



Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Designing and Implementing an Educational Support System

BACHELOR'S THESIS

Author

Nóra Szepes

Advisors

Dr. Sándor Gajdos
Bence Golda

2015

Contents

Kivonat	i
Abstract	ii
1 Introduction	1
2 Comparing JavaScript Frameworks	2
2.1 React	2
2.2 AngularJS	2
2.3 Mithril	3
2.4 Conclusion	3
3 specification	4
4 conceptual	5
5 design	6
6 implementation	7
7 test	8
8 deployment	9
9 conclusion	10
Acknowledgements	iii
List of Figures	iv
List of Tables	v
Bibliography	vi
Appendix	vii

HALLGATÓI NYILATKOZAT

Alulírott *Szepes Nóra*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2015. október 12.

Szepes Nóra
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon \LaTeX alapú, a *TeXLive* \TeX -implementációval és a PDF- \LaTeX fordítóval működőképes.

Abstract

This document is a L^AT_EX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* T_EX implementation, and it requires the PDF-L^AT_EX compiler.

Chapter 1

Introduction

A bevezető tartalmazza a diplomaterv-kiírás elemzését, történelmi előzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását.

A bevezető szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

Chapter 2

Comparing JavaScript Frameworks

I've worked in vanilla JavaScript and jQuery before but for this project I wanted to choose a JavaScript framework for faster development. I was looking for something fast to learn and easy to use with good documentation and tutorials. With the help of TodoMVC [11] I tried some frameworks to see how fast I can create a simple website with a basic AJAX request, read the documentation to find out how routing works and how can I access a server with AJAX requests.

2.1 React

My first choice was React [9]. It is developed by Facebook and Instagram since 2013. With the help of the official website tutorials and videos I've managed to create websites easily with JSX [4] because I can use XML-like syntax for creating views.

After reading the documentation I found out that it gave me no other option than inline jQuery for AJAX requests and routing is not part of the main core although there are many good routing libraries, e.g. React/Router [10].

2.2 AngularJS

AngularJS [1] is one of the most famous JavaScript frameworks nowadays. It is maintained by Google for 6 years now. It focuses mostly on dynamic views in web-applications. On the homepage they provide many examples in written tutorials and videos on YouTube.

Creating a website is done with an extended HTML vocabulary, like Android Layouts where you declare everything in XML. Building a website wasn't that hard but the data binding is different. It uses a two-way data binding template [3] which means whenever either the View or the Model is changed, it will update the other one.

Angular AJAX requests with the shortcut methods are similar to the AJAX methods in jQuery. It can also be used in unit tests with ngMock [2].

In a single-page application we want to make the browser think it is always on the same page. When the user clicks on a new link, the browser won't reload the whole page, it will just simply load the new view into the old frame. Angular binds a special listener to links. If the user clicks on the link, Angular will simply push the page to the history and replace the view with the new page. This method only works if Angular is loaded into memory otherwise the application needs a page where Angular can be loaded.

2.3 Mithril

My last choice was Mithril [5]. It is really similar to React. With the help of MSX [8], you can use the same HTML-like syntax to create websites. MSX uses JSX, but transforms the output to be compatible with Mithril.

It provides a simple way to work with AJAX requests [6]. A basic request returns a special *m.prop* getter-setter, what is a utility class.

Mithril provides utilites to handle routing [7]. It's organized around URL routes. For a URL we can create a new module with a controller and a view. When the user clicks on the link a new instance will be made and passed to the view.

2.4 Conclusion

React is basically only for building views and I should use it combined with other frameworks to have full MVC experience. I really liked the fast way of creating UIs but I was looking for a framework that supports everything what I need.

Although Angular contains everything what I need, I didn't like the extended HTML vocabulary. I liked the fast and easy way of creating websites in React. Mithril provides a similar way of doing it.

In this project I'll use Mithril. It is a simpler framework than Angular, because it doesn't implement features what make it feel like a new programming language and not a JavaScript framework.

Chapter 3

specification

Chapter 4

conceptual

Chapter 5

design

Chapter 6

implementation

Chapter 7

test

Chapter 8

deployment

Chapter 9

conclusion

Acknowledgements

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

List of Figures

List of Tables

Bibliography

- [1] AngularJS. <https://www.angularjs.org/>.
- [2] AngularJS, API Reference, http. [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http).
- [3] AngularJS, Developer Guide, Data Binding. <https://docs.angularjs.org/guide/databinding>.
- [4] JSX Specification. <https://facebook.github.io/jsx/>.
- [5] Mithril. <http://mithril.js.org/>.
- [6] Mithril, m.request. <http://mithril.js.org/mithril.request.html>.
- [7] Mithril, routing. <http://mithril.js.org/routing.html>.
- [8] MSX. <https://github.com/insin/msx>.
- [9] React. <https://facebook.github.io/react/>.
- [10] React/router. <https://github.com/rackt/react-route>.
- [11] TodoMVC. <http://todomvc.com/>.

Appendix