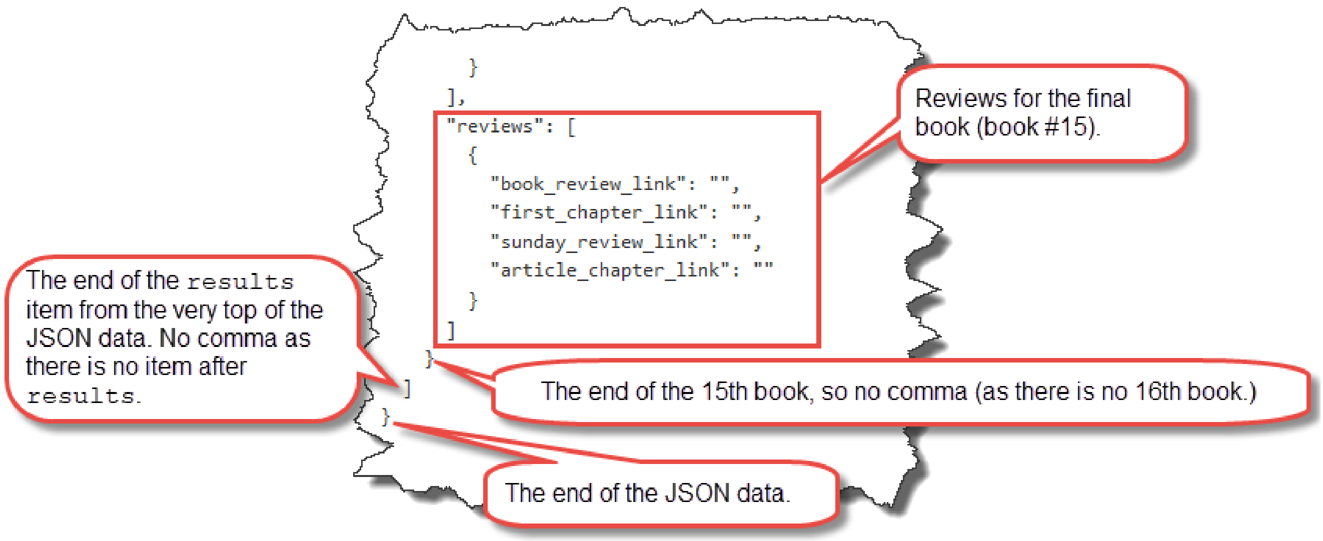
```
  ],
  "reviews": [
    {
      "book_review_link": "",
      "first_chapter_link": "",
      "sunday_review_link": "",
      "article_chapter_link": ""
    }
  ]
},
{
  "list_name": "Hardcover Fiction",
  "display_name": "Hardcover Fiction",
```

10. The next line starts a new opening curly brace which marks the start of the second book.



```
  ],
  "reviews": [
    {
      "book_review_link": "",
      "first_chapter_link": "",
      "sunday_review_link": "",
      "article_chapter_link": ""
    }
  ]
},
{
  "list_name": "Hardcover Fiction",
  "display_name": "Hardcover Fiction",
```

11. The second book contains the same types of information as the first book. Scroll down to the very bottom of the JSON data and note the purpose of all those closing braces and brackets.



```
      }
    ],
    "reviews": [
      {
        "book_review_link": "",
        "first_chapter_link": "",
        "sunday_review_link": "",
        "article_chapter_link": ""
      }
    ]
  }
 ]
}
```

Reviews for the final book (book #15).

The end of the `results` item from the very top of the JSON data. No comma as there is no item after `results`.

The end of the 15th book, so no comma (as there is no 16th book.)

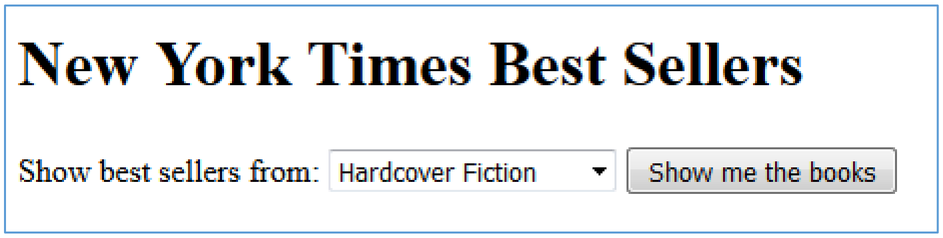
The end of the JSON data.

Part 2 – A Book Search App

Step 1 – Create the HTML Page

1. Create the following HTML form page, saving it as ***best-seller-list.htm***. Note the select field has a name of `list` because `list` is the parameter name defined in the API documentation we looked at earlier.

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>New York Times Best Sellers</title>
  <script src="js/jquery-3.2.1.min.js"></script>
  <script src="js/books.js"></script>
</head>
<body>
  <h1>New York Times Best Sellers</h1>
  <form id="bestsellers">
    Show best sellers from: <select name="list" id="list">
        <option value="hardcover-fiction">Hardcover Fiction</option>
        <option value="hardcover-nonfiction">Hardcover nonfiction</option>
    </select>
    <input type="submit" value="Show me the books">
  </form>
</body>
</html>
```



New York Times Best Sellers

Show best sellers from: Hardcover Fiction ▾ Show me the books

2. Place a copy of the jQuery file, ***jquery-3.2.1.min.js***, within a ***js*** folder (as specified in your ***script*** tag).

Step 2 – Create the Document Ready Function

1. Create a new blank JavaScript file, saving it as ***books.js***. Save it to the ***js*** folder.
2. Create a document ready function with an alert as a sanity check.

```
$('document').ready(function(){
     alert('test');
});
```

3. Save the JS file, load the HTML page in your browser, and verify the alert displays.
4. If you didn't get the alert, fix it! If you did get the alert, edit the function to call the ***sendData()*** function (which you will create next).

```
$('document').ready(function(){
     sendData();
});
```

Step 3 – Create the sendData() Function

All the code in this section should be written inside the sendData() function.

1. Outside the *ready* function, create a new function block:

```
function sendData(){

} // end function
```

2. Bind submit to the form, making sure we don't accidentally bind to some other form on the page:

```
$('#bestsellers').submit();
```

3. Call an anonymous function on submit:

```
$('#bestsellers').submit(function(){});
```

4. Pass the anonymous function the event object:

```
$('#bestsellers').submit(function(evt){});
```

5. Break apart the curly braces of the anonymous function and comment the end of it:

```
$('#bestsellers').submit(function(evt){

}); // end submit anonymous function
```

6. Prevent the default form submission from firing:

```
$('#bestsellers').submit(function(evt){
    evt.preventDefault();

}); // end submit anonymous function
```

7. *SANITY CHECK BREAK:*
   Save your work and refresh your browser. The SUBMIT button should not work (default was prevented) and you should not see the page reloading. Look at the browser tab to see if there is a flash as the title reloads. There shouldn't be.

8. Serialize the form data:

```
$('#bestsellers').submit(function(evt){
    evt.preventDefault();
    var formData = $('#bestsellers').serialize();
}); // end submit anonymous function
```

9. Encode the serialized data to accommodate for non-alphanumerical characters:

```
$('#bestsellers').submit(function(evt){
    evt.preventDefault();
    var formData = $('#bestsellers').serialize();
    formData = encodeURI(formData);
}); // end submit anonymous function
```

10. Create a variable to store your API key:

```
$('#bestsellers').submit(function(evt){
    evt.preventDefault();
    var formData = $('#bestsellers').serialize();
    formData = encodeURI(formData);
    var api = '[YOUR API KEY]';
}); // end submit anonymous function
```

NOTE: Do not actually type **[YOUR API KEY]**! Replace the brackets and text with your real API key! Keep the quotes as the API key is a literal text string.

11. Create a variable to store the base URL as per the documentation. (Note the code screenshots now show less of the code.)

```
    formData = encodeURI(formData);
    var api = '[YOUR API KEY]';
var searchURL = https://api.nytimes.com/svc/books/v3/lists.json';
}); // end submit anonymous function
```

12. Tag on the API Key. Note the parameter we must use, `api-key`, was specified in the documentation we looked at earlier.

