Valuelink Course

Guide to Advanced Use Cases and Debugging

Course Overview

Note: Parts of this section are taken from Assignment 4. In addition this course proposal borrowed the outline from ("Piazza @360," 2017)

Logistics

This will be an individual project done by myself. I've gotten permission from work to reference (with modification) code currently used at work for the course. In addition I was encouraged to integrate the course once finished with our training program. The training program uses an in house platform and will not be considered for the scope of this class as it will be far in the future. Lastly some of the content may require referencing my coworkers for examples.

Content

The course will go over practical examples and best practices in depth for a node.js library called valuelink[1]. In particular, a focus will be on advanced use cases, debugging, and common mistakes. Additional commented sample code will be provided. If time permits, introductory material and theoretical underpinnings will be explored as well.

Format

The initial format of the course will be sample code, text, video, and project based. Based on feedback and time considerations the course style may change. The long term goal is to make documentation similar to Discover Meteor (Coleman & Sacha, 2015) or the PrettySwift (Johnson, 2015) project which does not require registration or going through a

[1] The author of the library has rebranded it as NestedLink as part of his NestedReact framework, on npm it is still installed via valuelink so I will continue to refer it as such.

class portal. Video will be used to demonstrate things like how to debug as it is the easiest method to do so. Text with source code commentary will be used to introduce other material.

Software

For making the course class I will use openEdX for organization and development purposes. I will also use the platform to hold feedback surveys. I will play around with (“Oppia,” 2015) for some of the modules. Oppia is a software that allows dynamic modules based on student responses. This allows students to skip certain sections they understand or already know or get additional information if needed.

For making videos I will be using screencapture software for Ubuntu (Kili, 2016) along with video editing software on Windows for the narrated sections.

In terms of sample code I will initially host it on my git server in a public repository. For stretch goals of an online based documentation that would probably use the sample project/code which spins up a webserver locally with instructions. I would use express, pug, and markdown for the sample server.

I may also create a slack or piazza for questions later on when testing the project among a larger test group.

Reasoning

I am one of the few people who have used the library extensively outside of where it was created. Thus, or other online introductory resources for open source libraries. Valuelink offers significant benefits for React programmers compared to other state management frameworks as demonstrated by the far fewer lines of code required to implement a TODOMVC (Balin, 2016a), a benchmark of sorts used to compare web frameworks.

There are some existing resources for learning and using valuelink, such as their github page which has some simple use cases (Balin, 2016b). Similarly the author wrote a medium post about handling complex state and has implemented a TODOMVC (Balin, 2016a). From personal experience, navigating the example code is not as easy to use compared to say, the lodash documentation which has commentary intermixed with source code. In addition at work I am asked by new users of the library some common questions and gotcha's for more advanced usage that is not present in existing resources. It will be good to put solutions, explanations, and best practices I have found in a comprehensive guide.

Course Requirements

ValueLink is a javascript/node.js library which builds on top of React.js. As an advanced library the course requires knowledge of the underlying libraries. Based on the survey on Piazza introductory material for the pre-requisites was rated as very important for students. Unfortunately time constraints may prevent detailed introductory material from being made.

The initial minimal viable product will be tested with friends and colleagues who are familiar with the pre-requisites. I will work on modules going over pre-requisites for the subsequent version after the MVP. Another reason is to avoid reinventing the wheel. Referring to official documentation when appropriate will avoid prerequisite sections being out of date.

In order to make the course easier, I will be making a Vagrant image with everything set up. While Docker is faster and what we use at work Vagrant is much easier for students to use. In addition while Node.js does work on Windows I will be making the course on Ubuntu[2]. It will be expected that students are familiar with Cygwin, Powershell, or other equivalents if they choose to use Windows[3].

Intended audience

The target group are programmers who plan to use React.js coding “form-like” applications. Due to time constraints the course will probably not be providing detailed guidance on how to get the environment setup on Windows or Mac.

---

[2] Should be mostly compatible with Mac. I can get installation instructions for Mac from a coworker who codes on Mac.

