



Build a Todo App with Node.js, ExpressJs, MongoDB and VueJs – Part 2



Samuel James   Feb 11

[#node](#) [#mongodb](#) [#vue](#) [#expressjs](#)

Create a Todo List

Todo Items

| | | |
|-------------------------------------|---------------------------|---|
| <input checked="" type="checkbox"/> | Going Shopping | X |
| <input type="checkbox"/> | Meeting with Dave by 4p | X |
| <input type="checkbox"/> | Dinner with Girlfriend by | X |

In [part 1](#) of this tutorial, we built APIs for a simple todo application and now we are here to put the front end together with VueJS. You need not to worry if you are new to VueJs. I wrote [VueJs: The basics in 4 mins](#) and [Creating your first component in VueJs](#) to help you pick up VueJs in no time.

Packages

There is a need to bundle JS and CSS resources into one javascript bundle that will be included on index page and we'll choose webpack for this purpose. We'll use a single package.json to manage dependencies for both backend and

frontend. For large applications, you should consider having separate `package.json` for your frontend and backend.

Having said that, let's go ahead and update `package.json`.

```
{
  "name": "node-todo",
  "version": "0.1.1",
  "description": "Simple todo application.",
  "main": "server.js",
  "author": "Samuel James",
  "scripts": {
    "build": "webpack"
  },
  "dependencies": {
    "body-parser": "^1.5.2",
    "express": "~4.7.2",
    "method-override": "~2.1.2",
    "mongoose": "~3.6.2",
    "bootstrap": "^4.0.0-beta.2",
    "axios": "^0.16.2",
    "vue": "^2.5.11",
    "vue-axios": "^2.0.2"
  },
  "devDependencies": {
    "prettier": "^1.9.2",
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-preset-env": "^1.6.0",
    "babel-preset-stage-3": "^6.24.1",
    "cross-env": "^5.0.5",
    "css-loader": "^0.28.7",
    "file-loader": "^1.1.4",
    "vue-loader": "^13.0.5",
    "vue-template-compiler": "^2.4.4",
    "webpack": "^3.6.0",
    "webpack-dev-server": "^2.9.1"
  }
}
```

```
}
```

From console, run `npm install` to install dependencies.

Create an '*src*' folder in public folder. We'll put all our source code for frontend in this `src` folder.

Basically, we need 2 major vue components: one for creating new items and the other for listing, updating and deleting todo items.

At some points, these components would need to communicate or share data with each other and this where event bus comes into play.

One of the best ways to handle communications between components in Vue.js is to use a global event bus such that when a component emits an event, an event bus transmits this event to other listening components.

Event Bus

We create a global event bus with this code :

```
//path/to/project/public/bus.js
'use strict'

import Vue from 'vue'

const bus = new Vue()
```

```
export default bus
```

Create Todo Component

Now that we have event bus created, let write the code for adding new todo items.

```
//public/src/components/CreateTodo.vue
```

```
<template>
  <div>
    <h2>Create a Todo List</h2>
    <form @submit.prevent>
      <div class="form-group">
        <input type="text" class="form-control" @keypress="typing=true" placeholder="Add new todo item" />
        <span class="help-block small text-center" v-show="typing">Hit enter to add item</span>
      </div>
    </form>
  </div>
</template>

<script>
  import axios from 'axios';
  import bus from "../../bus.js";

  export default {
    data() {
      return {
        todo: '',
        typing: false,
      }
    },
    methods: {
      addTodo(event) {
        if (event) event.preventDefault();
        let url = 'http://localhost:4000/api/add';
```

```
let param = {
  name: this.todo,
  done: 0
};

axios.post(url, param).then((response) => {
  console.log(response);
  this.clearTodo();
  this.refreshTodo();
  this.typing = false;
}).catch((error) => {
  console.log(error);
})

},
clearTodo() {
  this.todo = '';
},
refreshTodo() {
  bus.$emit("refreshTodo")
}
}
}
</script>
```

That out of the way, we also need to write code to get and also update todo items in the database.