```
    formData = encodeURI(formData);
    var api = '[YOUR API KEY]';
    var searchURL = 'https://api.nytimes.com/svc/books/v3/lists.json';
    searchURL += '?api-key=' + api;
}); // end submit anonymous function
```

13. Tag on the serialized and encoded form data:

```
    formData = encodeURI(formData);
    var api = '[YOUR API KEY]';
    var searchURL = 'https://api.nytimes.com/svc/books/v3/lists.json';
    searchURL += '?api-key=' + api;
    searchURL += '&' + formData;
}); // end submit anonymous function
```

14. Finally, send the URL string to the server and name the callback function you will use to process the server response:

```
    formData = encodeURI(formData);
    var api = '[YOUR API KEY]';
    var searchURL = 'https://api.nytimes.com/svc/books/v3/lists.json';
    searchURL += '?api-key=' + api;
    searchURL += '&' + formData;
    $.getJSON(searchURL, displayResults);
}); // end submit anonymous function
```

15. Because we are calling the displayResults() function, we need to create that function. If we don't, our code will be in error and will stop working. Add the new function after the sendData() function.

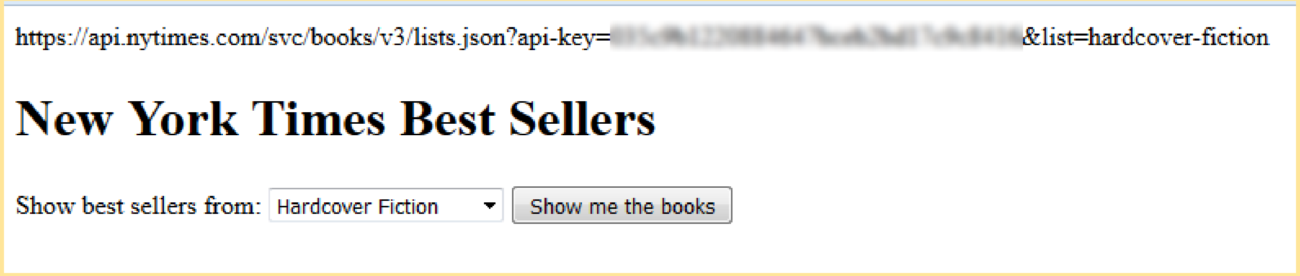
```
function displayResults(){

} // end function
```

16. *SANITY CHECK BREAK:*

    Make sure you are sending the correct URL to the server. View the contents of the searchURL variable at the top of the page by adding the URL string to the beginning of the BODY tag:

```
    $.getJSON(searchURL, displayResults);
    $('body').prepend(searchURL);
}); // end submit
```

Save your JS file and reload your page in the browser. You should see the searchURL at the top of the page.



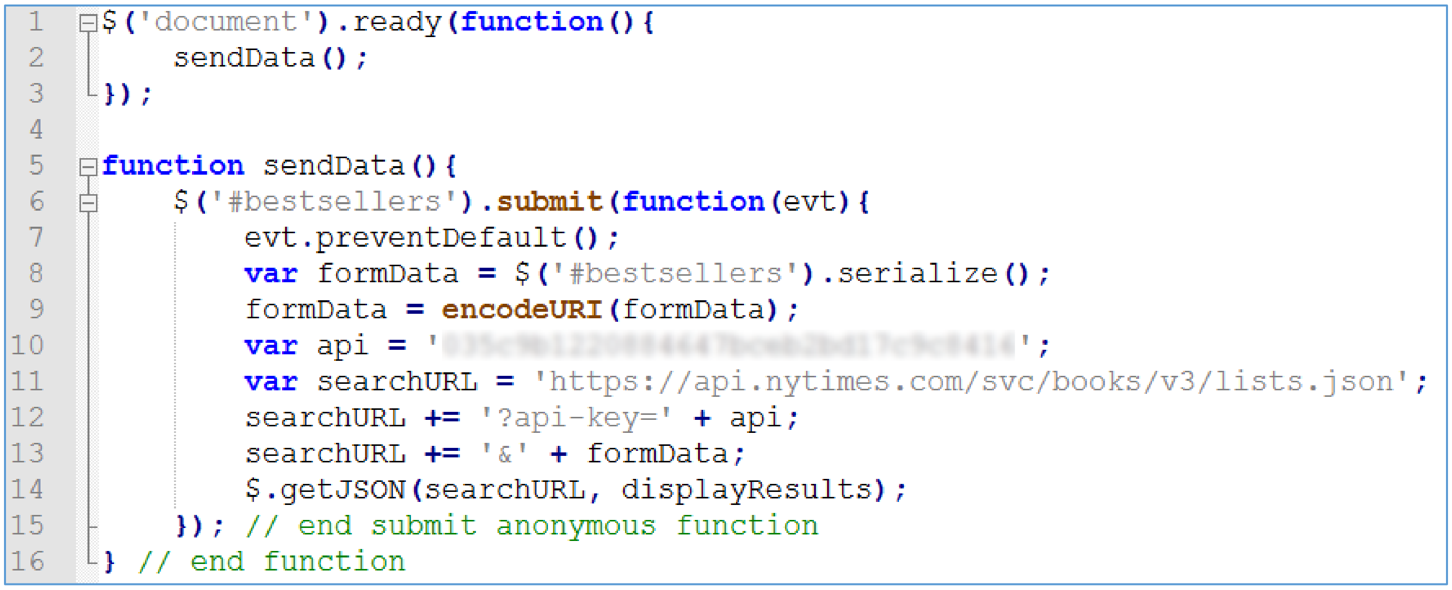
https://api.nytimes.com/svc/books/v3/lists.json?api-key=[redacted]&list=hardcover-fiction

New York Times Best Sellers

Triple-click the searchURL at the top of the page and tap [ctrl]+[c] to copy it. Open a new browser window, paste the copied URL into the address bar, and tap [enter]. If the searchURL is properly constructed, you should get a page of minified JSON data sent back from the server.

If it worked, close the JSON data tab and delete the PREPEND statement from your code. If you didn't get any JSON data returned, go back and find/fix your coding error.

17. The final code:



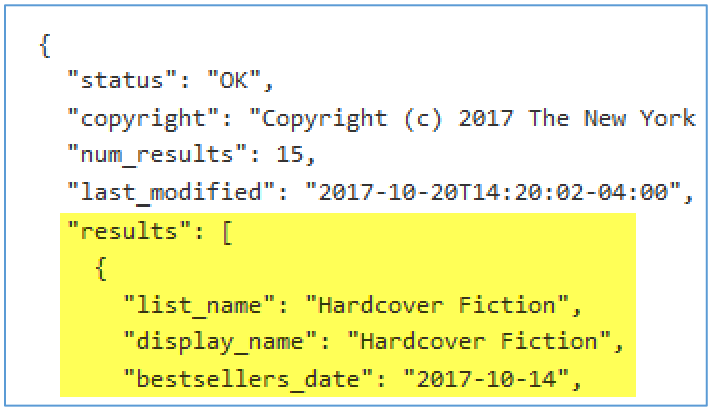
```
$('document').ready(function(){
    sendData();
});

function sendData(){
    $('#bestsellers').submit(function(evt){
        evt.preventDefault();
        var formData = $('#bestsellers').serialize();
        formData = encodeURI(formData);
        var api = '[redacted]';
        var searchURL = 'https://api.nytimes.com/svc/books/v3/lists.json';
        searchURL += '?api-key=' + api;
        searchURL += '&' + formData;
        $.getJSON(searchURL, displayResults);
    }); // end submit anonymous function
} // end function
```

Step 3 – Create the Callback Function

1. We need to give a variable name to the JSON data the server returns. We'll call it `dataFromServer`. Add that as a parameter to your callback function.

```
function displayResults(dataFromServer){

} // end function
```

2. We know the data returned from the server looks like this, and we want to access the books inside the `results` array.



```
{
  "status": "OK",
  "copyright": "Copyright (c) 2017 The New York
  "num_results": 15,
  "last_modified": "2017-10-20T14:20:02-04:00",
  "results": [
    {
      "list_name": "Hardcover Fiction",
      "display_name": "Hardcover Fiction",
      "bestsellers_date": "2017-10-14",
```

3. Create a variable to refer to the `results` array with a shorter name.

