JavaScript CRUD Tutorial – Step by Step Guide!

  
Previously, we learned how to create a simple [REST API in PHP](https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html). Today, we are going to learn how to create or insert, read, update and delete records with our JavaScript CRUD tutorial. We will use JavaScript, JSON, and PHP.

**Contents**[[hide](https://codeofaninja.com/javascript-crud-tutorial/)]

* [1 Overview](https://codeofaninja.com/javascript-crud-tutorial/#Overview)
* [2 Program Output – PHP AJAX CRUD Tutorial](https://codeofaninja.com/javascript-crud-tutorial/#Program_Output_8211_PHP_AJAX_CRUD_Tutorial)
* [3 Set Up The REST API](https://codeofaninja.com/javascript-crud-tutorial/#Set_Up_The_REST_API)
* [4 Basic Files & Folders](https://codeofaninja.com/javascript-crud-tutorial/#Basic_Files_Folders)
  + [4.1 File Structure](https://codeofaninja.com/javascript-crud-tutorial/#File_Structure)
  + [4.2 Create index.html file](https://codeofaninja.com/javascript-crud-tutorial/#Create_indexhtml_file)
  + [4.3 Create custom CSS file](https://codeofaninja.com/javascript-crud-tutorial/#Create_custom_CSS_file)
  + [4.4 Use jQuery, Bootstrap, and Bootbox.js libraries](https://codeofaninja.com/javascript-crud-tutorial/#Use_jQuery_Bootstrap_and_Bootboxjs_libraries)
  + [4.5 Create app.js file](https://codeofaninja.com/javascript-crud-tutorial/#Create_appjs_file)
  + [4.6 Create “products” scripts](https://codeofaninja.com/javascript-crud-tutorial/#Create_8220products8221_scripts)
  + [4.7 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output)
* [5 How To Read JSON Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Read_JSON_Data_Using_jQuery_AJAX)
  + [5.1 Show products on first page load](https://codeofaninja.com/javascript-crud-tutorial/#Show_products_on_first_page_load)
  + [5.2 Show products on click of a button](https://codeofaninja.com/javascript-crud-tutorial/#Show_products_on_click_of_a_button)
  + [5.3 Create showProducts() function](https://codeofaninja.com/javascript-crud-tutorial/#Create_showProducts_function)
  + [5.4 Get list of products](https://codeofaninja.com/javascript-crud-tutorial/#Get_list_of_products)
  + [5.5 Add “Create Product” button](https://codeofaninja.com/javascript-crud-tutorial/#Add_8220Create_Product8221_button)
  + [5.6 Build HTML table](https://codeofaninja.com/javascript-crud-tutorial/#Build_HTML_table)
  + [5.7 Build table row per record](https://codeofaninja.com/javascript-crud-tutorial/#Build_table_row_per_record)
  + [5.8 Inject to page content](https://codeofaninja.com/javascript-crud-tutorial/#Inject_to_page_content)
  + [5.9 5.9 Change page title](https://codeofaninja.com/javascript-crud-tutorial/#59_Change_page_title)
  + [5.10 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-2)
* [6 How To Create or Insert Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Create_or_Insert_Data_Using_jQuery_AJAX)
  + [6.1 Handle “Create Product” Button Click](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220Create_Product8221_Button_Click)
  + [6.2 Get list of categories from API](https://codeofaninja.com/javascript-crud-tutorial/#Get_list_of_categories_from_API)
  + [6.3 Build “categories option” select field](https://codeofaninja.com/javascript-crud-tutorial/#Build_8220categories_option8221_select_field)
  + [6.4 Add “Read Products” button](https://codeofaninja.com/javascript-crud-tutorial/#Add_8220Read_Products8221_button)
  + [6.5 Build “Create Product” HTML Form](https://codeofaninja.com/javascript-crud-tutorial/#Build_8220Create_Product8221_HTML_Form)
  + [6.6 Show “Create Product” form and change page title](https://codeofaninja.com/javascript-crud-tutorial/#Show_8220Create_Product8221_form_and_change_page_title)
  + [6.7 Handle “create product” form submit](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220create_product8221_form_submit)
  + [6.8 Get form data](https://codeofaninja.com/javascript-crud-tutorial/#Get_form_data)
  + [6.9 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-3)
* [7 How To Read One Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Read_One_Data_Using_jQuery_AJAX)
  + [7.1 Handle “read one” button click](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220read_one8221_button_click)
  + [7.2 Get product ID](https://codeofaninja.com/javascript-crud-tutorial/#Get_product_ID)
  + [7.3 Read one record from API](https://codeofaninja.com/javascript-crud-tutorial/#Read_one_record_from_API)
  + [7.4 Add “read products” button](https://codeofaninja.com/javascript-crud-tutorial/#Add_8220read_products8221_button)
  + [7.5 Show record data in HTML table](https://codeofaninja.com/javascript-crud-tutorial/#Show_record_data_in_HTML_table)
  + [7.6 Show “Read One Product” HTML table and change page title](https://codeofaninja.com/javascript-crud-tutorial/#Show_8220Read_One_Product8221_HTML_table_and_change_page_title)
  + [7.7 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-4)
* [8 How To Update Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Update_Data_Using_jQuery_AJAX)
  + [8.1 Handle “udpate product” button click](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220udpate_product8221_button_click)
  + [8.2 8.2 Get product ID](https://codeofaninja.com/javascript-crud-tutorial/#82_Get_product_ID)
  + [8.3 Read product information](https://codeofaninja.com/javascript-crud-tutorial/#Read_product_information)
  + [8.4 Get list of categories](https://codeofaninja.com/javascript-crud-tutorial/#Get_list_of_categories)
  + [8.5 Add “Read Products” button](https://codeofaninja.com/javascript-crud-tutorial/#Add_8220Read_Products8221_button-2)
  + [8.6 Build “Update Product” form](https://codeofaninja.com/javascript-crud-tutorial/#Build_8220Update_Product8221_form)
  + [8.7 Show “Update Product” form and change page title](https://codeofaninja.com/javascript-crud-tutorial/#Show_8220Update_Product8221_form_and_change_page_title)
  + [8.8 Handle “udpate product” form submission](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220udpate_product8221_form_submission)
  + [8.9 Get form data](https://codeofaninja.com/javascript-crud-tutorial/#Get_form_data-2)
  + [8.10 Send form data to server](https://codeofaninja.com/javascript-crud-tutorial/#Send_form_data_to_server)
  + [8.11 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-5)
* [9 How To Delete Data Using AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Delete_Data_Using_AJAX)
  + [9.1 Handle “Delete Product” button click](https://codeofaninja.com/javascript-crud-tutorial/#Handle_8220Delete_Product8221_button_click)
  + [9.2 Get product ID](https://codeofaninja.com/javascript-crud-tutorial/#Get_product_ID-2)
  + [9.3 Show “delete confirmation” dialog box](https://codeofaninja.com/javascript-crud-tutorial/#Show_8220delete_confirmation8221_dialog_box)
  + [9.4 Delete record using API](https://codeofaninja.com/javascript-crud-tutorial/#Delete_record_using_API)
  + [9.5 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-6)
* [10 How To Search Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Search_Data_Using_jQuery_AJAX)
  + [10.1 Include two JS file in index.html](https://codeofaninja.com/javascript-crud-tutorial/#Include_two_JS_file_in_indexhtml)
  + [10.2 Create products.js file](https://codeofaninja.com/javascript-crud-tutorial/#Create_productsjs_file)
  + [10.3 Create search-product.js file](https://codeofaninja.com/javascript-crud-tutorial/#Create_search-productjs_file)
  + [10.4 Change read-products.js](https://codeofaninja.com/javascript-crud-tutorial/#Change_read-productsjs)
  + [10.5 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-7)
* [11 How To Paginate Data Using jQuery AJAX?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Paginate_Data_Using_jQuery_AJAX)
  + [11.1 Change JSON URL](https://codeofaninja.com/javascript-crud-tutorial/#Change_JSON_URL)
  + [11.2 Show products using JSON URL](https://codeofaninja.com/javascript-crud-tutorial/#Show_products_using_JSON_URL)
  + [11.3 Add Pagination HTML](https://codeofaninja.com/javascript-crud-tutorial/#Add_Pagination_HTML)
  + [11.4 Output](https://codeofaninja.com/javascript-crud-tutorial/#Output-8)
* [12 How To Run The Source Code?](https://codeofaninja.com/javascript-crud-tutorial/#How_To_Run_The_Source_Code)
* [13 Download LEVEL 1 Source Code](https://codeofaninja.com/javascript-crud-tutorial/#Download_LEVEL_1_Source_Code)
* [14 Download LEVEL 2 Source Code](https://codeofaninja.com/javascript-crud-tutorial/#Download_LEVEL_2_Source_Code)
* [15 Download LEVEL 3 Source Code](https://codeofaninja.com/javascript-crud-tutorial/#Download_LEVEL_3_Source_Code)
* [16 Why download?](https://codeofaninja.com/javascript-crud-tutorial/#Why_download)
* [17 What’s Next?](https://codeofaninja.com/javascript-crud-tutorial/#What8217s_Next)
* [18 Related Tutorials](https://codeofaninja.com/javascript-crud-tutorial/#Related_Tutorials)
* [19 Some Notes](https://codeofaninja.com/javascript-crud-tutorial/#Some_Notes)
  + [19.1 Found An Issue?](https://codeofaninja.com/javascript-crud-tutorial/#Found_An_Issue)
  + [19.2 Subscribe to CodeOfaNinja](https://codeofaninja.com/javascript-crud-tutorial/#Subscribe_to_CodeOfaNinja)
  + [19.3 Thank You!](https://codeofaninja.com/javascript-crud-tutorial/#Thank_You)
  + [19.4 Share our tutorial](https://codeofaninja.com/javascript-crud-tutorial/#Share_our_tutorial)

