Practical Assessment 2

Adding Interactivity Add Edit & Delete

Weight: 25%;

Due Date: Check Class Moodle Instance

Introduction

In the last assignment you created most of the UI for the todo app. In this assignment you are going to create the reset of the pages required to complete the add edit and delete functionality required in the app.

As with the last assessment your sites will be graded at a maximum viewport size of 360px 756px . After the maximum viewport size is reached you can use a center page layout to frame the app for larger viewports.

Currently Pages

Your app currently has 3 pages:

1. the landing page
2. the to do list page
3. the custom 404 page.

New Pages To Build

Create the following new pages for the functionality that you're going to add to the todo list app.

1. Delete Page.
2. Add New Employee Page.
3. Edit Employee Page.

Required Functionality

You have to add the logic to the edit add and delete buttons in your app. The easiest way to perform this is to create a page for each task. In class you created the delete functionality. You will do the same for the add and edit functionality. With each new page you will have to add a route to the router. Keep the route path lower case. You will have to create a page that provides the end user with the controls to perform the task or cancel the task and return to the todo page. You can download the class employee directory with the delete functionality from https://github.com/dominait/employee-directory.

404 Page Not Found Update

Step 1. Update Router
Step 2. Create Redirect File
Step 3. Update the gulpfile.js
Step 4. Upload and Test

1. Update The Router

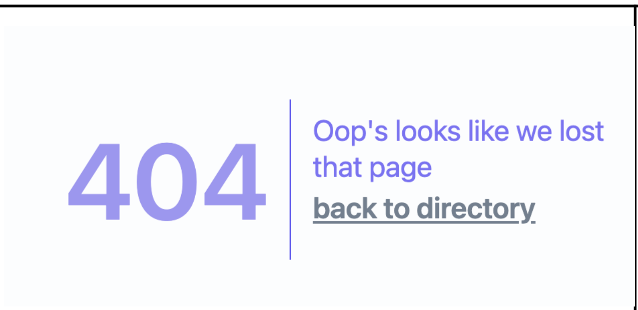
You will need to tweak your router to get it to work with netlify. Below you will find a copy of the update router. It adds a route /* to catch unhandled routes. Make the following changes to your router before beginning the assessment. Follow the next steps to create an _redirect file and update the gulpfile to add the _redirect file to the /dist folder.

```
 import homePage from '../pages/home'
import employeeDirectory from "../pages/directory";
import deletePage from "../pages/delete/";
import notFound from "../pages/notfound";

const routes = {
   "/": homePage,
   "/directory":employeeDirectory,
   "/delete":deletePage,
   "/*":notFound,
}

const Router =  function (pathname, params=null)   {
const isValidRoute =   Object.keys(routes).find(key=> key===pathname)
   const app = document.querySelector('#app')
   app.innerHTML = ''
   window.history.pushState(
      {},
      pathname,
      window.location.origin + pathname
   )
      if(isValidRoute === undefined || isValidRoute ===''){
         app.appendChild(notFound())
      }else{

         app.appendChild(routes[isValidRoute](params) )
      }
}
export { Router}
```

2. Create The _redirects File

In addition to tweaking the router you will have to add a file to the root of the development folder called _redirects.

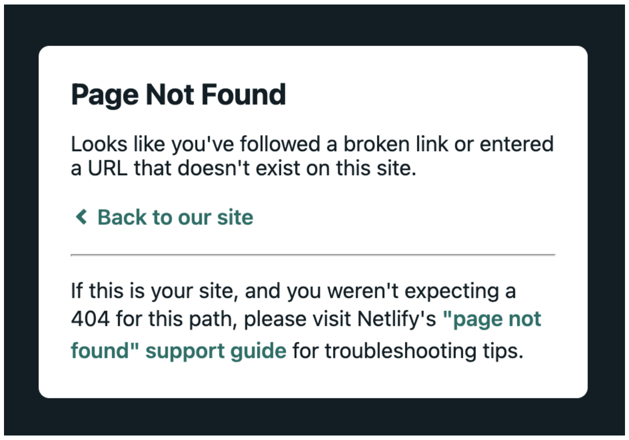
This file instructs Netlify to show your custom error page and not the default Netlify Page not found message.



