(

Axios Todo Project

28 AUGUST 2017

You will be creating a Todo list.

[Use the totally rad todo API created by the legendary Bob. Here (https://coursework.vschool.io/v-schools-todo-api-documentation/)](https://coursework.vschool.io/v-schools-todo-api-documentation/) are the docs for it.

For the requirements below, you'll be building a frontend site that pulls your list of todos down from the API to display them, and allows the user to perform other **CRUD** (CREATE | READ | UPDATE | DELETE)methods on them as well.

For example, when the user adds a new todo, it will also POST that todo to the database using the provided API. This way, once the change is made, it's made permanently. When you refresh the page, since it's pulling from the data on the API, it should pull the current list of todos.

You will do the parts of this assignment in steps. You're encouraged to have someone else look at your code between each step. This will help you to become comfortable talking about code, and might open you up to other ways of doing things. Any suggestions given to you by fellow students should be considered, but don't feel obligated to apply the suggestions you get.

### Requirements

Use the **checklist in Trello** to mark off each step before proceeding to the next.

## Part 1 - GET

The user can see their current list of todos. Todos show up as soon as the page loads.

If a todo item is complete, it should have a strikethrough line on it Images should be displayed as images if there are any

Hints:

A

createTodo

Userfulfunction that takes one todo and inserts it to the **DOM** is very

Use **postman** to get those first few todos in there with some images so you can style your list!

## Part 2 - POST

The user can add new todos to their list. The new item should be posted to the todo API so a future reload of the page will still display that new todo item. Making the new todo appear without a refresh is extra credit, but you're encouraged to attempt it.

A user should be able to give the item a price.

A user should be able to give the item a description. A user should be able to attach an imgUrl to the item

## Part 3 - PUT Part 1

Each todo will have a checkbox where it can be marked complete or incomplete

Checking the checkbox should update the database

## Part 4 - DELETE

A user will be able to delete todos (this is different from marking a todo as "completed")

Each todo should be rendered with a button marked "X" or "Delete" that when clicked, will delete the Todo

## Part 5 - PUT Part 2 (extra credit)

Each Todo will have an "edit" button.

When clicked, the info will change to input boxes that are autofilled with the old Todo data

A user can change the value of these inputs

When the "edit" button is clicked, it will change to a "save" button.

When "save" is clicked, the form will disapear, and the new values will be displayed.

On save, the todo will be edited in the database

[Read through the "using id" section in the API documentation (https://coursework.vschool.io/v-schools-todo-api-documentation/)](https://coursework.vschool.io/v-schools-todo-api-documentation/) to learn how to delete items using the item's unique id.

# API Documentation

[Check our documentation on the V School Todo API (https://coursework.vschool.io/v-schools-todo-api-documentation/)](https://coursework.vschool.io/v-schools-todo-api-documentation/) to learn how to use it.

### Styling ideas:

If you want some *pretty* looking ToDo list website examples take a look [HERE (https://www.google.com/search? q=todo+list&espv=2&biw=1483&bih=865&source=lnms&tbm=isch&sa=X& ved=0CAYQ\_AUoAWoVChMIlKjcxNSQxwIVRJaICh3CjAk- #tbm=isch&q=bootstrap+todo)](https://www.google.com/search?q=todo%2Blist&amp;espv=2&amp;biw=1483&amp;bih=865&amp;source=lnms&amp;tbm=isch&amp;sa=X&amp;ved=0CAYQ_AUoAWoVChMIlKjcxNSQxwIVRJaICh3CjAk-%23tbm%3Disch&amp;q=bootstrap%2Btodo)