```
function displayResults(dataFromServer){
  var results = dataFromServer.results;
} // end function
```

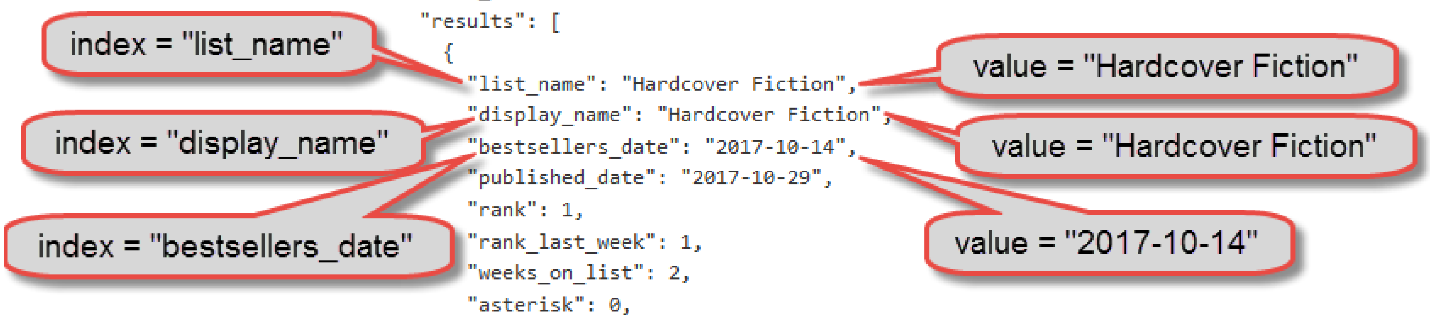
4. Loop through each member of the `results` array…

```
function displayResults(dataFromServer){
  var results = dataFromServer.results;
   $.each(results);
} // end function
```

5. …and "do something" to each one.

```
function displayResults(dataFromServer){
  var results = dataFromServer.results;
  $.each(results, function(){});
} // end function
```

6. Each member of the `results` array is identified by its index. In this case – a friendly name rather than a numerical index. Each index has a value.


index = "list_name"
index = "display_name"
index = "bestsellers_date"
value = "Hardcover Fiction"
value = "Hardcover Fiction"
value = "2017-10-14"

```
"results": [
  {
   "list_name": "Hardcover Fiction",
   "display_name": "Hardcover Fiction",
   "bestsellers_date": "2017-10-14",
   "published_date": "2017-10-29",
   "rank": 1,
   "rank_last_week": 1,
   "weeks_on_list": 2,
   "asterisk": 0,
```

7. When we loop through each member of this array, we need to extract each index and value and refer to them with variable names. While we're at it, break apart the anonymous function's curly braces and comment where they end.

```
  $.each(results, function(resultsIndex, resultsValue){
  }); // end each anonymous function
} // end function
```

8. Let's grab the name of the book list, which is the same for every book. Grab the value of the item with the index of `display_name`.

```
  $.each(results, function(resultsIndex, resultsValue){
    var bookList = resultsValue.display_name;
  }); // end each anonymous function
```

9. Create a string to display a phrase to use at the top of the page.

```
 $.each(results, function(resultsIndex, resultsValue){
   var bookList = resultsValue.display_name;
   var heading = 'Bestselling ' + bookList + ' books';
 }); // end each anonymous function
```

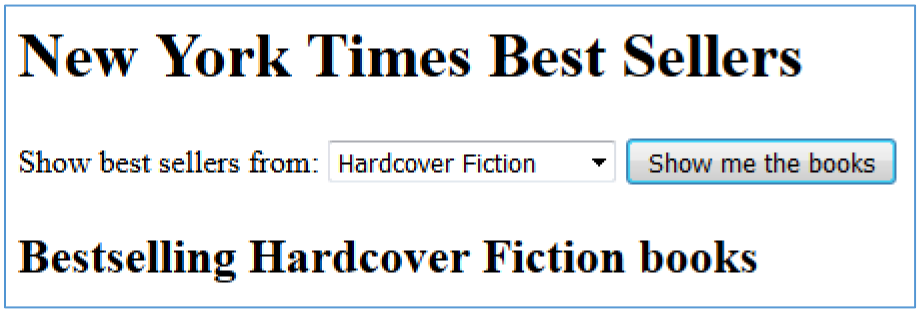
10. We need to put this text on the page. So let's back up and before we start looping through each result, let's add a new empty H2 (with an ID) after the form to title our eventual list of results. Then we can come back and fill it in with text. Add the H2 to the page as the first task of the displayResults() function so that it is in place and ready to populate with text once we start looping through each result.

```
function displayResults(dataFromServer){
  $('#bestsellers').after('<h2 id="resultsHeading"></h2>');
  var results = dataFromServer.results;
 $.each(results, function(resultsIndex, resultsValue){
```

11. Now head back to our loop and fill in the H2 with our text.

```
 $.each(results, function(resultsIndex, resultsValue){
   var bookList = resultsValue.display_name;
   var heading = 'Bestselling ' + bookList + ' books';
    $('#resultsHeading').text(heading);
 }); // end each anonymous function
```

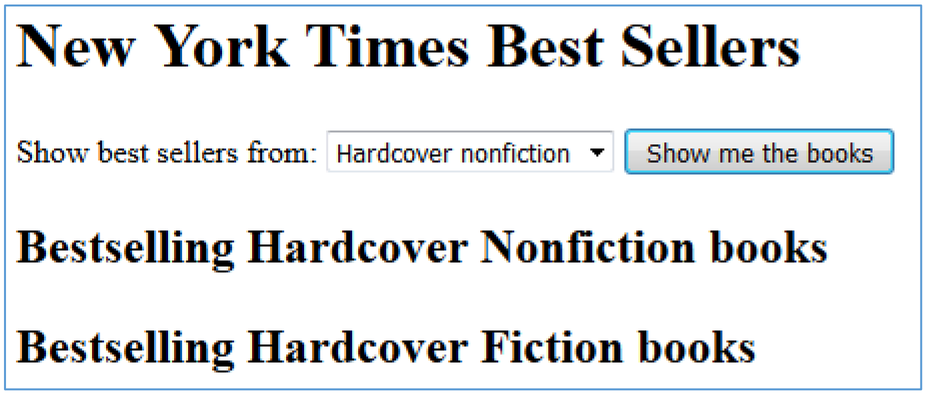
12. Save and refresh the page. Click the submit button. You should see your heading.


New York Times Best Sellers

Show best sellers from: Hardcover Fiction ▾ Show me the books

Bestselling Hardcover Fiction books

13. Change the select menu to the nonfiction option and submit it. You should the updated heading. But wait. Now we see two headings!


New York Times Best Sellers

Show best sellers from: Hardcover nonfiction

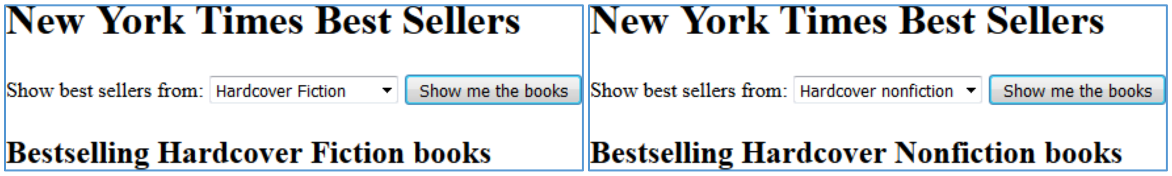
Bestselling Hardcover Nonfiction books

Bestselling Hardcover Fiction books

14. Back to the start of the function. Before we add that empty H2, we should delete the existing one from the page.