404

Oop's looks like we lost that page

back to directory



Page Not Found

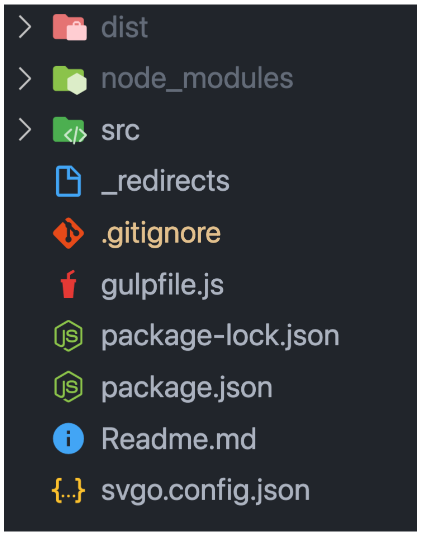
Looks like you've followed a broken link or entered a URL that doesn't exist on this site.

Back to our site

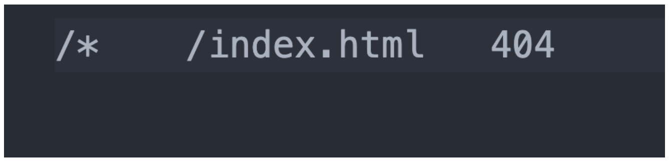
If this is your site, and you weren't expecting a 404 for this path, please visit Netlify's "page not found" support guide for troubleshooting tips.

Create a file called _redirects in the root of your project. In the _redirects file type the code you see below. This code matches the router /* path and reloads the index.html page allowing our notFound page to display.

/*   /index.html   404



```
dist
node_modules
src
_redirects
.gitignore
gulpfile.js
package-lock.json
package.json
Readme.md
svgo.config.json
```

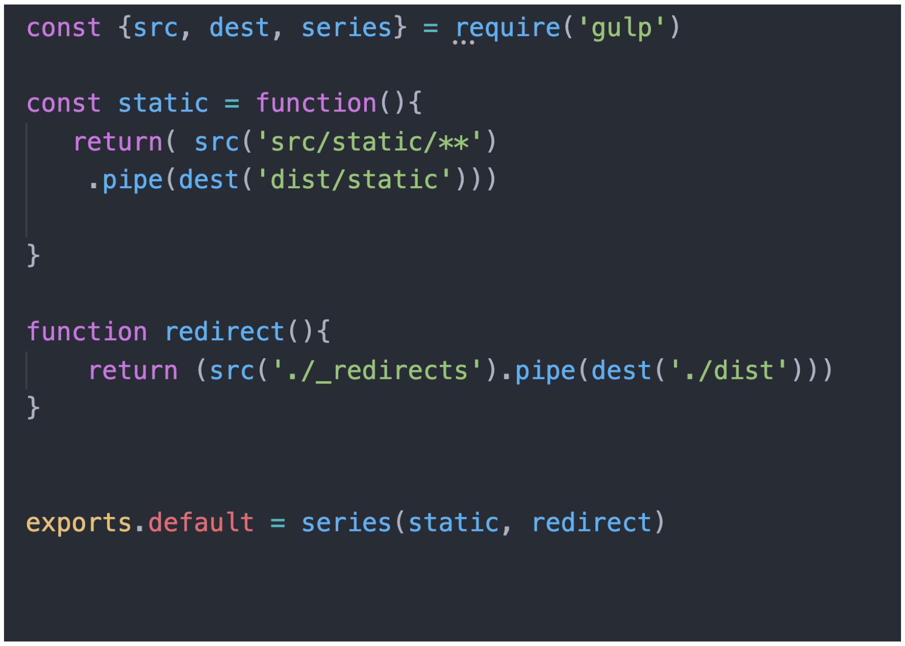


```
/*     /index.html    404
```

3. Update the gulpfile.js

Open your gulpfile.js. You need to update the file to move the _redirects folder at build time into the dist folder. Netlify uses the dist folder as the project's root folder. The _redirects file must be inside the root folder so we need to move it using gulp.

