The Life-Changing Magic of Tidying Up: MVC Architecture in JavaScript Applications

Ashleigh Halverstadt (heretoshleigh)

August 2022

The art of decluttering and organizing your code

- MVC stands for Model, View, Controller
- It's an approach to organizing back-end code
- Different parts of code do different things. MVC groups code into three categories based on what the code does.
- It's commonly used in JavaScript but can be used in lots of other languages too

There's a place for everything

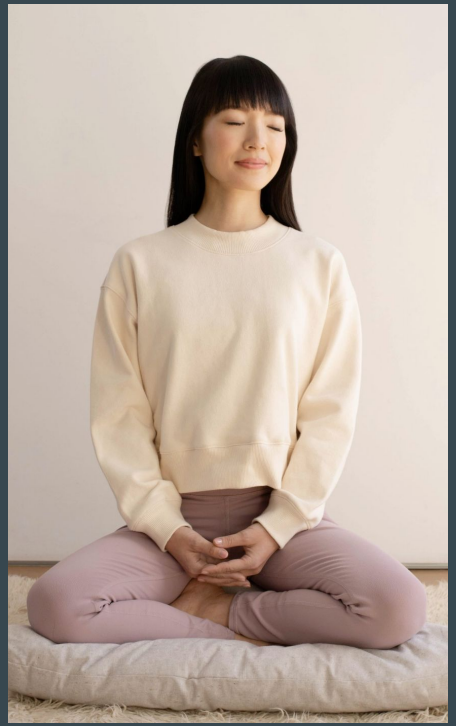
MVC organizes code according to purpose (separation of concerns!):

- Model - defines and manages an app's data
- View - determines what users see, and how it looks
- Controller - liaises between the user (or Route*), Model, and View, receiving requests and figuring out how to handle them

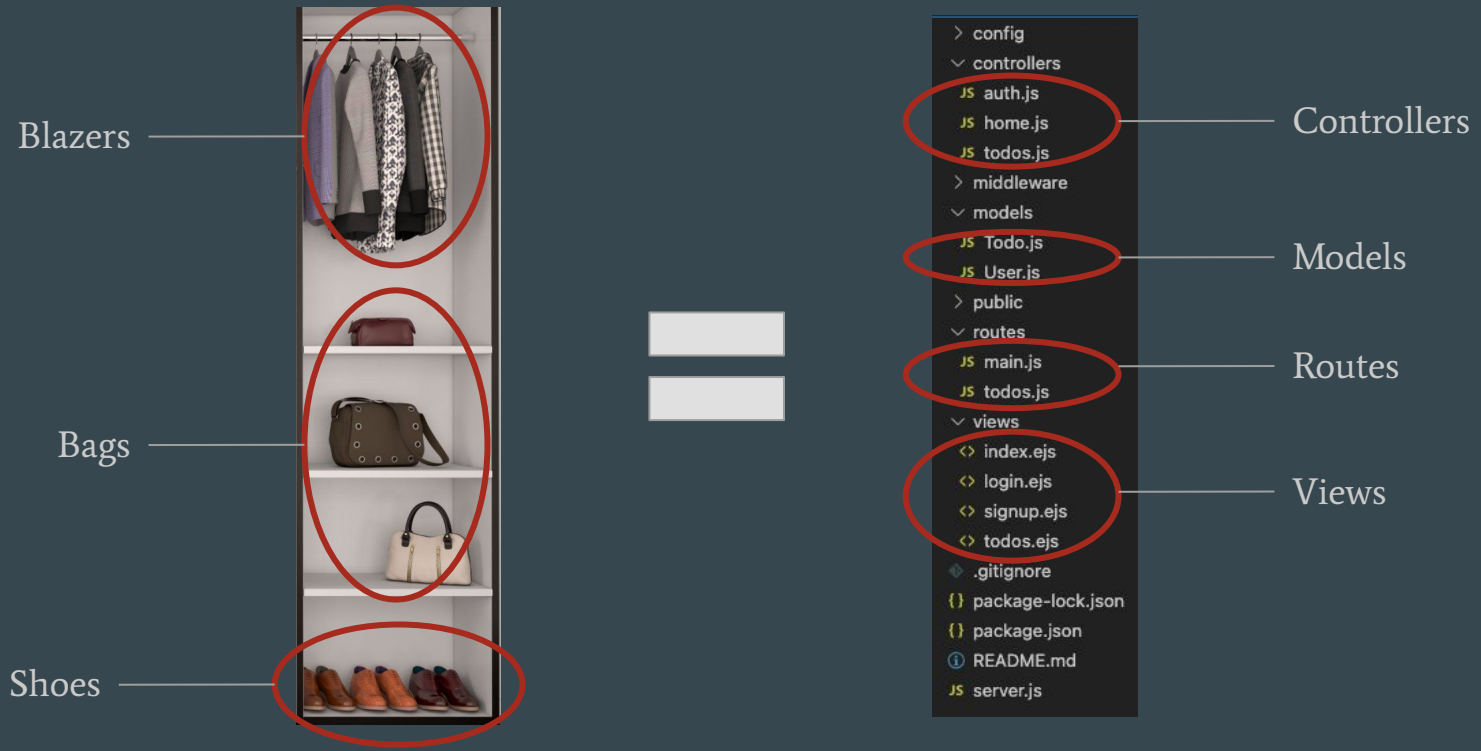
*A Route is commonly used as another layer of abstraction between the user and the Controller. It hears requests and sends them to the appropriate Controller.

Yeah, but will it really “transform my life”?