Overview

What Is AJAX? It stands for “[Asynchronous JavaScript and XML](https://en.wikipedia.org/wiki/Ajax_(programming))“.

I’ll try to explain it in the simplest way: Using AJAX will prevent re-loading the whole page for every button click you make. As a result, it makes the user experience better. Your web app will be faster.

Ajax is not a technology, but a group of technologies. This can include HTML, CSS, JavaScript, and server-side scripting like PHP.

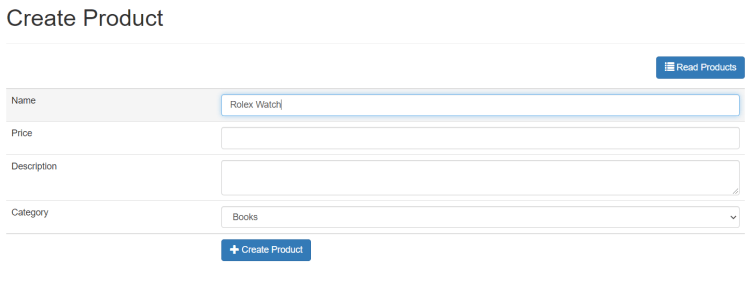
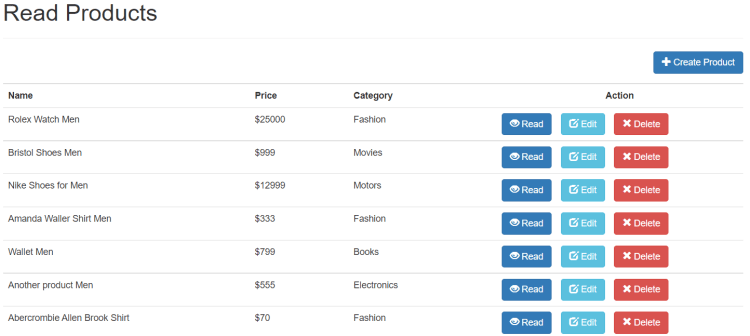
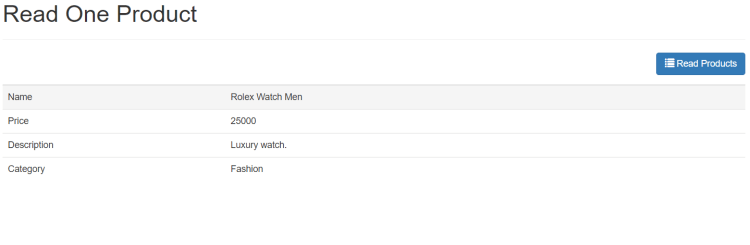
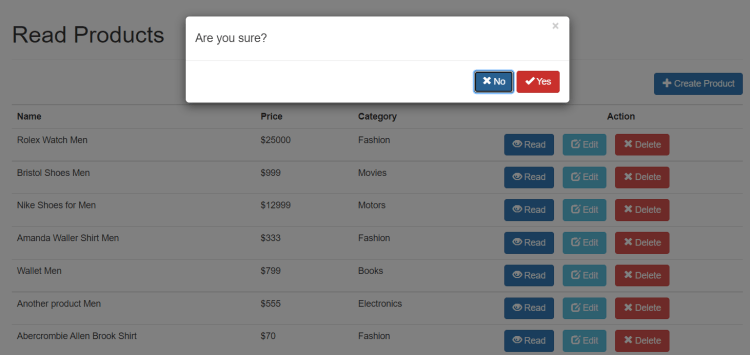
I highly recommend studying the previous tutorials first before proceeding here. But if you think you can take this one, then go on.

