Kentico

Lecture 5

Petr Svirák

Kentico

Table of contents

There are **few bugs** in some of the **tags** mentioned **in** slides' **headers** that follow. While there is **stable** code under the **latest** tag (lecture-5), you need to **fix** bugs **manually** when **checking** out a bug-**affected** commits. See tags in **GitHub repository** to find **what** to **fix** and **how** fix it.

Kentico

Routing

React 16

- React 16
  - Fiber
  - Better errors handling
  - Return multiple elements

- Read on:
  1. https://reactjs.org/blog/2017/09/26/react-v16.0.html
  2. https://edgecoders.com/react-16-features-and-fiber-explanation-e779544bb1b7

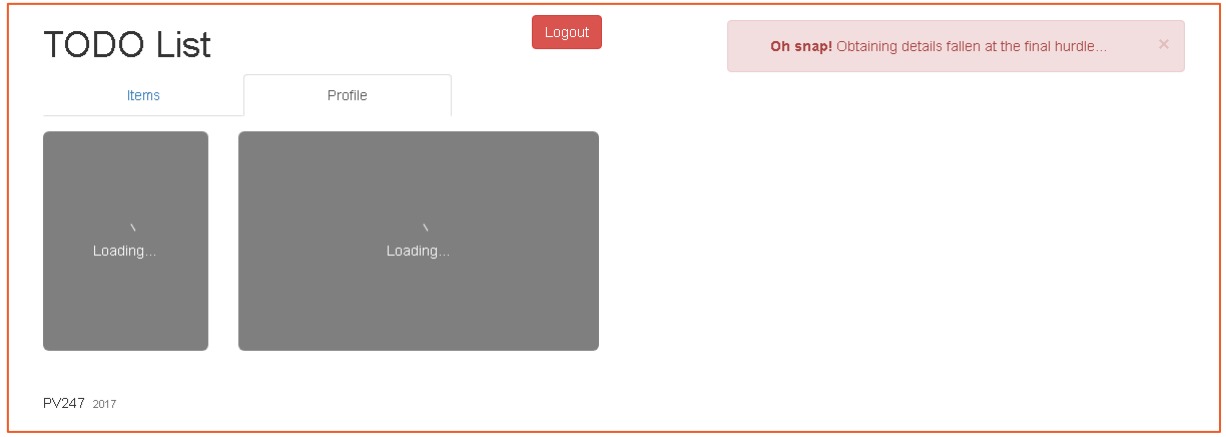
Static files

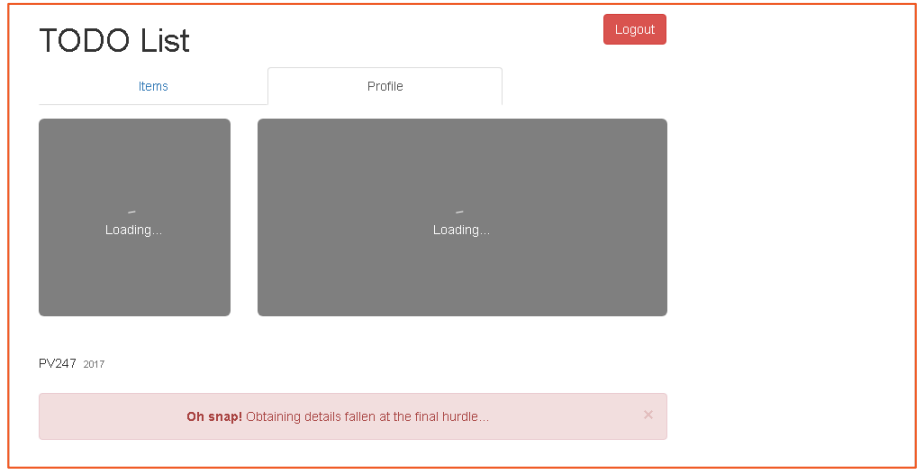
- Serving media files
- Changing app.html

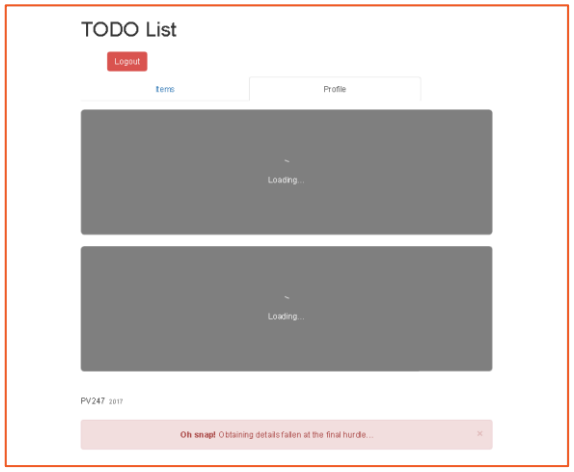
- Read on:
  - https://stackoverflow.com/a/27651720/1138663