- Organizing code by purpose makes it so much easier to maintain, update, collaborate on, and reuse!
- MVC code is abstracted and modular. When you need to fix a bug, make a change, or swap out a component entirely, you can go straight to the relevant code without sifting through or worrying about the rest.
- If you’re working on a team, it lets different developers work on different parts of the app simultaneously.
- It’s the difference between throwing your clothes into one giant heap vs. folding them neatly in drawers by category - and then asking your coworker to find you a pair of matching socks.

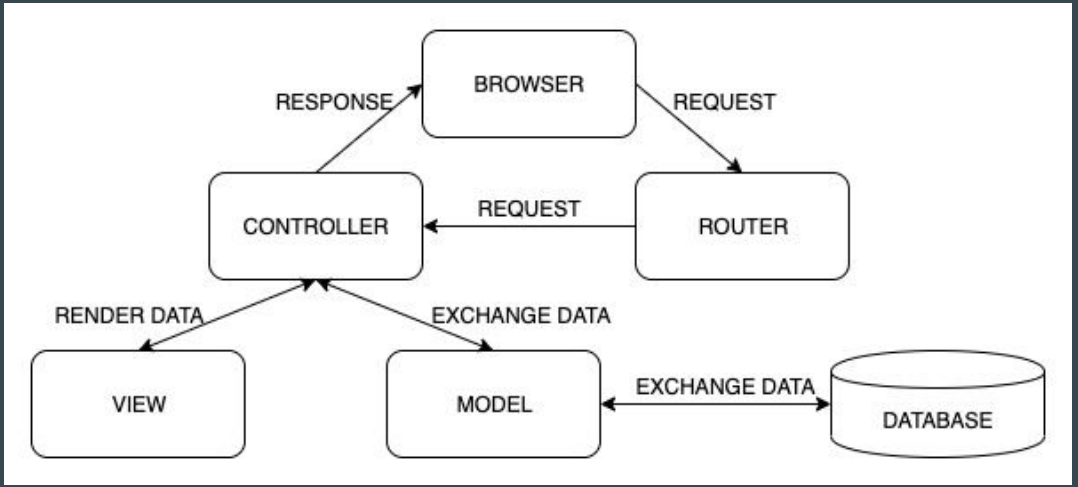


Don’t put your code in a server.js heap!



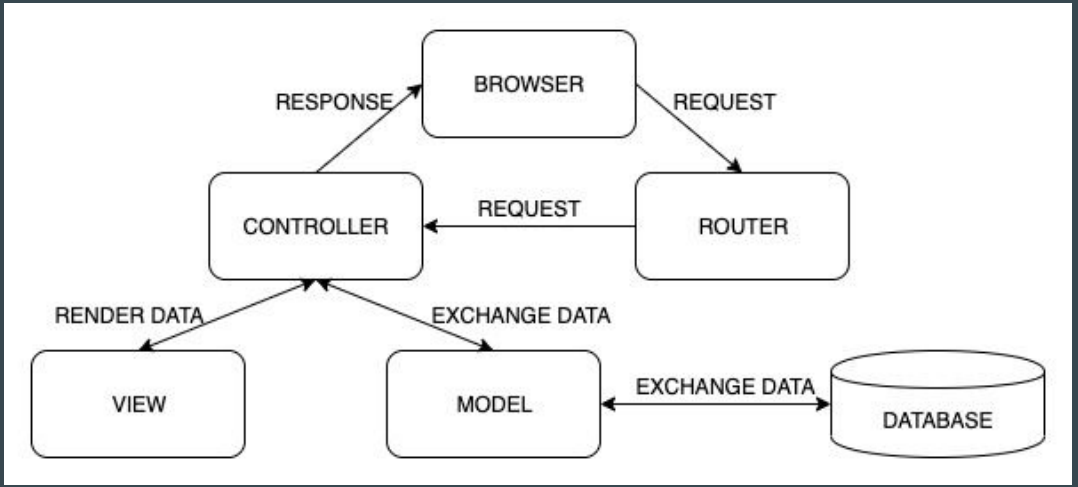
How it works: step by step

1. The user sends a request via their browser.
2. The appropriate Route hears the request and passes it to the appropriate Controller (don't stress, but yes, there are often multiple Routes, Controllers, Models, and Views).
3. The Controller decides what to do next.



How it works: step by step

4. If data needs to be fetched, changed, or removed, the Controller tells the appropriate Model, which accesses the data storage system (e.g., MongoDB) and does what was asked.
5. AND/OR, the Controller tells the appropriate View what to render, and the View does it using a templating language (e.g., EJS).
6. The Controller sends a response to the browser, and the user sees what they requested.



A word on using Models with Mongoose and MongoDB

Mongoose is a library that makes MongoDB easier to use. In Mongoose:

- Schemas are used to create a structure for each MongoDB collection and to validate incoming data (i.e., they define the properties each document in a collection will have, as well as the features of those properties, such as data type, whether the data is required, etc.)
- Schemas are compiled into Models, which can then communicate with Controllers and with MongoDB

```
1  const mongoose = require('mongoose')
2
3  const TodoSchema = new mongoose.Schema({
4    todo: {
5      type: String,
6      required: true,
7    },
8    completed: {
9      type: Boolean,
10     required: true,
11   },
12   userId: {
13     type: String,
14     required: true
15   }
16 })
17
18 module.exports = mongoose.model('Todo', TodoSchema)
```

Sample Mongoose Model & Schema

“[MVC] is not a mere set of rules on how to sort, organize, and put [code] away. It is a guide to acquiring the right mindset for creating order and becoming a tidy [developer].”

-Marie Kondo?