This tutorial will focus on creating, reading, updating, and deleting database records. We will do it using JavaScript, JSON, and PHP.

jQuery will help us with the AJAX part. JSON data will be handled by the REST API built using PHP.

Program Output – PHP AJAX CRUD Tutorial

Below are the screenshots of our tutorial’s final result. You can click an image to view the larger version of it.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-3.png?ssl=1)Create a record.[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-4.png?ssl=1)Read list of records.[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-5.png?ssl=1)Read one record.[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-7.png?ssl=1)When delete button was clicked.

Let’s proceed to the step-by-step tutorial of our LEVEL 1 source code. Enjoy!

Set Up The REST API

In this tutorial, we are going to use a REST API built with PHP.

We did not include REST API source code because we want you to focus on learning how to code with AJAX, not PHP.

But the good news is, we made a separate tutorial about how to build a simple REST API with PHP. [Click here](https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html) to learn the step-by-step PHP REST API tutorial.

I highly recommend learning our REST API tutorial first. This is because we are going to use that API for the rest of this tutorial.

But if you already have your own REST API that will work with this tutorial, that’s okay as well.

In my case, one example where I can access the REST API is: <http://localhost/api/product/read.php>

That link will show me the list of products from the database, in JSON format. It looks like the following screenshot.

[](https://i0.wp.com/www.codeofaninja.com/wp-content/uploads/2015/06/read-records-json.jpg?ssl=1)

The data above will be consumed by our AJAX app. The list of products will be displayed in the Bootstrap table with buttons like “Read One”, “Update” and “Delete”. You will see it in the “How To Read JSON Data Using jQuery AJAX?” section of this tutorial.

By the way, I’m using a Chrome extension called [JSONView](https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc" \t "_blank) to make the JSON data readable in the browser.

Basic Files & Folders

File Structure

We will have the following files and folders at the end of this LEVEL 1 source code tutorial.

├─ app/  
├─── assets/  
├────── css/  
├───────── style.css  
├────── js/  
├───────── bootbox.min.js  
├───────── jquery.js  
├─── products/  
├────── create-product.js  
├────── delete-product.js  
├────── read-one-product.js  
├────── read-products.js  
├────── update-product.js  
├─── app.js  
├─ index.html

On the next sections, we will start creating the files and folders to achieve the one above.

Create index.html file

Create index.html file on your project’s main folder. Open that file and put the following code.

|  |
| --- |
| <!**DOCTYPE** html>  <**html** lang="en">  <**head**>        <**meta** charset="utf-8">      <**meta** http-equiv="X-UA-Compatible" content="IE=edge">      <**meta** name="viewport" content="width=device-width, initial-scale=1">        <**title**>Read Products</**title**>        <!-- bootstrap CSS -->      <**link** href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>" rel="stylesheet" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">        <!-- custom CSS -->      <**link** href="app/assets/css/style.css" rel="stylesheet" />    </**head**>  <**body**>    <!-- our app will be injected here -->  <**div** id="app"></**div**>    <!-- jQuery library -->  <**script** src="<https://code.jquery.com/jquery-3.6.0.min.js>"          integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></**script**>    <!-- bootstrap JavaScript -->  <**script** src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js>"          integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous">      </**script**>    <!-- bootbox for confirm pop up -->  <**script** src="<https://cdnjs.cloudflare.com/ajax/libs/bootbox.js/4.4.0/bootbox.min.js>"></**script**>    <!-- app js script -->  <**script** src="app/app.js"></**script**>    <!-- products scripts -->  <**script** src="app/products/read-products.js"></**script**>  <**script** src="app/products/create-product.js"></**script**>  <**script** src="app/products/read-one-product.js"></**script**>  <**script** src="app/products/update-product.js"></**script**>  <**script** src="app/products/delete-product.js"></**script**>    </**body**>  </**html**> |

Create custom CSS file

1. Create “app” folder.
2. Open the “app” folder and create the “assets” folder.
3. Open the “assets” folder and create the “css” folder.
4. Open the “css” folder and create “style.css” file.

The “style.css" file is our custom CSS file. You can put any CSS in this file for additional web page styling. In our case, we have the following CSS code inside the “style.css” file.

|  |
| --- |
| .m-r-10px{ **margin-right**:10px; }  .m-b-10px{ **margin-bottom**:10px; }  .m-b-15px{ **margin-bottom**:15px; }  .m-b-20px{ **margin-bottom**:20px; }  .w-5-pct{ **width**:5%; }  .w-10-pct{ **width**:10%; }  .w-15-pct{ **width**:15%; }  .w-20-pct{ **width**:20%; }  .w-25-pct{ **width**:25%; }  .w-30-pct{ **width**:30%; }  .w-35-pct{ **width**:35%; }  .w-40-pct{ **width**:40%; }  .w-45-pct{ **width**:45%; }  .w-50-pct{ **width**:50%; }  .w-55-pct{ **width**:55%; }  .w-60-pct{ **width**:60%; }  .w-65-pct{ **width**:65%; }  .w-70-pct{ **width**:70%; }  .w-75-pct{ **width**:75%; }  .w-80-pct{ **width**:80%; }  .w-85-pct{ **width**:85%; }  .w-90-pct{ **width**:90%; }  .w-95-pct{ **width**:95%; }  .w-100-pct{ **width**:100%; }  .display-none{ **display**:none; }  .padding-bottom-2em{ **padding-bottom**:2em; }  .width-30-pct{ **width**:30%; }  .width-40-pct{ **width**:40%; }  .overflow-hidden{ **overflow**:hidden; }  .margin-right-1em{ **margin-right**:1em; }  .right-margin{ **margin**:0 .5em 0 0; }  .margin-bottom-1em { **margin-bottom**:1em; }  .margin-zero{ **margin**:0; }  .text-align-center{ **text-align**:center; } |

Use jQuery, Bootstrap, and Bootbox.js libraries

As you can see in the index.html file, we are using jQuery and Bootbox.js libraries.

jQuery JavaScript library is needed to make it easy for us to control interactions like button click and form submission. In this tutorial, we are using jQuery version 3.6.0. jQuery CDN is [here](https://code.jquery.com/).

Bootstrap makes it easy for us to have a good-looking user interface. We are using Bootstrap version 3.3.7 in this tutorial. Bootstrap CDN is [here](https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js).

Bootbox.js library is needed to make the “delete” confirmation dialog box look better. We are using Bootbox.js version 4.4.0 in this tutorial. Bootbox.js CDN is [here](http://bootboxjs.com/v4.x/getting-started.html).

Create app.js file

The “app.js” file contains some basic HTML and JavaScript functions that can be used by other JS files in our app.