Layout

- Structure changes
  - strongly affected by bootstrap design
  - matter of taste/personal (dis)ability







Kentico

Routing philosophy

- Static routing
  - Routes as component (single all-mighty switch)
  - pre-v4 react-router
- Dynamic routing
  - Route as you render (per-component switch)
  - More aligned with React philosophy
  - v4 react-router
- Read on:
  - https://reacttraining.com/react-router/core/guides/philosophy

react-router – different routers

- BrowserRouter
  - Based on HTML5 history API
  - Requires server handling dynamic requests (response to any possible URI)
- HashRouter
  - Based on window.location.hash
  - Can be used with static websites (server-less)
- MemoryRouter & StaticRouter – mostly used for testing
- Read on:
  1. https://medium.com/@pshrmn/a-simple-react-router-v4-tutorial-7f23ff27adf
  2. https://reacttraining.com/react-router/web/api/Router

react-router – routing

- Route
  - path → [component[1]|render[1]|children[2]]
    [1] - renders only when path matches [2] - always renders
  - exact
    - no sub-path makes match
    - not by default
  - Enhances inner components' props
- Link
  - Renders link to given route
  - Nastily customizable

- Read on:
  1. https://reacttraining.com/react-router/web/example/basic
  2. https://reacttraining.com/react-router/web/api/Route
  3. https://reacttraining.com/react-router/web/api/Link

With Redux

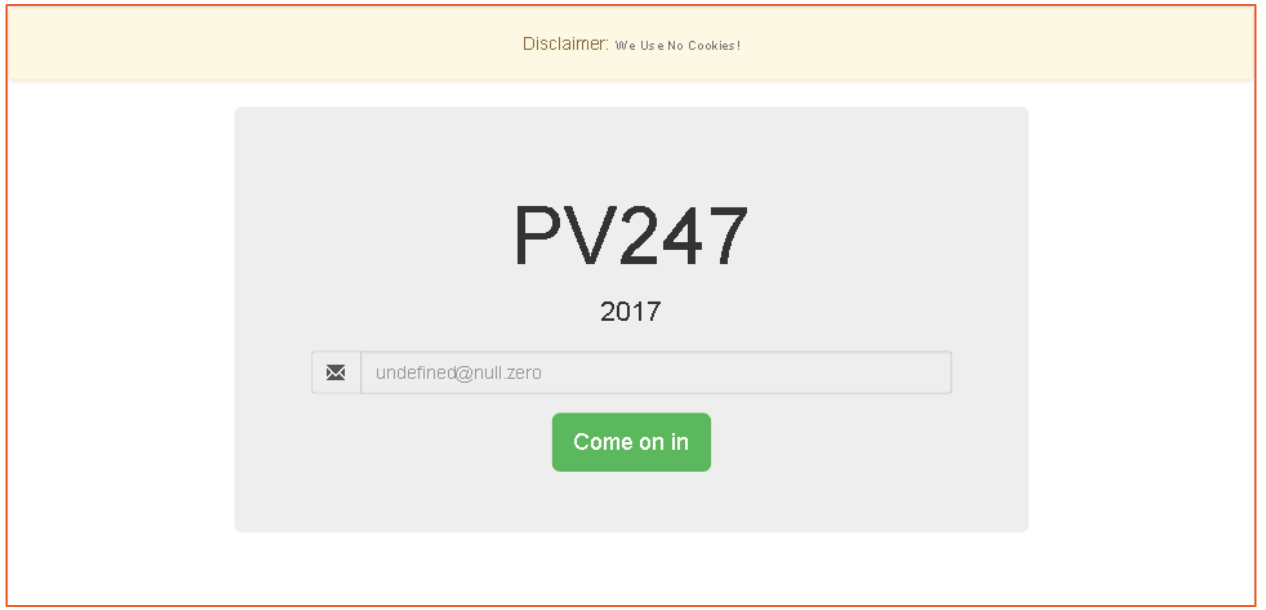
- react-router-redux (official plugin, but alfa only)→ connected-router-redux (same API)

- Why
  - Enabled time-traveling
  - Allows changing route based on an user action (redux thunk)