```
function displayResults(dataFromServer){
  $('#resultsHeading).remove();
  $('#bestsellers').after('<h2 id="resultsHeading"></h2>');
   var results = dataFromServer.results;
  $.each(results, function(resultsIndex, resultsValue){
```

15. Save refresh, and try both options from the select menu (fiction and nonfiction). You should get a single heading now for each.


New York Times Best Sellers

Show best sellers from: Hardcover Fiction

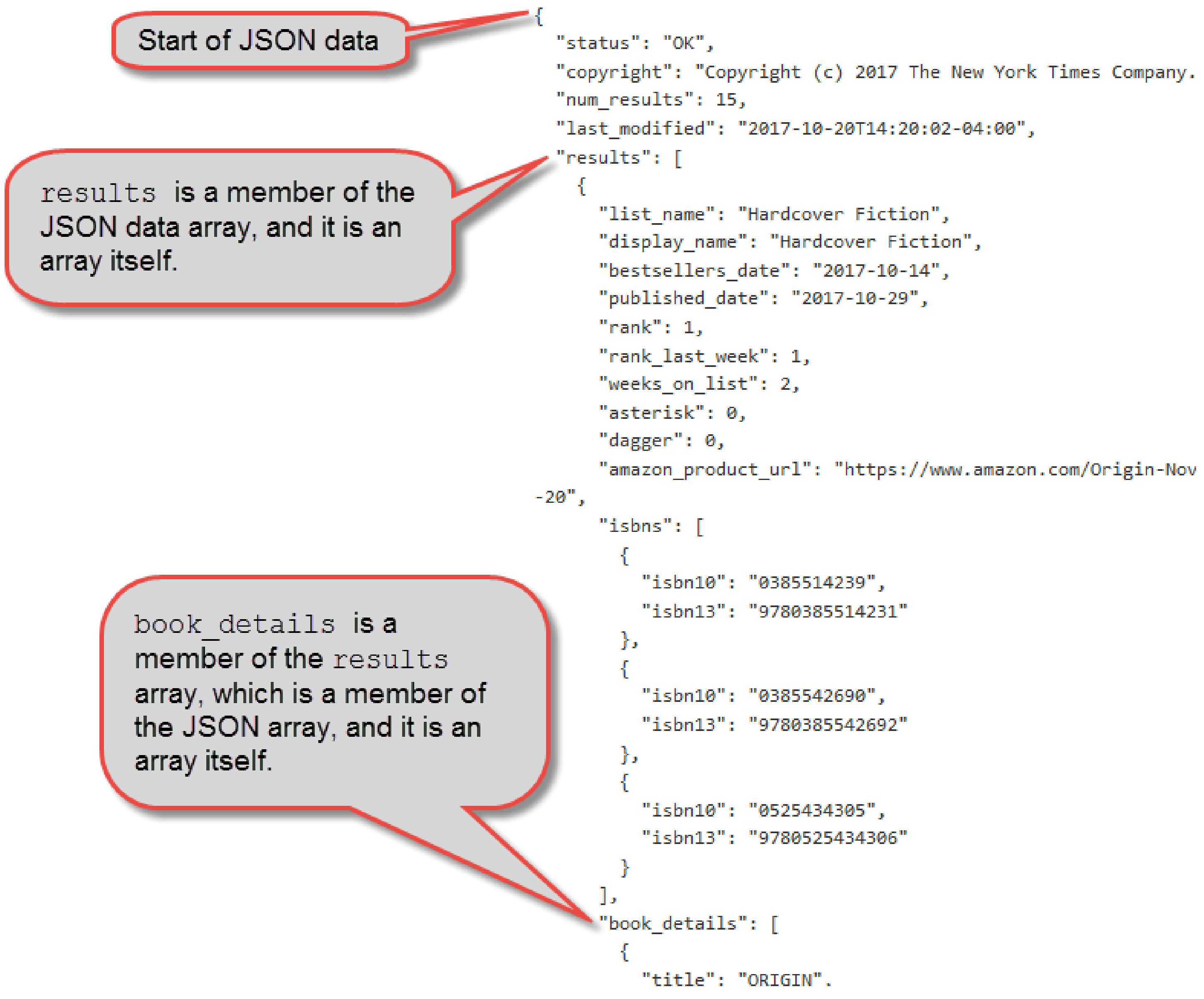
Bestselling Hardcover Fiction books

New York Times Best Sellers

Show best sellers from: Hardcover nonfiction

Bestselling Hardcover Nonfiction books

16. Now we need some details about each book such as the title, author name, and description. These things are actually stored in a `book_details` array, which is a member of `results`. It's an array within an array within an array!


Start of JSON data
results is a member of the JSON data array, and it is an array itself.
book_details is a member of the results array, which is a member of the JSON array, and it is an array itself.

```
{
  "status": "OK",
  "copyright": "Copyright (c) 2017 The New York Times Company.
  "num_results": 15,
  "last_modified": "2017-10-20T14:20:02-04:00",
  "results": [
    {
      "list_name": "Hardcover Fiction",
      "display_name": "Hardcover Fiction",
      "bestsellers_date": "2017-10-14",
      "published_date": "2017-10-29",
      "rank": 1,
      "rank_last_week": 1,
      "weeks_on_list": 2,
      "asterisk": 0,
      "dagger": 0,
      "amazon_product_url": "https://www.amazon.com/Origin-Nov
-20",
      "isbns": [
        {
          "isbn10": "0385514239",
          "isbn13": "9780385514231"
        },
        {
          "isbn10": "0385542690",
          "isbn13": "9780385542692"
        },
        {
          "isbn10": "0525434305",
          "isbn13": "9780525434306"
        }
      ],
      "book_details": [
        {
          "title": "ORIGIN",
```

17. We now need to loop through each `book_details` item to extract the title, author, and description. So this is an each loop within an each loop. Before we set up a nested `each()`, let's first grab the `book_details` item so we can refer to it using a shorter variable name.

```
$.each(results, function(resultsIndex, resultsValue){
     var bookList = resultsValue.display_name;
     var heading = 'Bestselling ' + bookList + ' books';
     $('#resultsHeading').text(heading);
     var details = resultsValue.book_details;
 }); // end each anonymous function
```

18. Now let's set up a nested `each()`.

```
$.each(results, function(resultsIndex, resultsValue){
     var bookList = resultsValue.display_name;
     var heading = 'Bestselling ' + bookList + ' books';
     $('#resultsHeading').text(heading);
     var details = resultsValue.book_details;
     $.each();
 }); // end each anonymous function
```

19. Specify that we want to loop through each of the book details…

```
$.each(results, function(resultsIndex, resultsValue){
     var bookList = resultsValue.display_name;
     var heading = 'Bestselling ' + bookList + ' books';
     $('#resultsHeading').text(heading);
     var details = resultsValue.book_details;
     $.each(details);
 }); // end each anonymous function
```

20. …and that we should run an anonymous function for each one. Note the code screenshot now shows less code.

```
     var details = resultsValue.book_details;
     $.each(details, function(){});
 }); // end each anonymous function
```

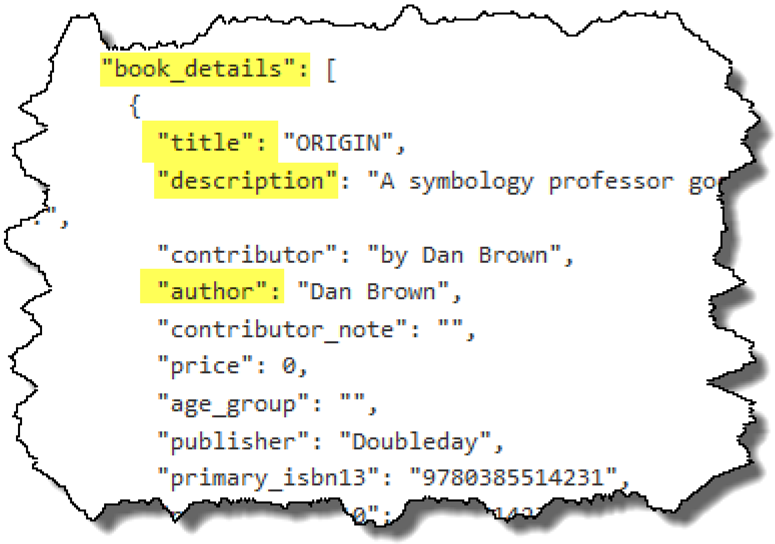
21. Break apart the curly braces and comment it.

```
     var details = resultsValue.book_details;
     $.each(details, function(){
     }); // end INNER each anonymous function
 }); // end each anonymous function
```

22. Pass arguments to extract the index and value for each.

