VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Tutorial Builder

Preliminary Report

Zoltan Debre

Supervisor: Dr Stuart Marshall

July 8, 2016

Submitted in partial fulfilment of the requirements for
Master of Computer Science.

Abstract

Presenting a computer programming problem or solution with a high-quality video is time-consuming and it is not easy to update.

Technical content producers are struggling to find an efficient and interactive way to show their work.

This research introduces a tool that can be used for creating an interactive tutorial.

Furthermore this tool shows programming code snippets in web based code editor so the content creator can build a playable step by step ”movie” with it.

Contents

Chapter 1

Introduction

This project is about building an online publishing tool prototype, using code editors and step by step instructions to present programming challenges and solutions for a computer science problem.

The prototype has two parts, an administration area, where the content creator can build a tutorial, and a player tool, where the recorded steps would be presented. This player should be embeddable in any blog post or website.

The primary target user is the creator, who composes a new tutorial. The creator can be a teacher, or an open source project owner, who would like to introduce their tool or code.

The secondary user is the consumer, who wants to learn or know more about a problem or a coding solution.

1.1 Motivation

We all have the unstoppable desire to learn. We are keen to know more about the world around us, about our hobby and our profession. In software development, in computer science, the knowledge is essential, it is the key to succeeding. Reading, studying, sharing. An infinite loop of collecting and adapting new practices.

In information technology, especially in programming languages, writing blog posts, creating static, step by step tutorials are a popular way to share or learn something new. Producing and sharing the content is easier nowadays, but still requires more effort from the creator, when they want to deliver an easy to understand high-quality tutorials.

Creating interactive tutorials are appealing, but the production cost is much higher. Recording a video tutorial or especially updating it is time-consuming, and it involves more effort from the creator.

I think an ideal solution would be a healthy mix of static and dynamic contents, where the learners can read instructions but meanwhile they can watch the steps in a code editor, in a more realistic environment.

Chapter 2

The problem and the solution

2.1 The problem

When a developer, teacher or hobbyist would like to present a computer programming problem or solution, most of the times they record a video and publish it on YouTube. Sometimes the easiest way to record a video of a conference/meetup presentation or a live coding session. However recording and editing a high-quality video is time-consuming and less flexible. It is also hard to update.

Other problem with showing tutorials with a simple video, that the audience cannot give it a try, they have to configure their computer and environment to play with the presented solution.

Most of the cases we would like to show instructions and code snippets, mainly text-based contents. It is preferred to show code snippets in a more realistic environment, such as in a code editor. Therefore using the online code editor with a pre-scripted way to play the presentation, where the user can navigate back and forth and can modify or play with the code is more interactive. It involves everyone and helps to understand a problem clearly.

Additionally, it is much easier to maintain, upgrade or fix for the content creator.

Furthermore, the user is able to experience it and can see the result, so the new information can be put into practice immediately.

2.2 Personas and their goals

2.2.1 Primary persona

Content creator, open source project maintainer, teacher.

Their motivation is to present a technical problem and the solution in a clear and an easy-to-understand way. The best option is to show a demo, what happens when we insert the suggested code, how easy it is to use. Most of the times a presentation involves more steps. For example, we would like to show a starting state, maybe a few lines of code which we are able to simplify, so in this case, the first step is to show the problem and after we show step by step how we solve that.

2.2.2 Secondary persona

The consumer, who watch the presentation, who reads the tutorial and who wants to learn more about the actual problem.

They would like to play, stop, and control the presentation. Control the flow, going forward or stepping back.

They would like to try the solution, for example how the final state changes when they modify the code.

2.3 The solution

I develop a prototype web app, where the content producer can create a simple step by step tutorial, and the content consumer can ”watch” this tutorial and can interact with it.

There are two different users:

- content producer, for example a teacher
- content consumer, for instance a student

2.4 Requirements

Requirements are separated in three different groups.

Requirements from the teacher perspective:

- Teacher can navigate to Admin page.
- Teacher can create a new tutorial.
- Teacher can add steps to the tutorial.
- A step could be four different type:
  - Instruction type is a text content.
  - Html type, which adds content to the html editor box.
  - Css type, which adds content to the css editor box.
  - JavaScript type, which adds content to the javascript editor box.
- Teacher can modify the sort of the steps with reorganizing with drag and drop.

Requirements from the student perspective:

- Student can see a list of tutorials.
- Student can click on a tutorial and can see the steps.
- Steps are presented in order.
- Student can ”play” and ”watch” the steps.
- Student can ”pause” and step ”backward”.