1. Open “app” folder.
2. Inside that “app” folder, create “app.js” file.
3. Open “app.js” file and put the following code.

|  |
| --- |
| $(document).ready(**function**(){        // app html  **var** app\_html=`          <div **class**='container'>                <div **class**='page-header'>                  <h1 id='page-title'>Read Products</h1>              </div>                <!-- **this** is where the contents will be shown. -->              <div id='page-content'></div>            </div>`;        // inject to 'app' in index.html      $("#app").html(app\_html);    });    // change page title  **function** changePageTitle(page\_title){        // change page title      $('#page-title').text(page\_title);        // change title tag      document.title=page\_title;  }    // function to make form values to json format  $.fn.serializeObject = **function**()  {  **var** o = {};  **var** a = **this**.serializeArray();      $.each(a, **function**() {  **if** (o[**this**.name] !== undefined) {  **if** (!o[**this**.name].push) {                  o[**this**.name] = [o[**this**.name]];              }              o[**this**.name].push(**this**.value || '');          } **else** {              o[**this**.name] = **this**.value || '';          }      });  **return** o;  }; |

Create “products” scripts

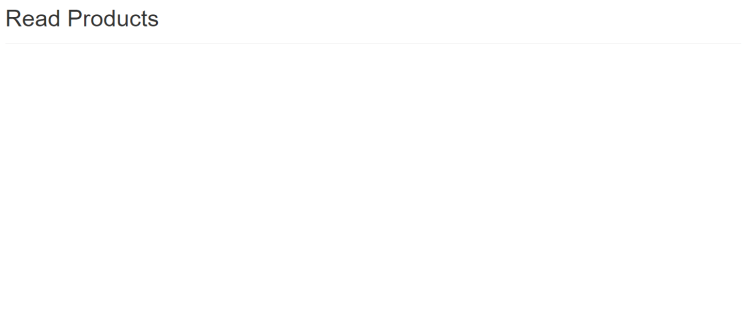
Now, we will create several JavaScript files.

1. Open “app” folder.
2. Create “products” folder inside “app” folder.
3. Create the following files inside “products” folder:
   * read-products.js
   * create-product.js
   * read-one-product.js
   * update-product.js
   * delete-product.js

What’s the code inside the JavaScript files above? For now, we will leave them empty. But we will fill them out in the next several sections of this tutorial.

Output

Our code so far will have almost empty output. It should look like the following.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-9.png?ssl=1)

How To Read JSON Data Using jQuery AJAX?

Show products on first page load

1. Open “app” folder.
2. Open “products” folder inside the “app” folder.
3. Open read-products.js file inside the “products” folder.

The following code will call the showProducts() method on first load of the web page.

The showProducts() will show the list of products in an HTML table. Put the following code inside read-products.js file.

|  |
| --- |
| $(document).ready(**function**(){        // show list of product on first load      showProducts();    });    // showProducts() method will be here |

Show products on click of a button

The following code will call showProducts() method in a click of a button with read-products-button class.

The button can be found in the “create product” and “update product” HTML template. We will see it in the next sections.

Put the following code under the showProducts(); of the previous section.

|  |
| --- |
| // when a 'read products' button was clicked  $(document).on('click', '.read-products-button', **function**(){      showProducts();  }); |

Create showProducts() function

Now we will create the showProducts() method. Replace // showProducts() method will be here comment in read-products.js file with the following code.

|  |
| --- |
| // function to show list of products  **function** showProducts(){    } |

Get list of products

The following code will contact our API to get the list of products in JSON format. Put the following code after the opening curly brace of the previous section.

|  |
| --- |
| // get list of products from the API  $.getJSON("<http://localhost/api/product/read.php>", **function**(data){    }); |

Add “Create Product” button

We have to add a “Create Product” button in the “products list” view. We will make this button work later in this tutorial.

Place the following code after the opening curly brace of the previous section.

|  |
| --- |
| // html for listing products  var read\_products\_html=`      <!-- when clicked, it will load the create product form -->      <**div** id='create-product' class='btn btn-primary pull-right m-b-15px create-product-button'>          <**span** class='glyphicon glyphicon-plus'></**span**> Create Product      </**div**> |

Build HTML table

We have to start building the HTML table where the list of products will appear.

The following code will build an HTML table with its heading. Place it after the previous section’s code.

|  |
| --- |
| <!-- start table -->  <**table** class='table table-bordered table-hover'>        <!-- creating our table heading -->      <**tr**>          <**th** class='w-25-pct'>Name</**th**>          <**th** class='w-10-pct'>Price</**th**>          <**th** class='w-15-pct'>Category</**th**>          <**th** class='w-25-pct text-align-center'>Action</**th**>      </**tr**>`;        // rows will be here    // end table  read\_products\_html+=`</**table**>`; |

Build table row per record

We will loop through each record returned by the API. In each record, we will create a table row.

Aside from product data, the table row will have the “action” buttons as well. These include the “Read One”, “Edit” and “Delete” buttons.

Replace “// rows will be here” comment of the previous section with the following code.

|  |
| --- |
| // loop through returned list of data  $.each(data.records, function(key, val) {        // creating new table row per record      read\_products\_html+=`          <**tr**>                <**td**>` + val.name + `</**td**>              <**td**>$` + val.price + `</**td**>              <**td**>` + val.category\_name + `</**td**>                <!-- 'action' buttons -->              <**td**>                  <!-- read product button -->                  <**button** class='btn btn-primary m-r-10px read-one-product-button' data-id='` + val.id + `'>                      <**span** class='glyphicon glyphicon-eye-open'></**span**> Read                  </**button**>                    <!-- edit button -->                  <**button** class='btn btn-info m-r-10px update-product-button' data-id='` + val.id + `'>                      <**span** class='glyphicon glyphicon-edit'></**span**> Edit                  </**button**>                    <!-- delete button -->                  <**button** class='btn btn-danger delete-product-button' data-id='` + val.id + `'>                      <**span** class='glyphicon glyphicon-remove'></**span**> Delete                  </**button**>              </**td**>            </**tr**>`;  }); |

Inject to page content

We have to make the HTML table appear on our webpage. We will do this by injecting the table into the “page-content” div.

Place the following code after the closing “table” tag

|  |
| --- |
| // inject to 'page-content' of our app  $("#page-content").html(read\_products\_html); |

5.9 Change page title

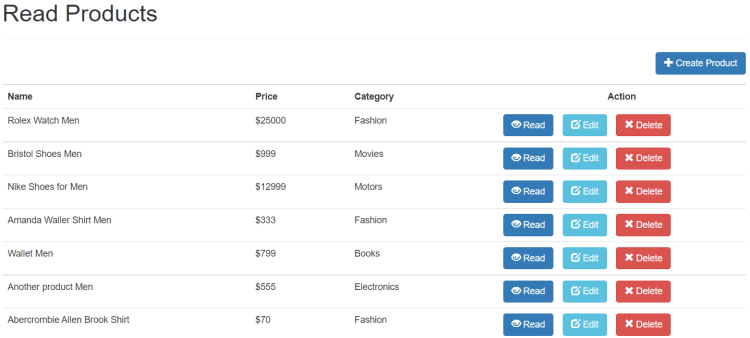
The following code will change the “title” seen on the web page and the “title” seen on the browser tab.