- Read on:
  1. http://redux.js.org/docs/advanced/UsageWithReactRouter.html
  2. https://github.com/supasate/connected-react-router
  3. https://github.com/reactjs/react-router-redux

Complex example: Login page

- Unauthenticated → login page
- Authentication → persist token
- Logout → a button
- Support refresh → token in storage



Page head

- Browser history shows static title (in great numbers)
- Many sites requires/reads specific tags (SEO, social networks, …)

- Read on:
  - https://github.com/nfl/react-helmet

Kentico

Forms

redux-forms

- Most applications has something for user to fill in
- Most react-redux application solve the same problems (and in the same way)

- Field
  - Wrapper component
  - props.input & props.metadata

- reduxForm
  - HoC like connect
  - static & dynamic configuration

- Read on:
  - https://redux-form.com/7.0.1/docs/gettingstarted.md/

Merging classes

- Usually required when rendering eye-candy components (e.g. validate input fields)
- Code becomes clutter with if statements (not so eye-candy in the inside)

- NPM to the rescue (for example): https://www.npmjs.com/package/classnames

redux-forms - validation

- Validation
  - Field-level validation
    - code readability
  - Form-level validation
    - sort-of legacy
    - client-side validation in general speeds up UX & prevents unnecessary traffic
  - Server (async) validation
    - Some validations (e.g. uniqueness) need most up-to-date state to work properly
    - Form/App should not freeze only because of request

- Read on:
  1. https://redux-form.com/7.0.1/examples/fieldlevelvalidation/
  2. https://redux-form.com/7.0.1/examples/asyncvalidation/
  3. https://redux-form.com/7.0.1/examples/syncvalidation/

Fetch

(Isomorphic) fetch

- Replacement API for XMLHttpRequest
  - Provides way to make asynchronous calls via HTTP
- Not support by all browsers
- Good polyfill: https://www.npmjs.com/package/isomorphic-fetch
  - fetch is but a function (window.fetch)
  - Only connection failures are reported as errors
  - Requires promises to work, supports Node & Browserify
- Read on:
  1. https://github.com/github/fetch
  2. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Promises

- Promise is an object
- Promise always returns Promise
- Promise is always either: Pending or Fulfilled or Rejected

- Represent eventual completion (or failure) of an asynchronous operation
- Newly created Promise accepts resolve and reject callbacks
- Object returned from a promise's then/catch methods means new fulfilled promise
- Object thrown from a promise's then/catch methods means new rejected promise

- Helps solving callback hell (pyramid of doom)

- Read on:
  1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
  2. https://strongloop.com/strongblog/node-js-callback-hell-promises-generators/
  3. https://github.com/stefanpenner/es6-promise

Complex example: Authentication

- Token authentication
  - API requires an email (no passwords in this API) before providing a token
  - Token is needed for all other API requests

- Token expiration
  - API sends no information about token's expiration
  - Assume a time span when token is surely valid and throw token away after given time
    - This avoids unnecessary API call that will surely fail
    - Any call can still fail when token really expires → this indicates no further request will succeed → user should be logged out (this is not implemented in this example)
- Read on:
  1. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
  2. https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication
  3. https://jwt.io/
  4. https://www.npmjs.com/package/react-loader-advanced

Complex example: Get profile details

- Load details from API
  - Data were pre-stored in customData field
  - Authenticated GET request is required by API
  - User should be logged out when token is invalid (status code 401)
  - User should be informed that an asynchronous operation is running

- Show errors on the right side of the app

Async/await

- Async functions always return Promise

- Allows flattening nesting Promise hell
- Allows using try-catch(-finally) to handle Rejected promises

- Read on:
  1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function
  2. https://strongloop.com/strongblog/node-js-callback-hell-promises-generators/
  3. https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9
  4. https://babeljs.io/docs/usage/polyfill/

Complex example: Change profile details

- User updates their details
- API does not validate details in any way
  - stores them as a string in customData field
- User should be notified the data are being stored since the operation can take some tike

Upload file

- FormData API

- react-dropzone
  - Allows handling file upload in a React way
  - https://www.npmjs.com/package/react-dropzone

- Read on:
  1. https://developer.mozilla.org/en-US/docs/Web/API/FormData
  2. https://abandon.ie/notebook/simple-file-uploads-using-jquery-ajax

Complex example: Upload avatar

- Hovering over existing avatar shows an overlay
- Overlay allows clicking or dragging an image in supported format to it
- Image is uploaded to the server storage
- User's custom data are updated with uploaded image id
- Avatar is reloaded (new link is used)