```
     var details = resultsValue.book_details;
     $.each(details, function(detailsIndex, detailsValue){
     }); // end INNER each anonymous function
 }); // end each anonymous function
```

23. Finally, create variables to store the information we want.



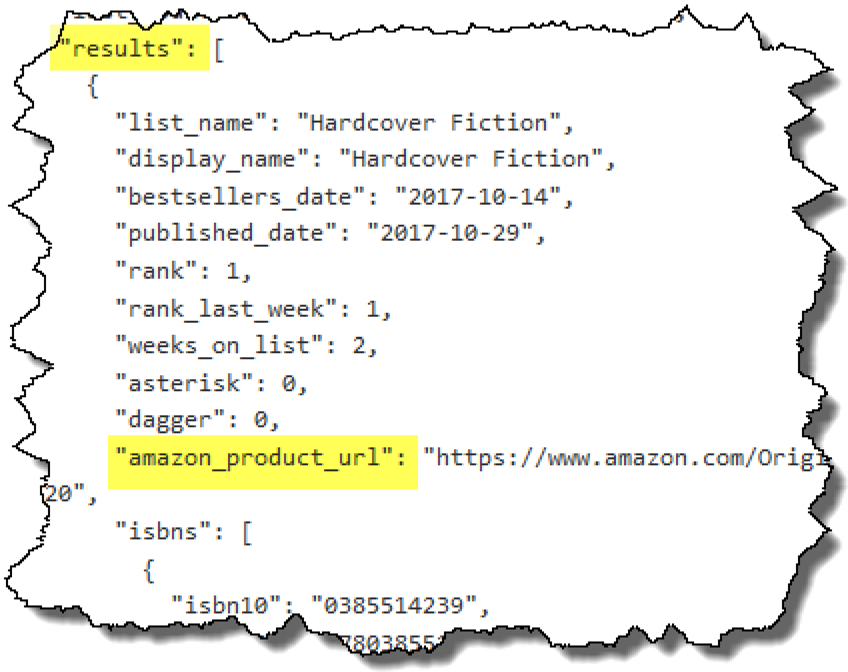
```
"book_details": [
  {
    "title": "ORIGIN",
    "description": "A symbology professor go
",

    "contributor": "by Dan Brown",
    "author": "Dan Brown",
    "contributor_note": "",
    "price": 0,
    "age_group": "",
    "publisher": "Doubleday",
    "primary_isbn13": "9780385514231",
```

```
    var details = resultsValue.book_details;
    $.each(details, function(detailsIndex, detailsValue){
      var title = detailsValue.title;
      var author = detailsValue.author;
      var description = detailsValue.description;
    }); // end INNER each anonymous function
}); // end each anonymous function
```

24. Construct a text string with this information.

```
    var details = resultsValue.book_details;
    $.each(details, function(detailsIndex, detailsValue){
       var title = detailsValue.title;
       var author = detailsValue.author;
       var description = detailsValue.description;
       var detailString = title + ' by ' + author;
       detailString += '<br>' + description;
    }); // end INNER each anonymous function
```

25. Let's also throw in the Amazon URL, which is part of the outer `results` array. We also want to make this a clickable link that opens in a new browser window. So we'll need and anchor tag with a target attribute. Note that the new code you type should be two statements – each written on a single line.



```
"results": [
  {
    "list_name": "Hardcover Fiction",
    "display_name": "Hardcover Fiction",
    "bestsellers_date": "2017-10-14",
    "published_date": "2017-10-29",
    "rank": 1,
    "rank_last_week": 1,
    "weeks_on_list": 2,
    "asterisk": 0,
    "dagger": 0,
    "amazon_product_url": "https://www.amazon.com/Origi
20",
    "isbns": [
      {
        "isbn10": "0385514239",
```

```
    var details = resultsValue.book_details;
     $.each(details, function(detailsIndex, detailsValue){
      var title = detailsValue.title;
      var author = detailsValue.author;
      var description = detailsValue.description;
      var detailString = title + ' by ' + author;
      detailString += '<br>' + description;
      detailString += '<br>' + description;
      detailString += '<br><a
href="resultsValue.amazon_product_url" target="_blank">' +
resultsValue.amazon_product_url + '</a>';
       }); // end INNER each anonymous function
```

26. We’ll display each book’s details as part of an ordered list. So let’s create an ordered list under our results heading, making sure to remove any previous list. Head back to the start of the `displayResults()` function for this.

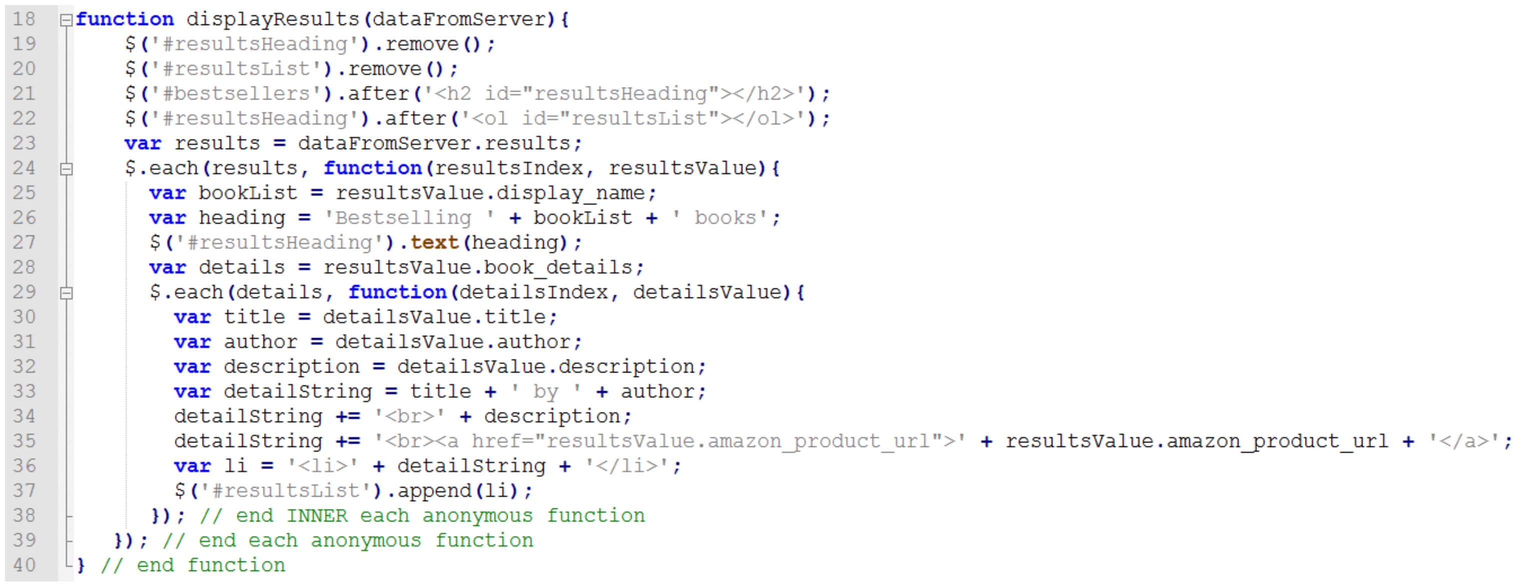
```
function displayResults(dataFromServer){
  $('#resultsHeading).remove();
  $('#resultsList').remove();
  $('#bestsellers').after('<h2 id="resultsHeading"></h2>');
  $('#resultsHeading').after('<ol id="resultsList"></ol>');
   var results = dataFromServer.results;
```

27. Finally, let’s create a new LI for each book and add the LI to the OL. Back down towards the bottom of this function where we left off before.