User interface requirements:

- The tutorial screen has five area:
- Instruction area.
- Html code editor textarea.
- Css code editor textarea.
- Javascript code editor textarea.
- Html preview textarea.

The main website has two main section:

- Admin page where Teacher can edit tutorials.
- Tutorials page where Students can select and watch tutorials.

Chapter 3

Background and Related Work

During the development and research work, I found a few related interesting projects. These are similar or I can use them partly.

3.1 CodeMirror Movie

I found this project when I checked the most popular web-based code editor tool, CodeMirror website. The creator of the CodeMirror wanted to present their tool with a realistic way, so CodeMirror Movie was born. [2]

This solution highly coupled with CodeMirror, it is similar to an add-on, so it is possible to attach any CodeMirror implementation. (More about CodeMirror in the next section.)

Adding CodeMirror Movie to our project is straightforward because the open source repository provides a CSS and a JS file, so they can be added to any page.

This tool mainly targets web-developers, so with the help of this tool they can add code and scripts to their websites.

Editing ”the movie” script is manual. There is a simple syntax which control the presentation steps and this script should be added to the textarea which will be in the code editor.

We clearly see that it is a very effective way to build a presentation, however it requires real development skills.

Pros:

- simple, lightweight implementation
- easy to add your project if you use CodeMirror and you are a developer
- simple script language to manage the presentation
- user can use the code editor to try the presented solution

Cons:

- mainly for developers only
- highly coupled with CodeMirror

3.2 Comparison of online code editors

There are 3 popular web based code editors: CodeMirror, Ace Editor and Monaco.

CodeMirror and Ace Editor are commonly used on websites and different projects. Monaco is a new solution from Microsoft and it is extracted from their popular Microsoft Visual Studio Code developer tool.

There is not significant differences between them. All has the most important code editor features, like supporting more than 100 languages, autocompletion, syntax highlighting, controlling with shortcuts.

I choose CodeMirror, because it has already Ember.js support. Thanks for the ivy-codemirror Ember addon, it can be added to any Ember.js project with the installation of the addon.

CodeMirror

- Github link: `https://github.com/codemirror/CodeMirror`
- Website: `http://codemirror.net/`
- Popularity (GitHub Star): 9396
- Ember.js Addon: `https://www.emberobserver.com/addons/ivy-codemirror`

Ace Editor

- Github link: `https://github.com/ajaxorg/ace`
- Website: `https://ace.c9.io`
- Popularity (GitHub Star): 12950
- Ember.js Addon: none

Monaco

- Github link: `https://github.com/Microsoft/monaco-editor`
- Website: `https://microsoft.github.io/monaco-editor/`
- Popularity (GitHub Star): 2322
- Ember.js Addon: none

3.3 Reviewing code sharing websites

We can use code editor and sharing platforms also when we want to demo a small feature or describe a problem. These websites are combinations of code editors and an iframe where we can see the preview of the code snippets.

One of the common features is splitting the screen and providing different windows for editing html, css and javascript separately.

User can save the edited content also. Most of them can be embed in a blog post or in other website.

Most important findings:

- All use Code Mirror as code editor
- All of them separate the css, html and javascript editing in different screens, but they merge into one file, and preview of this merged html file is possible in an iframe.
- Saving the different type of code (css, javascript, html) separately.

	Code Pen	JSBin	JSFiddle	Ember Twiddle
Link to open source project	not open source	[a]	not open source	[b]
Website	[c]	[d]	[e]	[f]
Code Editor	Code Mirror	Code Mirror	Code Mirror	Code Mirror
Support embedding	yes	yes	yes	yes

Links:

- a: `https://github.com/jsbin/jsbin`
- b: `https://github.com/ember-cli/ember-twiddle`
- c: `http://codepen.io`
- d: `http://jsbin.com`
- e: `https://jsfiddle.net/`
- f: `https://ember-twiddle.com/`

Chapter 4

Completed Work

4.1 The technology of choice

One of the most popular programming languages in web development is JavaScript. The usage of this frontend focused technology is growing quickly. It is the 7th on Tiobe Index, which is a good indicator of programming languages popularity. [1]

Learning and teaching JavaScript, HTML and CSS is important. My tool focuses on this three main building blocks of the web.