List Todo Component

Create a file *ListTodo.vue* in components' folder:

```
//public/src/components/ListTodo.vue

<template>
  <div>
    <div class="col-md-12" v-show="todos.length>0">
      <h3>Todo Items</h3>
```

```

<div class="row mrb-10" v-for="todo in todos">
  <div class="input-group m-b-5">
    <span class="input-group-addon addon-right"><input type="checkbox" v-model="todo.done" /></span>
    <input type="text" class="form-control" :class="todo.done ? 'form-control' : 'form-control'"/>
    <span class="input-group-addon addon-left" title="Delete" v-on:click="delete(todo._id)"></span>
  </div>
  <span class="help-block small" v-show="todo.editing">Hit enter to save this item</span>
</div>
</div>
<div class="row alert alert-info text-center" v-show="todos.length==0">
  <p class="alert alert-info">
    <strong>All Caught Up</strong>
    <br/>
    You do not have any todo items</p>
  </div>
</div>
</template>

<script>
import axios from 'axios';
import bus from '../bus.js'

export default {
  data() {
    return {
      todos: []
    }
  },
  created: function() { // get todo items and start listening to events
    this.fetchTodo();
    this.listenToEvents();
  },
  methods: {
    fetchTodo() {
      let uri = 'http://localhost:4000/api/all';
      axios.get(uri).then((response) => {
        this.todos = response.data;
      });
    },
    updateTodo(todo) {
      let id = todo._id;
      let uri = 'http://localhost:4000/api/update/' + id;
    }
  }
}

```

```

        todo.editing = false;
        axios.post(uri, todo).then((response) => {
            console.log(response);
        }).catch((error) => {
            console.log(error);
        })
    },
    deleteTodo(id) { //delete todo item
        let uri = 'http://localhost:4000/api/delete/' + id;
        axios.get(uri);
        this.fetchTodo();
    },
    listenToEvents() {
        bus.$on('refreshTodo', ($event) => {
            this.fetchTodo(); // referesh or update todo list on the p
        })
    }
}
}
}
</script>
<style scoped>
    .delete__icon {}
    .todo__done {
        text-decoration: line-through !important
    }
    .no_border_left_right {
        border-left: 0px;
        border-right: 0px;
    }
    .flat_form {
        border-radius: 0px;
    }
    .mrb-10 {
        margin-bottom: 10px;
    }
    .addon-left {
        background-color: none !important;
        border-left: 0px !important;
        cursor: pointer !important;
    }
    .addon-right {
        background-color: none !important;

```

```
border-right: 0px !important;  
}  
</style>
```

The code is pretty straight forward but I will do justice by taking a moment to explain what is going on here.

We created 4 functions in the snippet : `fetchTodo()` makes calls to backend and fetches todo items , `updateTodo(param)` is called when changes are made to todo items. Basically, `updateTodo()` forwards your changes to backend where it can be persisted.

When you click on delete (*X button*), `deleteTodo(param)` is executed and thus remove the item. When a todo item is deleted successfully from the database, it's idea to update our page to reflect the change. Since we have multiple components, the only way to let component B know that todo item A is no longer available is to fire an event (`refreshTodo`) and the listening component B can now request for a fresh list. That said, `fetchTodo()` is executed when ever `refreshTodo` event is fired and our page is updated.

App Component

For some reasons, I have come to love wrapping or putting all my components in a parent component named App.

```
//public/src/components/App.vue
```



```
<template>
  <div id="app">
    <div class="container">
      <div class="row col-md-6 offset-md-3">
        <create-todo></create-todo>
        <list-todo></list-todo>
      </div>
    </div>
  </div>
</template>
```

```
<script>
import CreateTodo from './CreateTodo.vue';
import ListTodo from './ListTodo.vue';
export default {
  name: 'app',
  data() {
    return {}
  },
  components: {CreateTodo, ListTodo},
}
</script>
<style>
.fade-enter-active,
.fade-leave-active {
  transition: opacity .5s
}
.fade-enter,
.fade-leave-active {
  opacity: 0
}
</style>
```

Root Instance

A root instance must be defined for every vue application. We can see a Vue instance or root instance as the root of the tree of

components that make up our app.

We define a root instance by creating a new file `main.js` in **public folder** with this code.

```
//public/main.js
'use strict'
import Vue from 'vue'
import 'bootstrap/dist/css/bootstrap.min.css';
import App from './components/App.vue'

new Vue({
  el: 'app',
  created: function () {
    console.log('root instance was created')
  },
  components: {App},
  methods: {}
})
```

Let's take a moment to go through what is happening in the file we just created.

