Dynamic View Rendering Using ReactJS and jQuery

Article · August 2016

CITATIONS
0

READS
380

1 author:

Wildan S. Nahar
Bandung Institute of Technology
10 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Dynamic View Rendering Using ReactJS and jQuery

Wildan Syahrun Nahar*

*) wildansnahar@gmail.com

A project based on my internship as frontend engineer at YesBoss Indonesia

ABSTRACT
Rendering in html is one of the most common problems in web development world. Especially when the data that rendered get bigger and more complex. React JS, a view in MVC schema that developed by Facebook is one of the most powerful front-end framework that can deals with large data with fast dynamic rendering. In this project, I create a feature that embedded in Google Chrome Extension for Customer Service to use in dashboard. Helped by jQuery, AJAX, and some javascript stuffs, this feature helps a lot for customer service to provide dynamics data with fast rendering.

Keywords
Render, dynamic data, state, view.

1. INTRODUCTION
Web development nowadays is one of the most common technology to provide robust and fastest service to user. Javascript is the largest web language to use in web development, especially in startup environment which dynamic content and service happens daily. In this article, I'll cover my project in my internship period at PT YesBoss Indonesia.

In this project, I use technologies like jQuery, AJAX, ReactJS and Restful API to make features on Google Chrome Extension. The features are aim to fetch data from other server filled with cities, cinemas, and movies and display it in dashboard for customer service.

jQuery
jQuery is a fast, small, and feature-rich Javascript library. It makes things like HTML document traversal and manipulation, event handling, animation, and AJAX much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write Javascript.[1]

AJAX
Asynchronous Javascript and XMLHTTP, is a technique web programming to create interactive web application. [2]

ReactJS
React is a javascript library for building user interfaces. These are 3 fundamental features of React :

1. Declarative
React makes it painless to create interactive Uis. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug.

2. Component-based
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in Javascript instead of templates, you can easily pass rich data through your app and keep state of the DOM.

3. Learn Once, Write Anywhere
We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.[3]

Restful API
Representational state transfer (REST) is an architectural style used for web development. Systems and sites designed using this style aim for fast performance, reliability and the ability to scale (to grow and easily support extra users). To achieve these goals, developers work with reusable components that can be managed and updated without affecting the system as a whole while it is running.

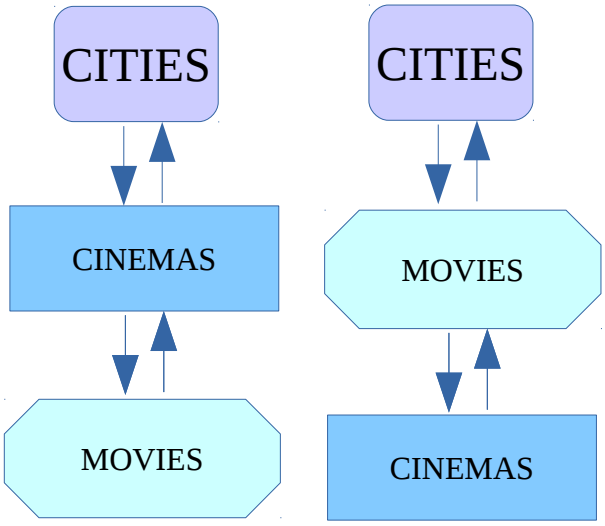
2. PROBLEMS

Problems that occur in daily use apps is that we could not fetch data simultaniously. With newest technology, especially using React and AJAX, we could do this.

3. BUSINESS LOGIC

Our dashboard has to be featured with list of all recommendation movie card. It is a feature that enable our CS to provide list of movies with its schedules based on city and cinema selected. Vice versa, it has to be able to display all cinemas in certain city that show the selected movie. So, it's about handling data of cities, cinemas, and movies so all of the data can beautifully display to user.

This is the schema looks like :



So, the story is when the CS select a city from list of cities, it will get all the cinemas that available on the selected city. When fetching data is successful, it will load all the cinemas in dropdown menu. Then, when selected cinema is choosen, it will get all the movies and each schedules and display it in a list with checkbox for each movie. All the data is in JSON mode. JSON makes it so easy to do manipulate, change, and restructure to what we want.

Here are the sample data of cinemas :

```
data = [
    "Bogor": [
        {
            "movie_name": "IRON MAN",
            "movie_code": "IRM",
            "cinema":"Botani XXI",
            "schedules":"12:25"
        },
        {
            "movie_name": "IRON MAN",
            "movie_code": "IRM",
            "cinema":"Botani XXI",
            "schedules":"14:50"
        },

    ],
    "Jakarta": [
        {
            "movie_name": "WARCRAFT",
            "movie_code": "WAR",
            "cinema":"Blok M Plaza XXI",
            "schedules":"12:30"
        },
        {
            "movie_name": "WARCRAFT",
            "movie_code": "WAR",
            "cinema":"Blok M Plaza XXI",
            "schedules":"15:30"
        }
    ]
]
```

Those data are JSON. When dealing with those particular data, we can dynamically change, restore, and take any part we need. So, basically it's pretty simple to display the data we want.

4. TECHNICAL EXECUTION

a. Story

Customer Service want the feature that list movies with eac schedules so it can be delivered to user. CS also can decrease the time to look forward the movie schedules based on selected city and cinema.