Place the following code after the previous section’s code.

|  |
| --- |
| // chage page title  changePageTitle("Read Products"); |

Output

After doing all the steps above, the output should look like the following.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-10.png?ssl=1)

How To Create or Insert Data Using jQuery AJAX?

Handle “Create Product” Button Click

1. Open “app” folder.
2. Open “products” folder inside the “app” folder.
3. Open create-product.js file inside the “products” folder.

The following code will handle a click of a button. This button should have “create-product-button” class.

Place the following code inside create-product.js file.

|  |
| --- |
| $(document).ready(**function**(){        // show html form when 'create product' button was clicked      $(document).on('click', '.create-product-button', **function**(){          // categories api call will be here      });        // 'create product form' handle will be here  }); |

Get list of categories from API

We need to get list of categories from the API because we will build the “categories” select field. This is where the user can select the category of the product.

Replace “// categories api call will be here” of the previous section with the following code.

|  |
| --- |
| // load list of categories  $.getJSON("<http://localhost/api/category/read.php>", **function**(data){    }); |

Build “categories option” select field

This is where we build the HTML “select” field with the “categories” option.

Place the following code after the opening curly brace of the previous section.

|  |
| --- |
| // build categories option html  // loop through returned list of data  **var** categories\_options\_html=`<select name='category\_id' **class**='form-control'>`;  $.each(data.records, **function**(key, val){      categories\_options\_html+=`<option value='` + val.id + `'>` + val.name + `</option>`;  });  categories\_options\_html+=`</select>`; |

Add “Read Products” button

The “read products” button is needed so that we can go back to the products list.

Place the following code after the previous section’s code.

|  |
| --- |
| // we have our html form here where product information will be entered  // we used the 'required' html5 property to prevent empty fields  var create\_product\_html=`        <!-- 'read products' button to show list of products -->      <**div** id='read-products' class='btn btn-primary pull-right m-b-15px read-products-button'>          <**span** class='glyphicon glyphicon-list'></**span**> Read Products      </**div**> |

Build “Create Product” HTML Form

Now we will build the actual “creat product” HTML form. This is where the user can enter the new product information that will be sent to the server.

Place the following code after the previous section’s code.

|  |
| --- |
| <!-- 'create product' html form -->  <**form** id='create-product-form' action='#' method='post' border='0'>      <**table** class='table table-hover table-responsive table-bordered'>            <!-- name field -->          <**tr**>              <**td**>Name</**td**>              <**td**><**input** type='text' name='name' class='form-control' required /></**td**>          </**tr**>            <!-- price field -->          <**tr**>              <**td**>Price</**td**>              <**td**><**input** type='number' min='1' name='price' class='form-control' required /></**td**>          </**tr**>            <!-- description field -->          <**tr**>              <**td**>Description</**td**>              <**td**><**textarea** name='description' class='form-control' required></**textarea**></**td**>          </**tr**>            <!-- categories 'select' field -->          <**tr**>              <**td**>Category</**td**>              <**td**>` + categories\_options\_html + `</**td**>          </**tr**>            <!-- button to submit form -->          <**tr**>              <**td**></**td**>              <**td**>                  <**button** type='submit' class='btn btn-primary'>                      <**span** class='glyphicon glyphicon-plus'></**span**> Create Product                  </**button**>              </**td**>          </**tr**>        </**table**>  </**form**>`; |

Show “Create Product” form and change page title

We have to make the HTML button and form appear on our web page. We will change the page title as well.

Place the following code after the previous section’s code.

|  |
| --- |
| // inject html to 'page-content' of our app  $("#page-content").html(create\_product\_html);    // chage page title  changePageTitle("Create Product"); |

Handle “create product” form submit

If the “create product” form is submitted, we need a script to handle it.

Find “// 'create product form' handle will be here” and replace it with the following code.

|  |
| --- |
| // will run if create product form was submitted  $(document).on('submit', '#create-product-form', **function**(){      // form data will be here  }); |

Get form data

This is how we get data entered in our “create product” HTML form.

Replace “// form data will be here” of the previous section with the following code.

|  |
| --- |
| // get form data  **var** form\_data=JSON.stringify($(**this**).serializeObject()); |

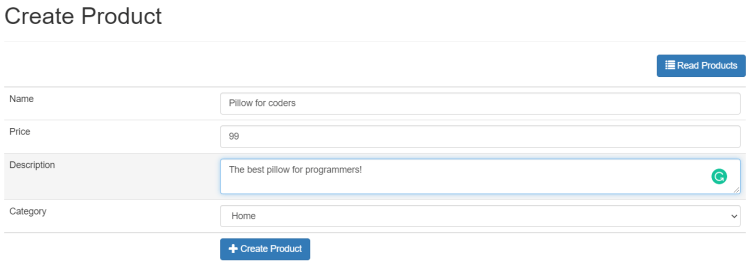
Now we will send the data to the server.

Place the following code after the previous section’s code.

|  |
| --- |
| // submit form data to api  $.ajax({      url: "<http://localhost/api/product/create.php>",      type : "POST",      contentType : 'application/json',      data : form\_data,      success : **function**(result) {          // product was created, go back to products list          showProducts();      },      error: **function**(xhr, resp, text) {          // show error to console          console.log(xhr, resp, text);      }  });    **return** **false**; |

Output

The output should look like the following.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-11.png?ssl=1)

How To Read One Data Using jQuery AJAX?

Handle “read one” button click

The “read one” button is seen on the “product list” view. When click, it should show the complete product details.

1. Open “app” folder.
2. Inside the “app” folder, open “products” folder.
3. Inside the “products” folder, open “read-one-product.js” file.

Place the following code inside “read-one-product.js” file.

|  |
| --- |
| $(document).ready(**function**(){        // handle 'read one' button click      $(document).on('click', '.read-one-product-button', **function**(){          // product ID will be here      });    }); |

Get product ID

Our script need to identify the record to be read. We do it by getting the product ID.

Replace “// product ID will be here” of the previous section with the following code.

|  |
| --- |
| // get product id  **var** id = $(**this**).attr('data-id'); |

Read one record from API

We will send the product ID to the API. It will return the data based on the given ID.

Place the following code after the previous section’s code.

|  |
| --- |
| // read product record based on given ID  $.getJSON("<http://localhost/api/product/read_one.php?id=>" + id, **function**(data){      // read products button will be here  }); |

Add “read products” button

We need the “read products” button so we can go back to the products list.