We imported bootstrap css and also imported `App` from *App Component* and defined as a component on the root instance. Pretty simple, yeah?

We've can this far, we can now build our assets by running:

```
$ npm run build
```

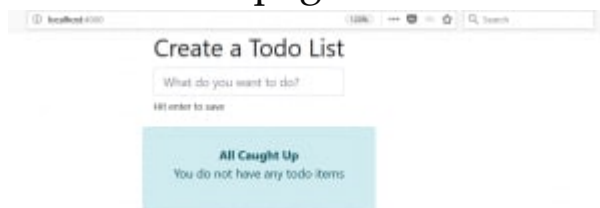
You should now see a new file named `bundle.js` created in `/public/build` folder.

To wrap this up, we update our index page created in part 1 of this tutorial with this code:

```
<!--/public/index.html -->

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>VueJS NodeJS and Express  Todo App</title>
  </head>
  <body>
    <app></app>
    <script src="/build/bundle.js"></script>
  </body>
</html>
```

If you start your server and navigate to <http://localhost:4000>, you will be greeted with a page like this.



This source code of this application can be found [here](https://dev.to/abiodunjames/build-a-todo-app-with-nodejs-expressjs-mongodb-and-vuejs--part-2--3k11)



dev.to is where software developers stay in the loop and avoid career stagnation.

Signing up (for free!) is the first step.



Samuel James + FOLLOW

Gym, code and everything in between.

 samuelabiodunj  abiodunjames  samuelabiodun.com

Add to the discussion




PREVIEW

SUBMIT



pulkitgoel96 

Mar 29 

Hi,

I was following your tutorial line by line, and just before the last step(npm run build), I got an error "No configuration file found and no output filename configured via CLI option. A configuration file could be named 'webpack.config.js' in the current directory."

Please help.



REPLY



Mark Johnson 

Apr 3 

I was able to get this up and running by doing a few things:

1. make sure your public folder file structure matches what is in Samuels github repo here: [github.com/abiodunjames/NodeJs-Tod...](https://github.com/abiodunjames/NodeJs-TodoApp)

Mine was off a little bit based on how I read the instructions. Make sure you check out the subfolders for consistency as well.

1. Make sure you have a webpack.config.js in your root folder that looks like this: github.com/markjohnson303/vue-todo....

2. Don't forget the last step of updating index.html to reference bundle.js

I had to make a couple changes to the webpack.config.js that Samuel has in his repo, but this should get you going.

[REPLY](#)

Samuel James 

Apr 19 

Hi,
Sorry for late my response. I think you are webpack.config.js.
[github.com/abiodunjames/NodeJs-Tod...](https://github.com/abiodunjames/NodeJs-TodoApp)

[REPLY](#)

Raul Guerrero Carrasco 

May 4 

Hi,

Great tutorial. Thanks.

Only a doubt, it isn't finished, right?

[REPLY](#)

Samuel James 

May 5 

Thank you. Yes, it's finished. Part 2 is the last part of the series or do you have any topics you would want me to write on?

[REPLY](#)

[code of conduct - report abuse](#)

Classic DEV Post from May 24

Share Your Best Motivational Quotes



Sai gowtham

Share Your Best Motivational Quotes

[READ POST](#) [SAVE FOR LATER](#)[Another Post You Might Like](#)

What I Learned From Not Planning A Web App (from start to finish)



jen chan

I got an art commission to make a stylized website that included a question-and-answer form sent to a columnist's email. "Simple enough", I thought. I was wrong.

[READ POST](#) [SAVE FOR LATER](#)

Nuxt+Expressのプロジェクト作成で良さそうなのは？

dala00 - Aug 10



Using Strapi for Node.js Content Management with a React SPA

Charles Ouellet - Aug 9



Parsing JSON with Node.js

flavio ⚡🐛 - Aug 8



You don't need express to get started with socket.io

Sadick - Aug 8

[Home](#) [About](#) [Sustaining Membership](#) [Privacy Policy](#) [Terms of Use](#) [Contact](#)

[Code of Conduct](#) DEV Community copyright 2018 