Building a frontend heavy application, with a dynamic, user-friendly interface is more common nowadays. In the last few years JavaScript based frontend frameworks became mature, production ready tools. Server side technologies, like database management and time and resource heavy processes are separated from the user focused, design driven view layer, which is developed with usage of frontend frameworks.

The most popular tools are Angular.js, React.js and Ember.js. In my project I use Ember.js. It is an "opinionated" framework. Opinionated, convention over configuration driven framework means that developers should follow specific conventions, instead of using a tool freely. A more strict environment helps to adopt best practices and speed up the development process.

Certainly, we still have to store data and information, so we cannot live without backend and server technology. Luckily there are already cloud-based tools for managing databases. I use Firebase, which is a service provided by Google. Firebase is a cloud-based database, document-store solution and easy to integrate with Ember.js.

Additionally, I build a traditional backend server application also to support development and experimenting with a real server side and a cloud-based solutions parallel. My preferred technology on backend side is Ruby on Rails, a popular, also opinionated and convention over configuration driven backend framework.

4.2 The development environment

Following the most modern standard of web applications, I separate the user faced frontend development and the data store, backend development.

The user face frontend application uses Ember.js frontend framework. The primer data store is Firebase cloud-based database, however I already created a Ruby on Rails application which is running when the application is in development mode.

Ember.js development requires Node.js on the development machine to run the development environment. This development environment helps to run and modify frontend code quickly, and it generates the final, deployable production code also. The production version

of the application is only a static website. It means, there is one index.html, two JavaScript files and two CSS files.

4.3 Source control management

I have been using GitHub for managing source code and tracking code changes. Link to repository: `https://github.com/zoltan-nz/tutorial-builder`

4.4 Frontend features

The look and feel of the application will follow the standard Bootstrap style. Bootstrap is added to the project. I use SaSS version of the Bootstrap, so I can customize it with the modification of the SaSS variables. SaSS is a modern CSS development environment, helps to programmatically modify the CSS.

The home page of the application is only a placeholder. I added a navigation bar with the following links: Home, Sandbox, Tutorials, Admin.

I implemented a breadcrumb bar also, which helps in navigation.

First, I created a Sandbox area, where I experiment with the CodeMirror code editor and an iFrame, which shows the preview. This Sandbox page contains the editor. When the source code is updated, the preview page automatically shows the generated website. This feature uses Ember.js default two-way bindings capability.

Database management is already implemented. The main adapter is the Firebase adapter, which automatically update data to the Firebase server. Firebase is a real time database. The limited, free to use version is enough for experimenting and for demo.

The secondary adapter is a JSONApi Adapter. JSON Api [3] is a new standard of data communication format. This is the Ember.js default adapter.

4.5 Static website hosting

I have been using surge.sh [4] for static website hosting, so the actual state of the prototype is updated there regularly. Link to the live version: `http://tutorial-builder.surge.sh`

4.6 Implemented requirements

Previously listed requirements, the followings are already implemented.

Requirements from the teacher perspective:

- ✓ Teacher can navigate to Admin page.
- ✓ Teacher can create a new tutorial.
- ✓ Teacher can add steps to the tutorial.

Requirements from the student perspective:

- ✓ Student can see a list of tutorials.

User interface requirements:

- ✓ Html code editor textarea.
- ✓ Html preview textarea.

The main website has two main section:

- ✓ Admin page where Teacher can edit tutorials.
- ✓ Tutorials page where Students can select and watch tutorials.

4.7 Future work

In the following four months I will focus on the implementation of the rest of the features. Additionally, it is important to validate the prototype, so I would like to use the tutorial builder on a public website and collecting feedback from test users.

Requirements from the teacher perspective:

- A step can be four different types:
  - Instruction type is a text content.
  - Html type, which adds content to the html editor box.
  - Css type, which adds content to the css editor box.
  - JavaScript type, which adds content to the javascript editor box.
- Teacher can modify the sort of the steps with reorganizing with drag and drop.

Requirements from the student perspective:

- Student can click on a tutorial and can see the steps.
- Steps are presented in order.
- Student can ”play” and ”watch” the steps.
- Student can ”pause” and step ”backward”.

User interface requirements:

- The tutorial screen has five area:
- Instruction area
- Css code editor textarea.
- Javascript code editor textarea.

Bibliography

[1] Tiobe.com - http://www.tiobe.com/tiobe_index

[2] CodeMirror Movie - https://github.com/sergeche/codemirror-movie

[3] JSON Api - http://jsonapi.org

[4] Surge.sh Static Website Hosting Service - http://surge.sh