Replace “// read products button will be here” of the previous section with the following code.

|  |
| --- |
| // start html  var read\_one\_product\_html=`        <!-- when clicked, it will show the product's list -->      <**div** id='read-products' class='btn btn-primary pull-right m-b-15px read-products-button'>          <**span** class='glyphicon glyphicon-list'></**span**> Read Products      </**div**> |

Show record data in HTML table

We will place the product information returned by the API to an HTML table.

Place the following code after the previous section’s code.

|  |
| --- |
| <!-- product data will be shown in this table -->  <**table** class='table table-bordered table-hover'>        <!-- product name -->      <**tr**>          <**td** class='w-30-pct'>Name</**td**>          <**td** class='w-70-pct'>` + data.name + `</**td**>      </**tr**>        <!-- product price -->      <**tr**>          <**td**>Price</**td**>          <**td**>` + data.price + `</**td**>      </**tr**>        <!-- product description -->      <**tr**>          <**td**>Description</**td**>          <**td**>` + data.description + `</**td**>      </**tr**>        <!-- product category name -->      <**tr**>          <**td**>Category</**td**>          <**td**>` + data.category\_name + `</**td**>      </**tr**>    </**table**>`; |

Show “Read One Product” HTML table and change page title

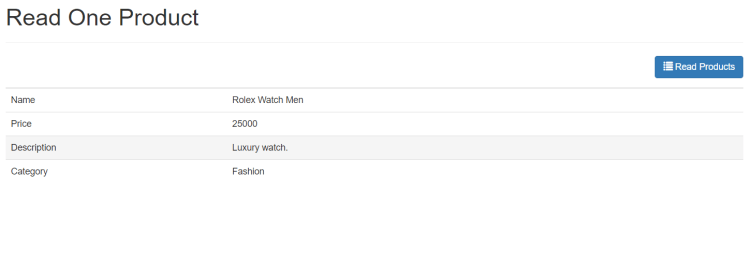
We have to make the HTML button and table appear on our web page. We will change the page title as well.

Place the following code after the previous section’s code.

|  |
| --- |
| // inject html to 'page-content' of our app  $("#page-content").html(read\_one\_product\_html);    // chage page title  changePageTitle("Create Product"); |

Output

The output should look like the following.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-12.png?ssl=1)

How To Update Data Using jQuery AJAX?

Handle “udpate product” button click

The “edit” button is seen on the “product list” view. When click, it should show the “update product” form filled out with product information.

1. Open “app” folder.
2. Inside the “app” folder, open “products” folder.
3. Inside the “products” folder, open “update-product.js” file.

Place the following code inside “update-product.js” file.

|  |
| --- |
| $(document).ready(**function**(){        // show html form when 'update product' button was clicked      $(document).on('click', '.update-product-button', **function**(){          // product ID will be here      });        // 'update product form' submit handle will be here  }); |

8.2 Get product ID

Our script need to identify the record to be read. We do it by getting the product ID.

Replace “// product ID will be here” of the previous section with the following code.

|  |
| --- |
| // get product id  **var** id = $(**this**).attr('data-id'); |

Read product information

To fill out our “update product” HTML form, we need to get product information from the API.

Place the following code after the previous section’s code.

|  |
| --- |
| // read one record based on given product id  $.getJSON("<http://localhost/api/product/read_one.php?id=>" + id, **function**(data){        // values will be used to fill out our form  **var** name = data.name;  **var** price = data.price;  **var** description = data.description;  **var** category\_id = data.category\_id;  **var** category\_name = data.category\_name;        // load list of categories will be here  }); |

Get list of categories

A list of categories is needed for product category options. Category records will be rendered as options in a “select” HTML input field.

Replace “// load list of categories will be here” of the previous section with the following code.

|  |
| --- |
| // load list of categories  $.getJSON("<http://localhost/api/category/read.php>", **function**(data){        // build 'categories option' html      // loop through returned list of data  **var** categories\_options\_html=`<select name='category\_id' **class**='form-control'>`;            $.each(data.records, **function**(key, val){              // pre-select option is category id is the same  **if**(val.id==category\_id){ categories\_options\_html+=`<option value='` + val.id + `' selected>` + val.name + `</option>`; }    **else**{ categories\_options\_html+=`<option value='` + val.id + `'>` + val.name + `</option>`; }          });          categories\_options\_html+=`</select>`;        // update product html will be here  }); |

Add “Read Products” button

The “read products” button is needed so that we can go back to the products list.

Replace “// update product html will be here” of the previous section with the following code.

|  |
| --- |
| // store 'update product' html to this variable  **var** update\_product\_html=`      <div id='read-products' **class**='btn btn-primary pull-right m-b-15px read-products-button'>          <span **class**='glyphicon glyphicon-list'></span> Read Products      </div> |

Build “Update Product” form

Now we will build the “update product” HTML form. This form will be built with an HTML table and the input fields are filled out with product information.

Place the following code after the previous section’s code.

|  |
| --- |
| <!-- build 'update product' html form -->  <!-- we used the 'required' html5 property to prevent empty fields -->  <**form** id='update-product-form' action='#' method='post' border='0'>      <**table** class='table table-hover table-responsive table-bordered'>            <!-- name field -->          <**tr**>              <**td**>Name</**td**>              <**td**><**input** value=\"` + name + `\" type='text' name='name' class='form-control' required /></**td**>          </**tr**>            <!-- price field -->          <**tr**>              <**td**>Price</**td**>              <**td**><**input** value=\"` + price + `\" type='number' min='1' name='price' class='form-control' required /></**td**>          </**tr**>            <!-- description field -->          <**tr**>              <**td**>Description</**td**>              <**td**><**textarea** name='description' class='form-control' required>` + description + `</**textarea**></**td**>          </**tr**>            <!-- categories 'select' field -->          <**tr**>              <**td**>Category</**td**>              <**td**>` + categories\_options\_html + `</**td**>          </**tr**>            <**tr**>                <!-- hidden 'product id' to identify which record to delete -->              <**td**><**input** value=\"` + id + `\" name='id' type='hidden' /></**td**>                <!-- button to submit form -->              <**td**>                  <**button** type='submit' class='btn btn-info'>                      <**span** class='glyphicon glyphicon-edit'></**span**> Update Product                  </**button**>              </**td**>            </**tr**>        </**table**>  </**form**>`; |

Show “Update Product” form and change page title

We need to show our “update product” HTML on our webpage. We will change the page title as well.

Put the following code after the previous section’s code.

|  |
| --- |
| // inject to 'page-content' of our app  $("#page-content").html(update\_product\_html);    // chage page title  changePageTitle("Update Product"); |

Handle “udpate product” form submission

If the “update product” form is submitted, we need a script to handle it.