```
   var details = resultsValue.book_details;
    $.each(details, function(detailsIndex, detailsValue){
     var title = detailsValue.title;
     var author = detailsValue.author;
     var description = detailsValue.description;
     var detailString = title + ' by ' + author;
     detailString += '<br>' + description;
     detailString += '<br>' + description;
     detailString += '<br><a href="resultsValue.amazon_product_url"
target="_blank">' + resultsValue.amazon_product_url + '</a>';
    var li = '<li>' + detailString + '</li>';
    $('#resultsList').append(li);
       }); // end INNER each anonymous function
```

28. Save and refresh the page. Test both select menu options and each should return a different list of books.

29. The final code:



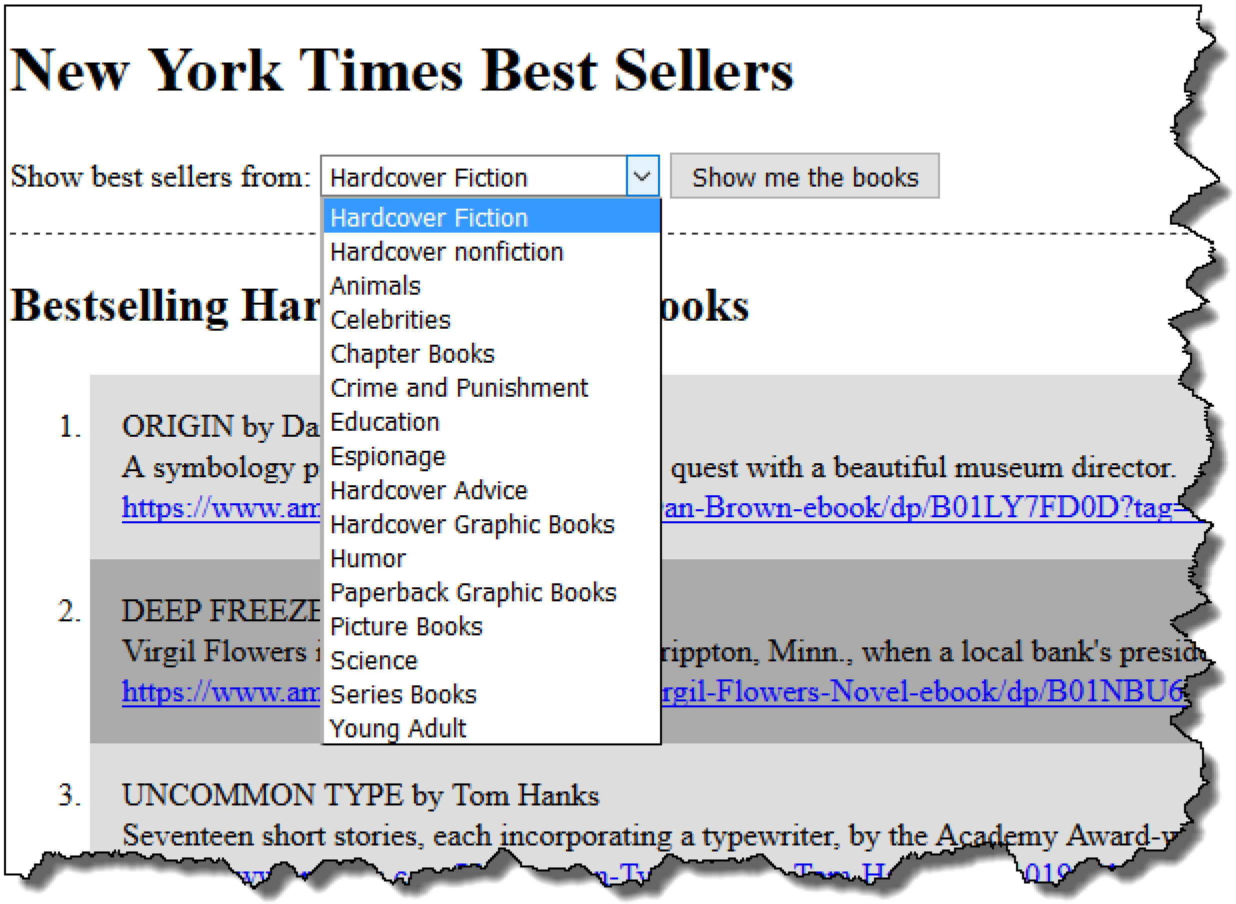
```
function displayResults(dataFromServer){
    $('#resultsHeading').remove();
    $('#resultsList').remove();
    $('#bestsellers').after('<h2 id="resultsHeading"></h2>');
    $('#resultsHeading').after('<ol id="resultsList"></ol>');
    var results = dataFromServer.results;
    $.each(results, function(resultsIndex, resultsValue){
      var bookList = resultsValue.display_name;
      var heading = 'Bestselling ' + bookList + ' books';
      $('#resultsHeading').text(heading);
      var details = resultsValue.book_details;
      $.each(details, function(detailsIndex, detailsValue){
        var title = detailsValue.title;
        var author = detailsValue.author;
        var description = detailsValue.description;
        var detailString = title + ' by ' + author;
        detailString += '<br>' + description;
        detailString += '<br><a href="resultsValue.amazon_product_url">' + resultsValue.amazon_product_url + '</a>';
        var li = '<li>' + detailString + '</li>';
        $('#resultsList').append(li);
      }); // end INNER each anonymous function
   }); // end each anonymous function
} // end function
```

30. The last thing to tweak may be some CSS to nicely format the list. Link a new CSS file to your HTML file and add this CSS code, or write your own.

```
#bestsellers {
     border-bottom: 1px dashed #333;
     padding-bottom: 1em;
}