[3] It is possible to develop Node projects in Windows and I do have coworkers who use our code on WIndows. However the logistics of supporting development on windows is far too complicated and time consuming for a single person to go over in the short term of the class. Too many unique quirks and ways things could go wrong that would require dedicated TAs on Piazza.

What this course aims to offer

- A practical guide to Valuelink with knowledge presented by lecture as well as learning by doing.
- Architectural considerations and best practices I've encountered

What this course aims to avoid

- A detailed guide to prerequisites
- Grading students via testing. The goal for the course is to impart practical knowledge.

Measures of Success

After taking the course students should:

- Know how to use ValueLink to create a React based form.
- Debug and resolve problems that may crop up independently in the future.
- Understand common problem types, causes, and solutions
- Understand the general theory of how ValueLink works

Course Outline

Guide to the Outline

This outline lists everything that I would love for the course to have in the course. Of course not everything can be completed in the amount of time available by a single person.

- Green text refers to what I consider the minimal viable product.
- Yellow text refers to topics I would like to be in the course.
- Red text refers to nice to haves which will probably not be finished by the end of this class.

Actual time required to complete sections may vary as I create course content. I tried to be conservative in estimating the amount I could get done. Since I haven't written instructional material before I didn't have much to base off my estimates on.

Introduction

Overview of Valuelink

- Quick overview of what valuelink is, along with a summary of existing resources
- What you should use Valuelink for
- What you should not use Valuelink for

Architecture Theory vs Practice

- This section will go over various Node.js architecture theories which influence existing tools.
- We will talk about Flux, One-Way Binding, and Two-Way Binding
- Will explain why Two-Way Binding is not a bad idea, depending on the use case
- Will compare with other languages such as Python, Java, and Scala.

Running and Debugging

- Introduction to how you debug the sample React/Valuelink code in the course
- Introduction to how to use sample code/start up the virtual machine/etc.
- Advanced information on prerequisites (using Oppia) as needed

Project 1

- Simple project which uses the basics of the library following existing documentation
- Students are expected to debug the project to fix purposely inserted errors
- Sample solutions will be provided along with commentary for students to follow along or reference

Fundamentals

Creating Reusable Components

- Introduction to built in bootstrap compatible components
- Discussion about how to create custom React Components and use it with Valuelink

Project 2

- Students will implement a reusable react component that is called by existing code. Students will be able to play around and see the results of their code live.
- Sample solutions will be provided along with commentary for students to follow along or reference

ValueLink is still One-Way Binding

- Even though Valuelink creates the illusion of two-way binding, it is still one-way and each child link is a copy of the state.
- Commentary examples of mistakes that can be made due to creating two different pointers
- Oppia will be used to make the commentary and will ask the student questions to gauge understanding.

Understanding React State

- Explanation of how Valuelink works with the React State
- Commentary examples of mistakes that can be made due to misunderstanding React
    - Calling .set() twice
    - Not calling .set() or .forceUpdate()
- Oppia will be used to make the commentary and will ask the student questions to gauge understanding.

Debugging State Problems

- Series of videos which explores how I go about debugging problems with state components

Project 3A

- The student will be provided with a project boilerplate. They will be instructed to create a form application which goes over the previous three modules.
- Sample solutions will be provided along with commentary

Inserting and Deleting from Arrays

- React state problems which can occur when you modify arrays. (eg ComponentDidMount and shared React objects).

Project 3B

- The student will be provided with a project boilerplate. They will be instructed to create a form application which goes over the previous three modules.
- Sample solutions will be provided along with commentary

Modifying Parent State

- Demonstrate what to do if a form input needs to modify the parent link state
- Explain onChange and other functions, how it is possible to store actions on a parent link object which is done whenever a child object is modified

Project 3C

- Building on top of project 3B, an addition to insertion/deleting from arrays will be requested which modifies parent state
- Sample solutions will be provided along with commentary

Best Practices

Use Props, Not State

- Why you should use props instead of state for child components
- Have a single source of truth at the top level component.
- When it is okay to use state

Declare Links at the Highest Level

- Even if code seems to work, you should always declare links at the highest level. This avoids problems later on if you happen to need to reference or modify elsewhere.