Below you will find the code to update your gulp file. The file has been updated to move the _redirects file into the folder at build time. You do not have to do anything to the package.json file.



```
const {src, dest, series} = require('gulp')

const static = function(){
   return( src('src/static/**')
    .pipe(dest('dist/static')))

}

function redirect(){
    return (src('./_redirects').pipe(dest('./dist')))
}


exports.default = series(static, redirect)
```

4. Upload and Test

Update your git repo and test your site on Netlify. You should now see your custom 404 page. This page will only display for paths at netlify.app/ and not work with paths /todoooodo/something.

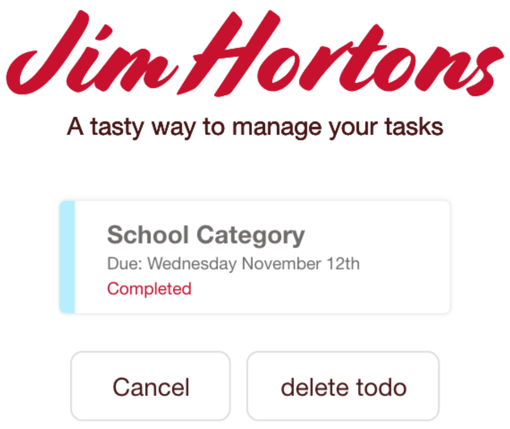
Visit https://github.com/dominait/employee-directory there is a working copy of the router, _redirects file and gulpfile.js that can be used.

Build Requirements

deletePage.js

The delete page should remove a todo item from the list. The easiest way to do this would be to create a page for the delete options. From the todo page send the delete page the item the end user is trying to delete as an object. Pass the data to the router and then forward the data onto the delete page.

Display the todo item being deleted along with the two button options and the page branding header.


Jim Hortons
A tasty way to manage your tasks
School Category
Due: Wednesday November 12th
Completed
Cancel
delete todo

Components

For this page you will display the todo item being deleted and offer the end user two options.

Cancel
This option should take the user back to the todo list without performing the delete.

Delete Todo
For this action send an object to the reducer function and remove the item from the store. Update the store with the new state. Once the store is updated use a callback function that returns the user to the todo list.

You can't use the artwork shown, you must create your own artwork for your app. When styling the app make sure that content gets the focus and not your interface. The interface should be subtle and in the background. Content is the most important thing on the screen along with a clear visual representation of the actions to perform.

Build Requirements

editPage.js

For this page when the user requests and edit to a todo item pass the object form the store to the router and then to the edit page using page props.

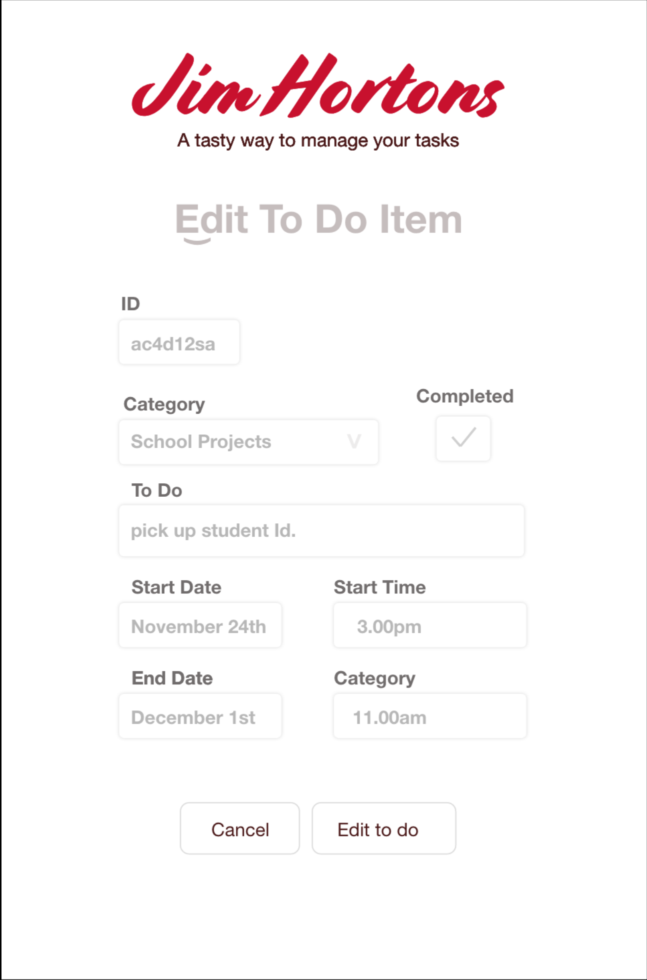
This screen should have form elements that are populated with the data from the object passed from the todo list page.