#resultsList li {
     background-color: #aaa;
     padding: 1em;
}

#resultsList li:nth-child(odd) {
     background-color: #ddd;
}
```

31. Edit your HTML select menu to include options for:

    a. Animals
    b. Celebrities
    c. Chapter Books
    d. Crime and Punishment
    e. Education
    f. Espionage
    g. Hardcover Advice
    h. Hardcover Graphic Books
    i. Humor
    j. Paperback Graphic Books
    k. Picture Books
    l. Science
    m. Series Books
    n. Young Adult



New York Times Best Sellers

Show best sellers from: Hardcover Fiction | Show me the books

Bestselling Har ooks

1. ORIGIN by Da
A symbology p quest with a beautiful museum director.
https://www.am an-Brown-ebook/dp/B01LY7FD0D?tag=

2. DEEP FREEZE
Virgil Flowers i rippton, Minn., when a local bank's presid
https://www.am rgil-Flowers-Novel-ebook/dp/B01NBU6

3. UNCOMMON TYPE by Tom Hanks
Seventeen short stories, each incorporating a typewriter, by the Academy Award-w

Summary

Of course, your results page has little CSS formatting and no sorting controls – so there is still work to be done. But – you should at least have a feel for working with an API now. So let's see what you've learned. Next page – you are on your own!

On Your Own

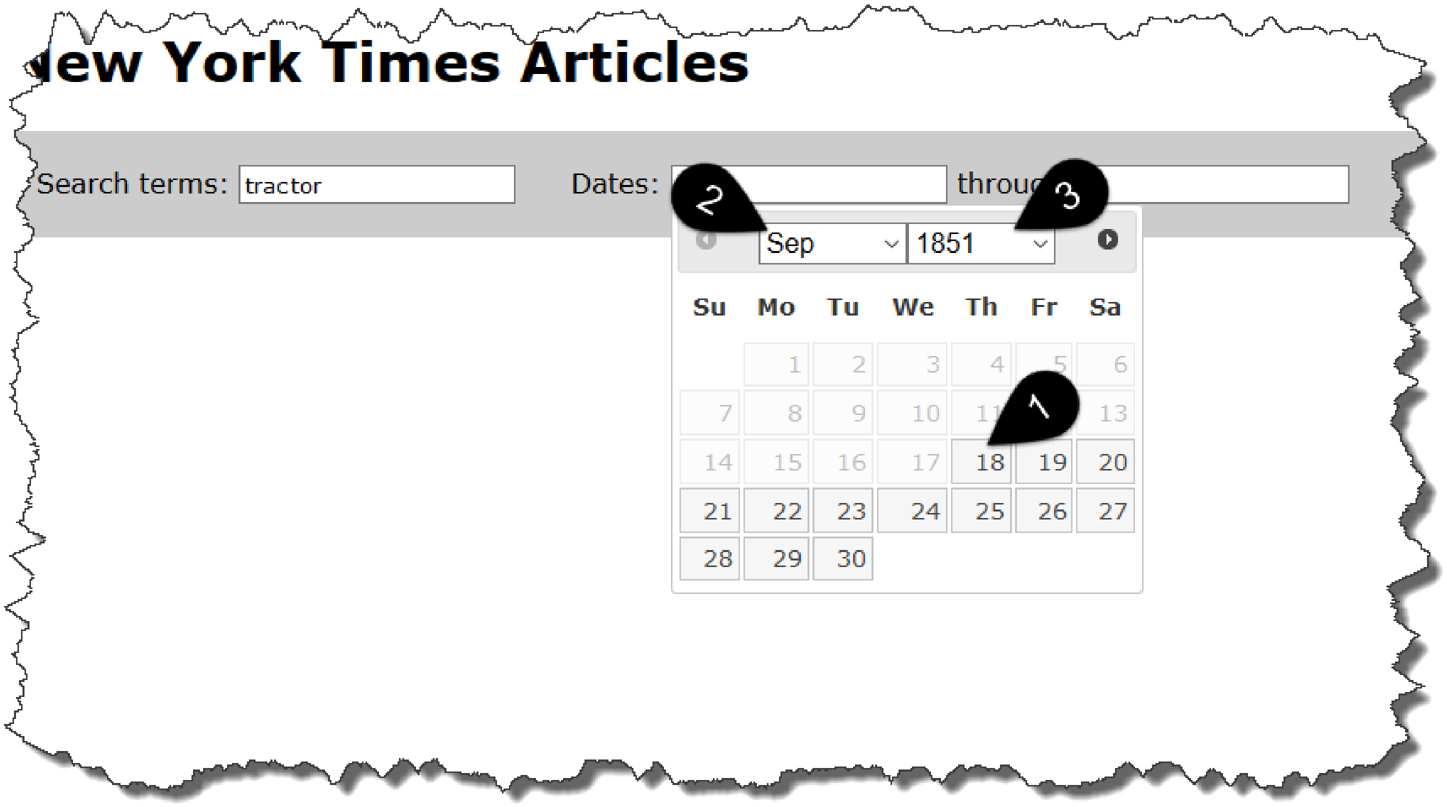
Now that you have an idea of how this works, create a new HTML page to search the New York Times' articles database. You may use the same API key.

- Be sure to read the documentation to learn what parameters are available/required.
- Use the ***article_search*** API
    - Use the **Try It Out** button on the documentation page to learn the base URL as it is different from the book API.
- Allow users to type in a search term.
- Allow users to enter start and end dates to define the date range for the article.
    - Integrate the jQuery datepicker widget
    - Set the minimum date to Sept 18, 1851 (the oldest date supported by the API)
    - Set the maximum date to today (can't have users searching for articles in the future)
- Include pagination to allow users to navigate to the next (or previous) 10 articles (the API returns 10 at a time)
- Make it usable!
    - Calendar widget
    - Disable Previous/Next links if there are no previous/next articles
    - Display the search term and date range specified by the user
    - Display useful header information, like "Viewing 11-20 of 723 articles"
- Format the page nicely with CSS.

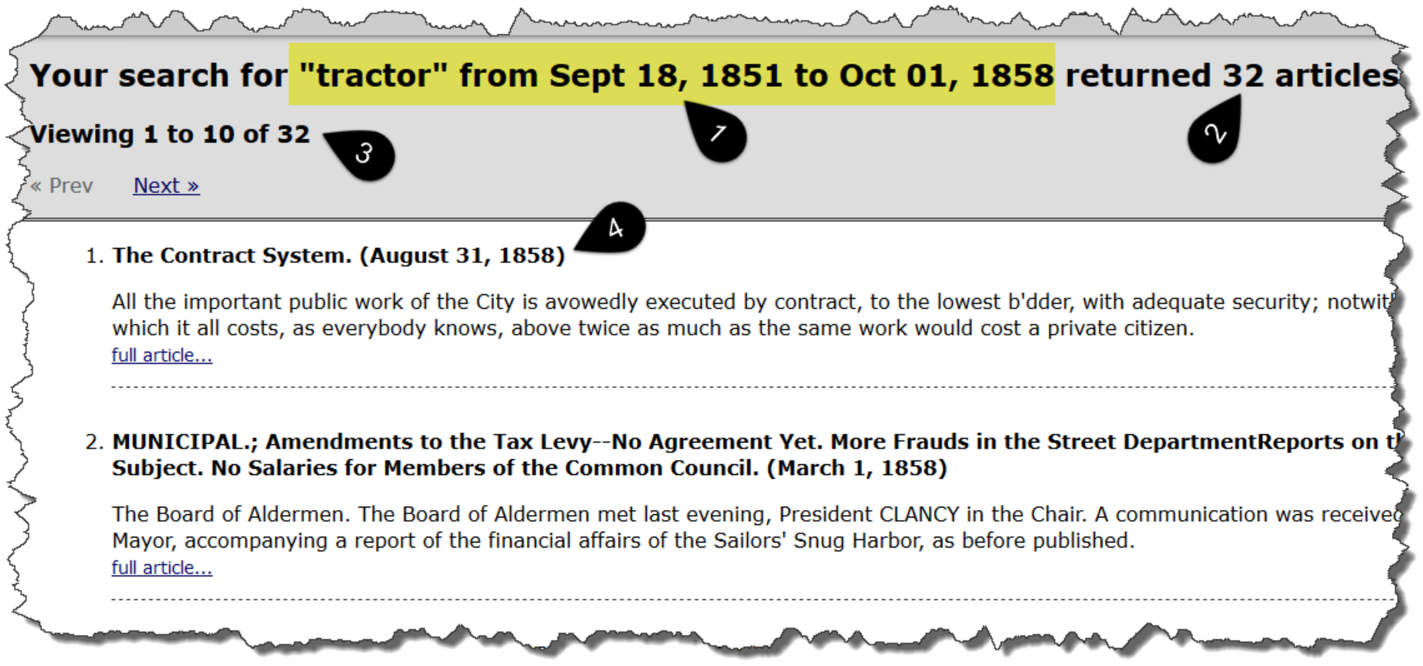
Start simple. Practice parsing the JSON data file and just get a single page of results to show up. Once you have a page of results, you can problem-solve the logic and arithmetic to get other details and pagination as in the following samples.

The next page shows some screenshots of my version. Yours will differ. Get creative. You have considerable creative freedom for this project.

- jQuery datepicker widget sets minimum date to Sept 18, 1851.
- Both month and year are available as drop-down menus.



- Header information displays search term and date range specified by user.
- Total results count is displayed.
- "Now viewing" count is displayed and the Previous link is disabled on "page 1" (and Next link is disabled on the last page.)