Vice versa, CS also can recommend cinemas in selected city which is displaying certain movie. So, when city is selected, then at the next dropdown menu with list of all movies which is shown in that city.

b. Workflow

1. Structure the project app
2. Define React component
3. Create some function to handle services (JS, AJAX, React, an other tools needed)
4. Data manipulation
5. Pass variable to other files
6. AB testing
7. Deploy
8. Use it in production

c. Code

Here are some codes that I've been working on. Note that not all the codes are shown completely because its copyright is on my former company. So, I only show the codes that in my opinion is representing this feature.

order_card.js

```
handleSearchMovie(callback = () => {}) {
    if (this.state.selectedTheater.length == 0) {
        alert("Please select a theater");
        return false;
    }

    jQuery.ajax({
        url: "https://xx.yy.com/api-management/movie-tickets/getMovieSchedules?theater_id="
        + this.state.selectedTheater,
        method: "get",
        dataType: 'json',
        beforeSend: function(xhr) {
            xhr.setRequestHeader("API-AUTH-TOKEN", "xxx");
        },
        success: (data) => {
            this.setState({
                movies: data
            });
            callback(false);
        },
        error: () => {
            alert('error');
            callback(true);
        }
    });
}
```

Function to get list of movie schedules using jQuery AJAX

```
var pageItems = Math.ceil(this.state.cinemaChecks.length / 5);
for (var i = 1; i <= pageItems; i++) {
    var arrItems = [];
    var start = (i - 1) * 5;
    var end = i * 5;
    if (this.state.cinemaChecks.length < end) end = this.state.cinemaChecks.length;
    for (var j = start; j < end; j++) {
        var objCinemaChecks = JSON.parse(this.state.cinemaChecks[j]);
        function titleCase(strs) {
            strs = strs.toLowerCase().split(' ');
            for (var i = 0; i < strs.length; i++) {
            strs[i] = strs[i].charAt(0).toUpperCase() + strs[i].slice(1);
        }
          return strs.join(' ');
        }
  var newMovies = titleCase(objCinemaChecks.movie_name)
  var str = objCinemaChecks.theater_name;
  function low(str){
    return str.toLowerCase().split(' ').map(function(word) {
      return word.replace(word[0], word[0].toUpperCase());
    }).join(' ');
  }
  var lowStr = low(str);
  var n = "Xxi";
  var baru = lowStr.includes(n);

  if(baru == true){
    var jenis = n.toUpperCase();
    var cinema = lowStr + " " + jenis;
    var newCinema = lowStr.replace("Xxi", jenis);
  }
  else {
    var newCinema = lowStr;
  }
```

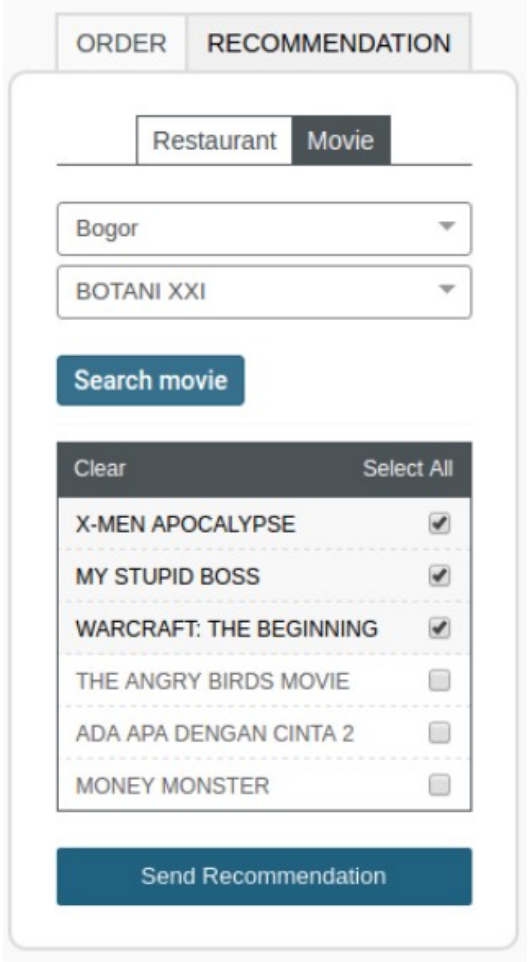
Data manipulation

```
order_card
├── components
│   ├── order_add_label.js
│   ├── order_add_partner.js
│   ├── order_card.js
│   ├── order_detail_item.js
│   ├── order_detail.js
│   ├── order_detail_partner.js
│   ├── order_form.js
│   ├── order_item.js
│   ├── order_list.js
│   ├── order_navigation.js
│   ├── order_nav_tab.js
│   ├── recommendation_flight.js
│   ├── recommendation_flight_search.js
│   ├── recommendation.js
│   ├── recommendation_movie.js
│   ├── recommendation_movie_lists.js
│   ├── recommendation_movie_results.js
│   ├── recommendation_movie_search.js
│   ├── recommendation_restaurant_blogs.js
│   ├── recommendation_restaurant.js
│   ├── recommendation_restaurant_lists.js
│   ├── recommendation_restaurant_reservation.js
│   ├── recommendation_restaurant_results.js
│   ├── recommendation_restaurant_search.js
│   └── recommendation_tabs.js
├── main.js
├── style.css
└── utils.js
```

Project structure files

d. Result

Here are the results, in dashboard for CS and in the app for users.



Display in chrome extension

Display in apps

5. CONCLUSION

So, the feature that I've worked is now in production for everyday use both CS and users. It has many ways to develop, from backend to frontend, from data manipulation till API. Hopefully in the future there will be many of them that will help people to get their needs.

6. FURTHER READING

For further reading and learning, you should read about React Native and GraphQL for better performance.

7. ACKNOWLEDGMENTS

Thanks to my former team in YesBoss Engineering.

8. REFERENCES

[1] http://jquery.com
[2] http://w3schools.com/ajax
[3] https://facebook.github.io/react/