Performance

- Examples of performance problems and how to handle it. Note this section will require asking a coworker for details as I personally did not face issues in my use cases.

ESLint and Strict Mode

- Go over Javascript best practices
- Mention how to disable these checks in the sample project if necessary
- Talk about some issues with ES6 React Classes where sometimes traditional createClass is required
- Link to Babel.js article on ES6 React

Advanced Use Cases

Integration With Django/Plain Javascript

- Examples of integrating with Django or plain javascript
- Common mistakes while integrating
    - Eg saving a singular valuelink instance to the initial DOM rather than a dynamic reference

Standalone/Dynamic Children Applications

- Advanced use case from work which involves detached children and parent React components.
- Shows how Valuelink can be used to simplify a very complicated workflow in react

Using State Managers

- How Valuelink can work even if you are using Flux or another state manager
- Example of how to code a component to accept either Valuelink or Flux

Using JQuery Libraries

- How to use Valuelink with libraries such as select2 which have their own JQuery managed states.

Asynchronous Callbacks and State Updates

- How to handle asynchronous callbacks (AJAX calls) and state updates.
- Example will build on top of autocomplete libraries such as select2.

Other

Boilerplate Code

- Go into detail over how the provided boilerplate works so students can modify it as needed for their own applications

Sample React Components

- Some generic react components commonly used by myself

Recommended Libraries

- List of good Javascript libraries.

Proposed Schedule

The goal is to spend around 20 hours each week on the deliverables.

Status Check 1 - 06/25/2017

- Get initial MVP test group onboarded by EoW
- Reach out to the creator of the library to see if they would have time to review course material
- Rough draft of course text (green sections in fundamentals) sent to initial MVP test group for feedback at EoW.

Milestone 1 - 07/02/2017

- Complete Projects 2 & 3A, 3B
- Finish green sections in fundamentals

Status Check 3 - 07/09/2017

- Complete Project 1
- Get VM and pre-requisite material section completed
- Create poll for which yellow sections students are interested in. Ask for signups from fellow classmates who want to test the alpha version of course
- Work on documentation for milestone 2.

Milestone 2 - 07/16/2017

- Revisions to course from feedback. Revise based on feedback from creator of library if applicable.
- Finish all green sections, add project solutions
- Put material on OpenEdX to open up alpha of course from the class.

Status Check 5 - 07/23/2017

- Continue revisions based on feedback
- Provide support, clarifications to alpha cohort students
- Finish sections in yellow, priority/order based on alpha cohort poll

Final Project - 07/30/2017

- Work on project, paper, and presentation.

Future - ?/?/2017

- Complete all sections in yellow
- Make a web based site for ValueLink similar to Pretty Swift/Discover Meteor so those interested don't need to sign up for OpenEdX
- Make sure course is correct based on feedback from creator of library

- Further revisions to course based on additional feedback/testing. Will probably be included in training program at work.
- Ask a better writer to copyedit the course material.

References

Balin, V. (2016a). gaperton/todomvc-nestedreact. Retrieved May 28, 2017, from https://github.com/gaperton/todomvc-nestedreact

Balin, V. (2016b). Volicon/NestedLink. Retrieved May 28, 2017, from https://github.com/Volicon/NestedLink

Coleman, T., & Sacha, G. (2015). Discover Meteor. Retrieved May 28, 2017, from https://www.discovermeteor.com/

Johnson, B. (2015). Pretty Swift. Retrieved May 28, 2017, from http://www.prettyswift.co/lessons/introduction-to-swift/

Kili, A. (2016). 8 Best Screen Recorders for Desktop Screen Recording in Linux. Retrieved June 18, 2017, from https://www.tecmint.com/best-linux-screen-recorders-for-desktop-screen-recording/

Piazza @360. (2017). Retrieved June 4, 2017, from https://piazza.com/class/j2drfcbtkjabn?cid=360