- Each result displays a title, its publication date, a snippet from the article, and a link to the full article on the NYT web site.



Your search for "tractor" from Sept 18, 1851 to Oct 01, 1858 returned 32 articles

Viewing 1 to 10 of 32

« Prev Next »

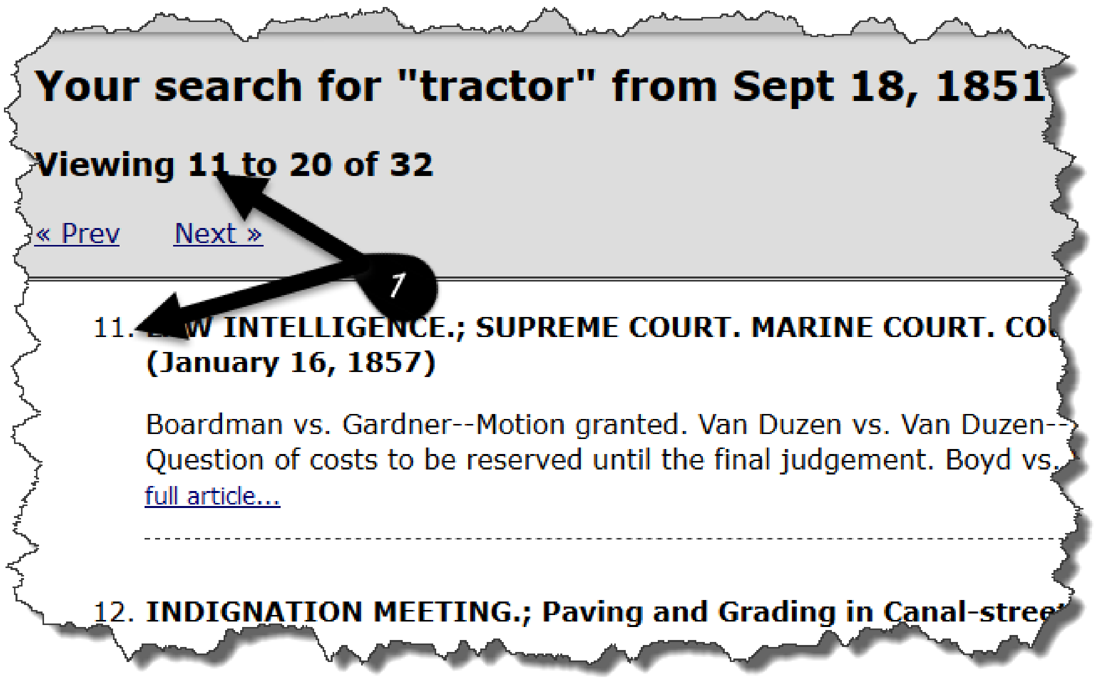
1. **The Contract System. (August 31, 1858)**

   All the important public work of the City is avowedly executed by contract, to the lowest b'dder, with adequate security; notwit which it all costs, as everybody knows, above twice as much as the same work would cost a private citizen.
   full article...

2. **MUNICIPAL.; Amendments to the Tax Levy--No Agreement Yet. More Frauds in the Street DepartmentReports on t Subject. No Salaries for Members of the Common Council. (March 1, 1858)**

   The Board of Aldermen. The Board of Aldermen met last evening, President CLANCY in the Chair. A communication was receive Mayor, accompanying a report of the financial affairs of the Sailors' Snug Harbor, as before published.
   full article...

- First list item on the page matches the starting ‘now viewing’ number



Your search for "tractor" from Sept 18, 1851

Viewing 11 to 20 of 32

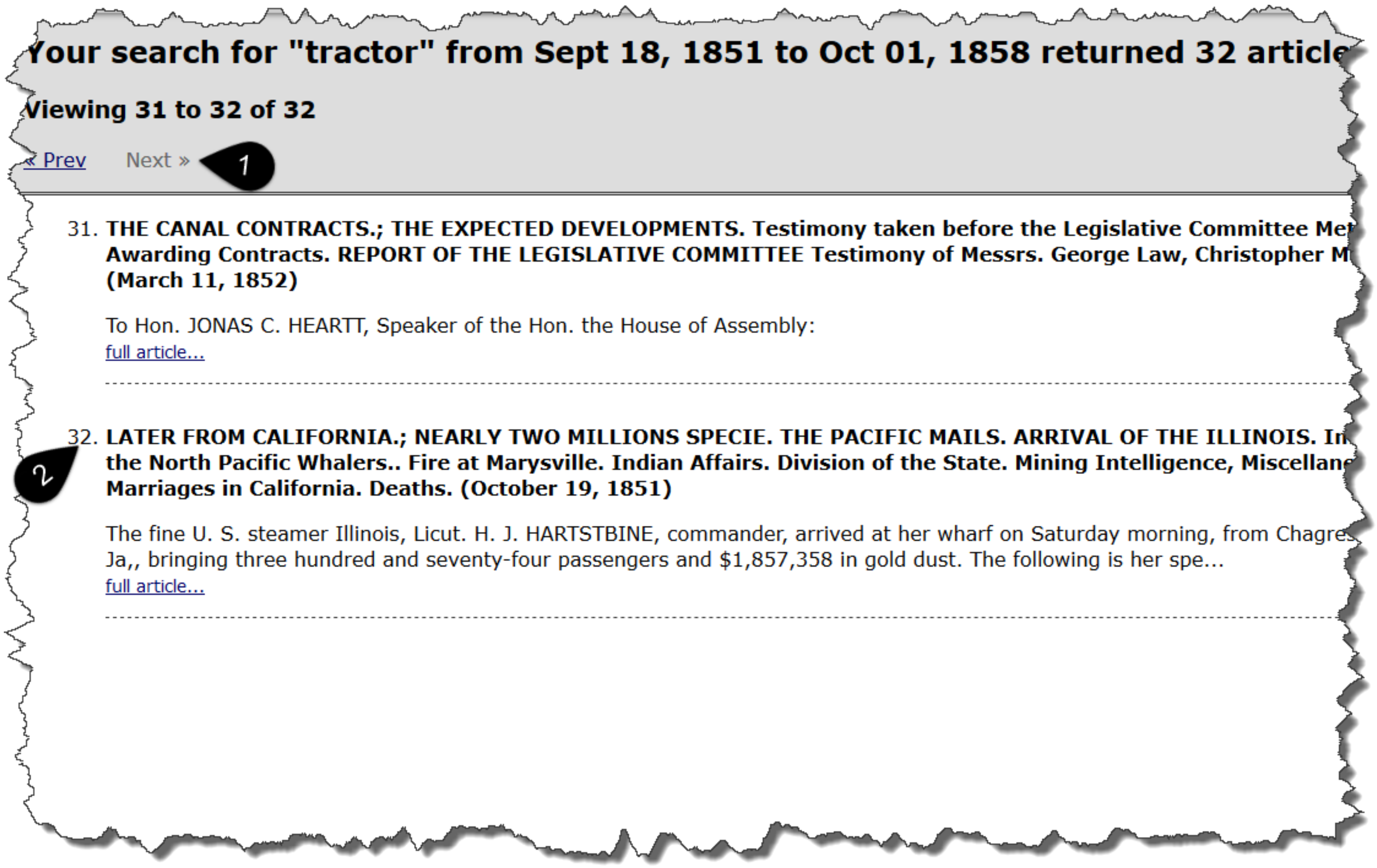
« Prev   Next »

11. LAW INTELLIGENCE.; SUPREME COURT. MARINE COURT. CO
(January 16, 1857)

Boardman vs. Gardner--Motion granted. Van Duzen vs. Van Duzen--
Question of costs to be reserved until the final judgement. Boyd vs.
full article...

12. INDIGNATION MEETING.; Paving and Grading in Canal-stree

- ‘Next’ link is disabled on last page.
- List items match the “now viewing” numbers.



Your search for "tractor" from Sept 18, 1851 to Oct 01, 1858 returned 32 article

Viewing 31 to 32 of 32

« Prev   Next »

31. THE CANAL CONTRACTS.; THE EXPECTED DEVELOPMENTS. Testimony taken before the Legislative Committee Me
Awarding Contracts. REPORT OF THE LEGISLATIVE COMMITTEE Testimony of Messrs. George Law, Christopher M
(March 11, 1852)

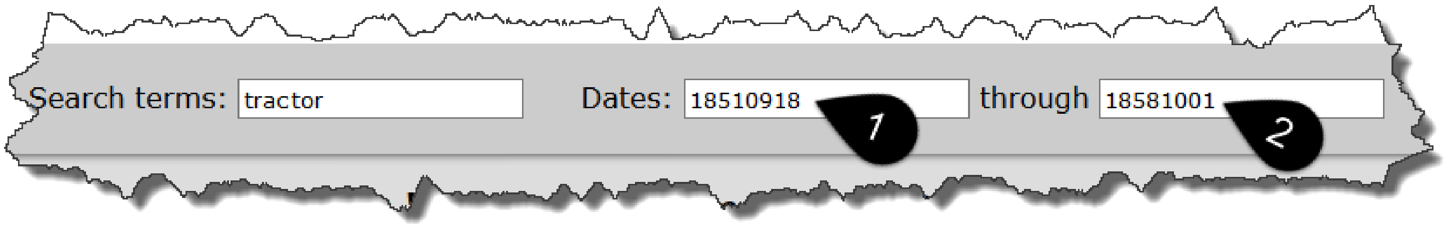
To Hon. JONAS C. HEARTT, Speaker of the Hon. the House of Assembly:
full article...

32. LATER FROM CALIFORNIA.; NEARLY TWO MILLIONS SPECIE. THE PACIFIC MAILS. ARRIVAL OF THE ILLINOIS. In
the North Pacific Whalers.. Fire at Marysville. Indian Affairs. Division of the State. Mining Intelligence, Miscellan
Marriages in California. Deaths. (October 19, 1851)

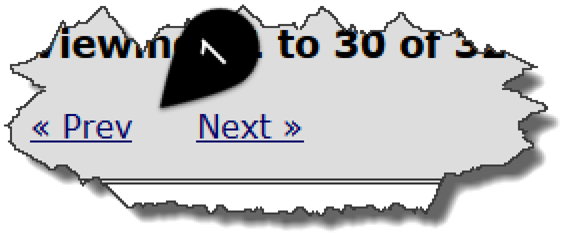
The fine U. S. steamer Illinois, Licut. H. J. HARTSTBINE, commander, arrived at her wharf on Saturday morning, from Chagre
Ja,, bringing three hundred and seventy-four passengers and $1,857,358 in gold dust. The following is her spe...
full article...

Possible improvements:

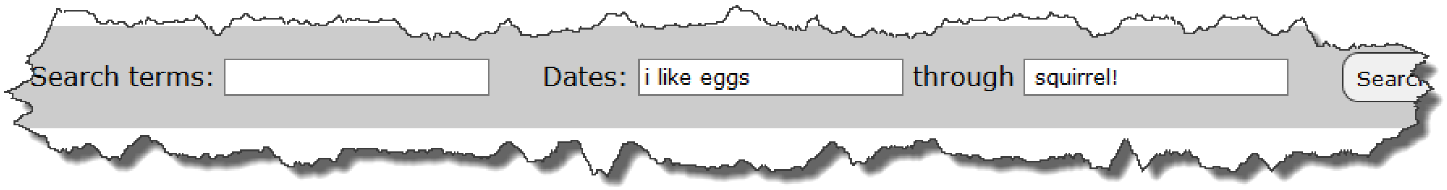
- Make the date display in a user-friendly format rather than yyyymmdd.



- Determine the total number of pages and let users navigate to a specific page. So instead of just having **Previous** and **Next** links, have links like **<< Previous 1 2 3 4 Next >>**.



- Add form validation to ensure users type a valid date and search term.


Search terms:
Dates:
i like eggs
through
squirrel!
Searc