The Model-View-Controller patterns

Relevant Links

- Addy Osmani's Patterns book, MVC section[1]
- Backbone's Models[2]
- Backbone's Views/Controllers[3]
- Addy Osmani's Backbone book[4]

Notes

MVC in general

The MVC pattern is an important Architectural pattern for designing applications that interact with a user. It designates 3 components that have different responsibilities but work together to produce the desired effect. The three components are:

Model The model is responsible for managing the data of your application. It maintains a "state" and accepts observers to that state.

The model knows how to save and load the data, how to validate the data for consistency and so on. It contains all of the application's "business logic".

View The view is responsible for showing information to the user, and creating the user interface. It reads this information from the model.

In Javascript, views are typically just HTML code, often implemented via templates, and most of their logic rolls into the controller.

Controller The controller is responsible for coordinating the interaction between the user and the application. It registers itself to listen for updates of the model, as well as user-driven events from the view.

- The controller responds to user events communicated via the view by triggering model updates.
- The controller responds to model updates by instructing the view to update what it shows.

In general in a complex application we would have many different models, views and controllers, loosely coupled to each other.

The crucial part of this structure is that models should be completely decoupled from their visual representations. You should be able to test that your model works properly without having to worry about interacting with the DOM.

[1] http://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailmvcmvp
[2] http://backbonejs.org/#Model
[3] http://backbonejs.org/#View
[4] http://addyosmani.github.io/backbone-fundamentals/

MVC for our TODOapp

Let us consider our TODOapp application, and review its main parts. Browse through the GitHub page[5] for details, specifically the files app/todoList[6], app/todoListController[7] and app/todoStorage[8].

Models

- TodoTask: A single task item
- TodoList[9]: Maintains a list of all task items

Let us start with these two before we discuss more. There is already a decision to be made: Should TodoTask be a "model" itself, or should necessary functionality be baked into TodoList? There are tradeoffs.

The key question is whether we want observers to be able to observe direct changes to items, or whether the only observer who needs to know that is the one already observing the whole list. **Models are observable**.

In our example we opted to have only a TodoList "collection", and baked all functionality in there.

The model is responsible for maintaining a list of todo tasks, **but** it is not responsible for how those tasks are actually stored (on a server or whatnot). It delegates that responsibility to a TodoStorage class. This allows us to easily swap the TodoStorage class in and out and freely swap between local storage, server storage etc.

Models want to always mix in an **Event** module to make themselves "observable". They will therefore always have "on", "off" and "trigger" methods. They trigger events when they change, and they can decide how fine-grained that triggering process has to be. For instance they can fire a single "change" event and nothing else, or they can fire individual events for "add", "remove", "update" etc.

Controllers

- TodoListController[10]: Responsible for creating the view for the list and establishing a relation between the list and its visual representation.

The Controller is initialized given a UI element and a todoList to manage. It then registers itself to be notified whenever the todoList changes, as well as when the user interacts with the page. It uses a "render" method and a template to get its work done.

[5] https://github.com/skiadas/WebAppsTodo
[6] https://github.com/skiadas/WebAppsTodo/blob/master/app/todoList.js
[7] https://github.com/skiadas/WebAppsTodo/blob/master/app/todoListController.js
[8] https://github.com/skiadas/WebAppsTodo/blob/master/app/todoStorage.js
[9] https://github.com/skiadas/WebAppsTodo/blob/master/app/todoList.js
[10] https://github.com/skiadas/WebAppsTodo/blob/master/app/todoListController.js