Find “// 'update product form‘ submit handle will be here” and replace it with the following code.

|  |
| --- |
| // will run if 'create product' form was submitted  $(document).on('submit', '#update-product-form', **function**(){        // get form data will be here    **return** **false**;  }); |

Get form data

We will get the product information from our “update product” HTML form.

Replace “// get form data will be here” of the previous section with the following code.

|  |
| --- |
| // get form data  **var** form\_data=JSON.stringify($(**this**).serializeObject()); |

Send form data to server

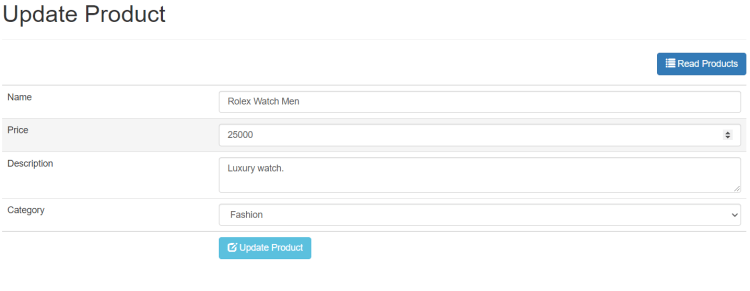
After getting the form data, we will send the data to our API.

Place the following code after the previous section’s code.

|  |
| --- |
| // submit form data to api  $.ajax({      url: "<http://localhost/api/product/update.php>",      type : "POST",      contentType : 'application/json',      data : form\_data,      success : **function**(result) {          // product was created, go back to products list          showProducts();      },      error: **function**(xhr, resp, text) {          // show error to console          console.log(xhr, resp, text);      }  }); |

Output

The output should look like the following.

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-13.png?ssl=1)

How To Delete Data Using AJAX?

Handle “Delete Product” button click

The “delete product” button is seen in the “read products” view. When it was clicked, we need to handle it.

1. Open “app” folder.
2. Inside the “app” folder, open “products” folder.
3. Inside the “products” folder, open “delete-product.js” file.

Place the following code inside “delete-product.js” file.

|  |
| --- |
| $(document).ready(**function**(){        // will run if the delete button was clicked      $(document).on('click', '.delete-product-button', **function**(){          // product id will be here      });  }); |

Get product ID

The product ID is needed to identify which record to delete using the API.

Replace “// product id will be here” of the previous section with the following code.

|  |
| --- |
| // get the product id  **var** product\_id = $(**this**).attr('data-id'); |

Show “delete confirmation” dialog box

This is where we will use the Bootbox.js library. We will show a dialog box with “Are you sure?” message with “Yes” and “No” buttons.

Place the following code after the previous section’s code.

|  |
| --- |
| // bootbox for good looking 'confirm pop up'  bootbox.confirm({        message: "<h4>Are you sure?</h4>",      buttons: {          confirm: {              label: '<span class="glyphicon glyphicon-ok"></span> Yes',              className: 'btn-danger'          },          cancel: {              label: '<span class="glyphicon glyphicon-remove"></span> No',              className: 'btn-primary'          }      },      callback: **function** (result) {          // delete request will be here      }  }); |

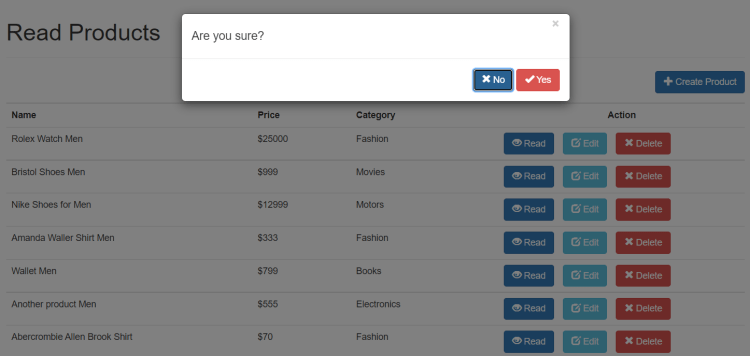
Delete record using API

If the user clicked “Yes” on the dialog box, a “delete” request will be sent to the API.

Replace “// delete request will be here” of the previous section with the following code.

|  |
| --- |
| **if**(result==**true**){        // send delete request to api / remote server      $.ajax({          url: "<http://localhost/api/product/delete.php>",          type : "POST",          dataType : 'json',          data : JSON.stringify({ id: product\_id }),          success : **function**(result) {                // re-load list of products              showProducts();          },          error: **function**(xhr, resp, text) {              console.log(xhr, resp, text);          }      });    } |

Output

[](https://i0.wp.com/codeofaninja.com/wp-content/uploads/2021/09/image-14.png?ssl=1)

How To Search Data Using jQuery AJAX?

This feature is part of our LEVEL 2 source code.

Include two JS file in index.html

|  |
| --- |
| <!-- products scripts -->  <script src="app/products/products.js"></script>  <script src="app/products/search-product.js"></script> |

Create products.js file

The “products.js” file will contain any functions that can be used by other product components like our “read-products.js” or “search-products.js” files.

Open “app” folder. Open “products” folder. Create “products.js” file.

Open the “products.js” file and put the following code.

|  |
| --- |
| // product list html  **function** readProductsTemplate(data, keywords){    **var** read\_products\_html=`          <!-- search products form -->          <form id='search-product-form' action='#' method='post'>          <div **class**='input-group pull-left w-30-pct'>                <input type='text' value='` + keywords + `' name='keywords' **class**='form-control product-search-keywords' placeholder='Search products...' />                <span **class**='input-group-btn'>                  <button type='submit' **class**='btn btn-default' type='button'>                      <span **class**='glyphicon glyphicon-search'></span>                  </button>              </span>            </div>          </form>            <!-- when clicked, it will load the create product form -->          <div id='create-product' **class**='btn btn-primary pull-right m-b-15px create-product-button'>              <span **class**='glyphicon glyphicon-plus'></span> Create Product          </div>            <!-- start table -->          <table **class**='table table-bordered table-hover'>                <!-- creating our table heading -->              <tr>                  <th **class**='w-25-pct'>Name</th>                  <th **class**='w-10-pct'>Price</th>                  <th **class**='w-15-pct'>Category</th>                  <th **class**='w-25-pct text-align-center'>Action</th>              </tr>`;        // loop through returned list of data      $.each(data.records, **function**(key, val) {            // creating new table row per record          read\_products\_html+=`<tr>                <td>` + val.name + `</td>              <td>$` + val.price + `</td>              <td>` + val.category\_name + `</td>                <!-- 'action' buttons -->              <td>                  <!-- read product button -->                  <button **class**='btn btn-primary m-r-10px read-one-product-button' data-id='` + val.id + `'>                      <span **class**='glyphicon glyphicon-eye-open'></span> Read                  </button>                    <!-- edit button -->                  <button **class**='btn btn-info m-r-10px update-product-button' data-id='` + val.id + `'>                      <span **class**='glyphicon glyphicon-edit'></span> Edit                  </button>                    <!-- **delete** button -->                  <button **class**='btn btn-danger delete-product-button' data-id='` + val.id + `'>                      <span **class**='glyphicon glyphicon-remove'></span> Delete                  </button>              </td>          </tr>`;      });        // end table      read\_products\_html+=`</table>`;        // inject to 'page-content' of our app      $("#page-content").html(read\_products\_html);  } |

Create search-product.js file

The “search-product.js” file will contain a code that catches the submission of the “search product” form.