The end user can make edits then either cancel or edit the to do item. If the user chooses to edit the item pass the a data object to the store as a payload with the edit action.

Then update the object using its id to search the current store's to do items by id. When you find the item replace it using a technique similar to the techniques shown in the video's below.

https://egghead.io/lessons/react-redux-avoiding-array-mutations-with-concat-slice-and-spread

https://egghead.io/lessons/react-redux-avoiding-object-mutations-with-object-assign-and-spread



todo.js components

Form Field
Add form fields for each property in the todo object found in the todos.json file.

Actions

Cancel
End user can cancel out of the operation and return to the todo list page.

Edit todo
Create an object from the form field elements. This object should have the same properties as the objects you created in assessment one.

```
{
 id,
 category,
 title,
 isCompleted,
 startDate,
 startTime,
 endData,
 endTime
}
```

Pass the data object to the reducer function along with an action type. Find the todo item using the unique id that you assigned to the item. Update the data and update the store.

When you finish the edit create a callback function that returns the user to the todo list page.

Build Requirements

addPage.js

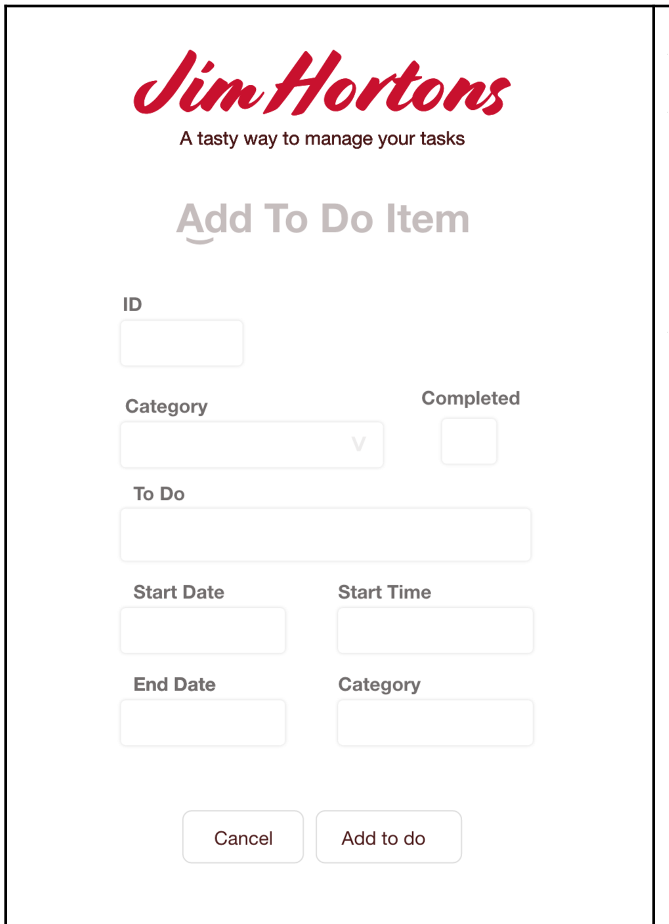
For this page when the user wants to add a new todo item take the end user to a page where they can fill out all of the required fields to build a data object. An easy way to handle this is when you enter the screen create a unique id and enter it as the value of the id input. Set a default value for the category drop down if you use one.