Open “app” folder. Open the “products” folder. Create “search-products.js” file.

Open the “search-products.js” file and put the following code.

|  |
| --- |
| $(document).ready(**function**(){        // when a 'search products' button was clicked      $(document).on('submit', '#search-product-form', **function**(){            // get search keywords  **var** keywords = $(**this**).find(":input[name='keywords']").val();            // get data from the api based on search keywords          $.getJSON("<http://localhost/api/product/search.php?s=>" + keywords, **function**(data){                // template in products.js              readProductsTemplate(data, keywords);                // chage page title              changePageTitle("Search products: " + keywords);            });            // prevent whole page reload  **return** **false**;      });    }); |

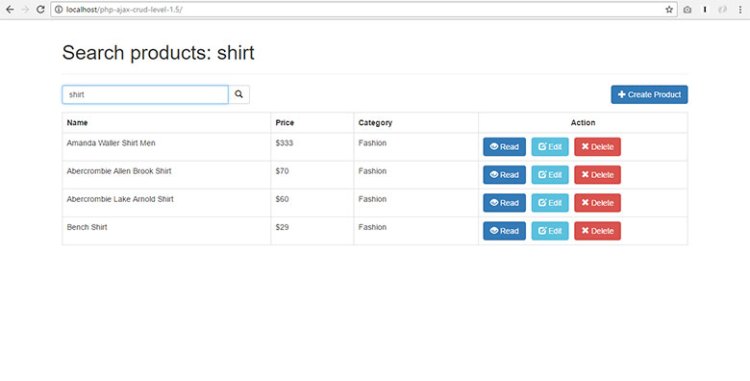
Change read-products.js

We want the “product list” and “search product” to have the same HTML table template. To do this, we will use the readProductsTemplate() function of the products.js file.

Open “app” folder. Open “products” folder. Open the “read-products.js’ file. Change the showProducts() function to the following code.

|  |
| --- |
| // function to show list of products  **function** showProducts(){        // get list of products from the API      $.getJSON("<http://localhost/api/product/read.php>", **function**(data){            // html for listing products          readProductsTemplate(data, "");            // chage page title          changePageTitle("Read Products");        });  } |

Output

[](https://i0.wp.com/www.codeofaninja.com/wp-content/uploads/2015/06/output-search-product.jpg?ssl=1)

How To Paginate Data Using jQuery AJAX?

This feature is part of our LEVEL 2 and LEVEL 3 source codes.

Change JSON URL

To make pagination work, we’ll have to change the JSON URL. The contents of this new JSON data will include the “paging” node. It looks like the following.

[](https://i0.wp.com/www.codeofaninja.com/wp-content/uploads/2015/06/json-api-paging-node.jpg?ssl=1)

So we will change the JSON URL from:

|  |
| --- |
| <http://localhost/api/product/read.php> |

to

|  |
| --- |
| <http://localhost/api/product/read_paging.php> |

It means we have to change something in our code. See the change in the next section.

Show products using JSON URL

Open /app/products/read-products.js file. Replace the code with the following.

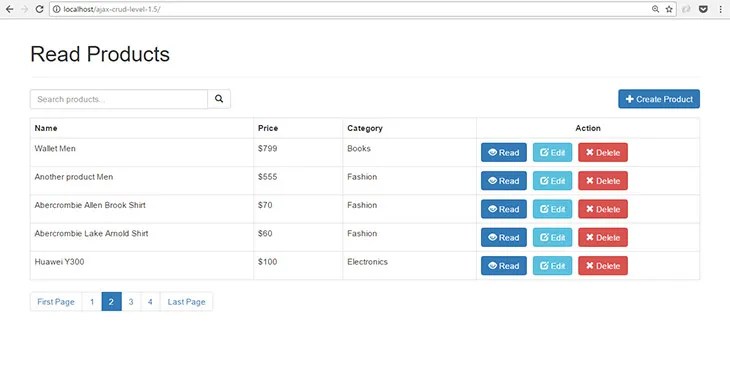
|  |
| --- |
| $(document).ready(**function**(){        // show list of product on first load      showProductsFirstPage();        // when a 'read products' button was clicked      $(document).on('click', '.read-products-button', **function**(){          showProductsFirstPage();      });        // when a 'page' button was clicked      $(document).on('click', '.pagination li', **function**(){          // get json url  **var** json\_url=$(**this**).find('a').attr('data-page');            // show list of products          showProducts(json\_url);      });    });    **function** showProductsFirstPage(){  **var** json\_url="<http://localhost/api/product/read_paging.php>";      showProducts(json\_url);  }    // function to show list of products  **function** showProducts(json\_url){        // get list of products from the API      $.getJSON(json\_url, **function**(data){            // html for listing products          readProductsTemplate(data, "");            // chage page title          changePageTitle("Read Products");        });  } |

Add Pagination HTML

Open /app/products/products.js file. Find the ending “table” tag and put the following code under it.

|  |
| --- |
| // pagination  **if**(data.paging){      read\_products\_html+="<ul class='pagination pull-left margin-zero padding-bottom-2em'>";            // first page  **if**(data.paging.first!=""){              read\_products\_html+="<li><a data-page='" + data.paging.first + "'>First Page</a></li>";          }            // loop through pages          $.each(data.paging.pages, **function**(key, val){  **var** active\_page=val.current\_page=="yes" ? "class='active'" : "";              read\_products\_html+="<li " + active\_page + "><a data-page='" + val.url + "'>" + val.page + "</a></li>";          });            // last page  **if**(data.paging.last!=""){              read\_products\_html+="<li><a data-page='" + data.paging.last + "'>Last Page</a></li>";          }      read\_products\_html+="</ul>";  } |

Output

After making the changes above, run index.html again. Do a hard refresh. You should see the paging buttons like the one below.  
[](https://i0.wp.com/www.codeofaninja.com/wp-content/uploads/2015/06/ajax-crud-tutorial-pagination.jpg?ssl=1)