Once the end user has filled out all the fields and clicks the add to do. Send the input values as an object to the reducer function along with an action to perform.

For the add action you just have to add the item to the store. Try one of these non destructive techniques.

https://egghead.io/lessons/react-redux-avoiding-array-mutations-with-concat-slice-and-spread

https://egghead.io/lessons/react-redux-avoiding-object-mutations-with-object-assign-and-spread



todo.js components

Form Field
Add form fields for each property in the todo object found in the todos.json file.

Actions

Cancel
End user can cancel out of the operation and return to the todo list page.

Add todo
Create an object from the form field elements. This object should have the same properties as the objects you created in assessment one.

```
{
  id,
  category,
  title,
  isCompleted,
  startDate,
  startTime,
  endData,
  endTime
}
```

Pass the data object to the reducer function along with the action to be performed.

Add the todo item to the store and make sure you have provided the item with a unique id. Update the store and then use a callback function to return the end user to the todo page.

Submission

You are required to submit the url to the deployed app on Netlify and the url for the GitHub repository that contains the files for your todo app.

Submit To Moodle
Netlify todo app url
GitHub repository url

Late submission will not be graded.

Marking Key

Task	Mark	Task Value
deletePage.js		
page components and styles logo title and tagline buttons and todo item and any other components that you wish to add to customize your app.		3
Cancel Action return to the todo list page		3
Delete Action remove to do item from the store and update store + display		3
editPage.js		
page components Form inputs page title and tagline buttons and any other components that you wish to add to customize your app.		3
Cancel Action return to the todo list page.		3
Edit Action edit the todo item, update the item in the store. Edit should appear in the todo list page.		3
addPage.js		
page components Form inputs page title and tagline buttons and any other components that you wish to add to customize your app.		3
Cancel Action return to the todo list page.		3
Add Action add a new todo item, add the new item to the store. New to do item should display in the todo list page.		3
GitHub Repository & Deployment Contains a readme.md file with instructions on how to start and run the project. There should be a link to the deployed site. Repository must be deployed to Netlify with a custom 404 page for domain sub paths: **netlify.domain/wrongpath** Do not worry about 404 errors with longer sub paths: **netlify.domain/wrongpath/wrong/**		3
		30

Marking Rubric

Marks	5 Marks Criteria
5 [0]	Task was completed with the highest proficiency, adhering to best practices. The tasks demonstrate a high level of proficiency implementing the subject matter and the presentation of the content is to a professional standard.
4 [-1]	Task was completed well, with some minor mistakes. Well above average work, shows good understanding of the implementation details of subject matter. Tasks show a high degree of competence.
3 [-2]	Task was completed satisfactorily. Some features are missing or incorrectly implemented. Shows a moderate level of understanding in the task with room for improvement.
2 [-3]	Task completion is below average, the task was poorly completed. Shows understanding of the task and the requirements to implement, but implementation was poorly executed.
1 [-4]	Some of the tasks were completed. Shows a lack of understanding in the subject matter and very poor execution.
0 [-5]	Not completed.

Marks	3 Marks Criteria [minus]
3 [0]	Task was completed well, adhering to best practices, and shows a high degree of proficiency with the subject matter guidelines.
2 [-1]	Task was completed satisfactorily. Some features are missing or incorrectly implemented. Shows a moderate level of understanding in the task with room for improvement.
1 [-2]	Some of the tasks were completed. Shows a lack of understanding in the subject matter and very poor execution.
0 [-3]	Shows a little to no degree of competence in completing the task; not completed.

Marks	1 Marks Criteria
1	Task completed satisfactorily
0	Task was not completed