Netninja

Repo link: <https://github.com/dat09loz/react-learning>

# Create a React Application

Npx create-react-app <app-name>

Index.js will render the app component and inject it in the root div (in index.html)

# Components and templates

Components contain templates and logic

Index.js: kickstart the application

JSX template: component always returns jsx template

ReactDOM.render()

# Dynamic Values in Templates

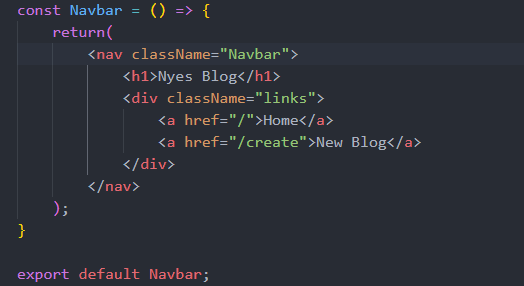
Text

Description automatically generated

# Multiple components

App.js: root component

* Nest other components to the root component
* Navbar.js



* Use it inside App.js

Text

Description automatically generated

# Click events

Separate function and invoke it by onClick

Text

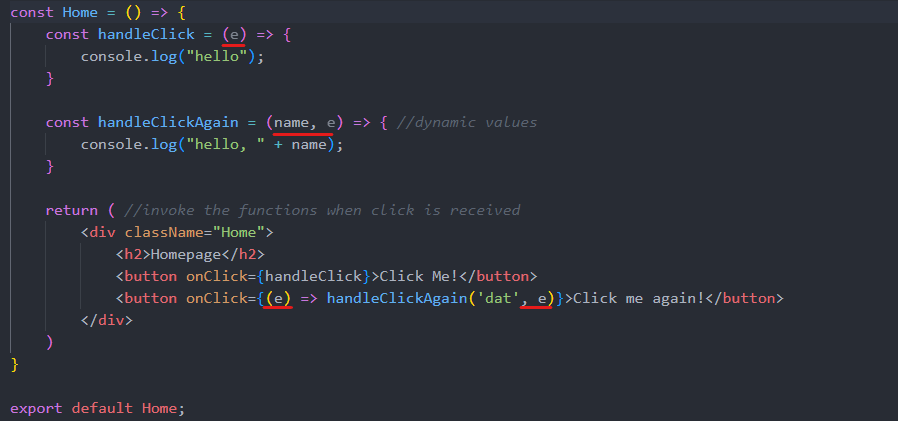
Description automatically generated

Anonymous function (pass in dynamic values) – wrapping in another function to prevent it from invoking immidiately

Graphical user interface, text

Description automatically generated

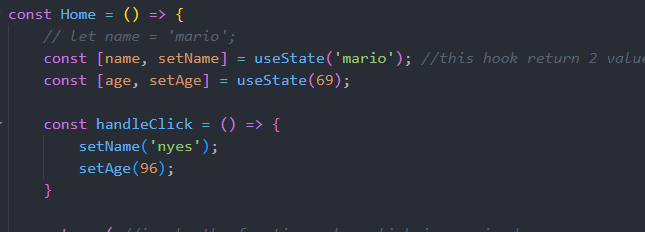
Event object (when the user do something – click a button)



# Using states

Create a “reactive” variables: a state hook useState

Use normal import if the component has a default export (export default <component>), it will return the default no matter how you call it; named export {<component>} export the specific export component name (if it allow export ofc)



* The first argument define a variable, second one define the action that will be execute after a click

# Intro to React Dev Tool

React development tools

# Outputting List

Map method: cycle through the array and do something to each of them

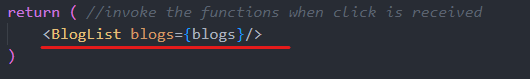
Key property: keep track of each item of the DOM as they output it

Text

Description automatically generated

# Props and Reusable Components

Props: pass data from a parent component to a child component

home.js (parent)

Text

Description automatically generatedBlogList.js (child)

* Multiple props 
* Destructor the props directly

Text

Description automatically generated

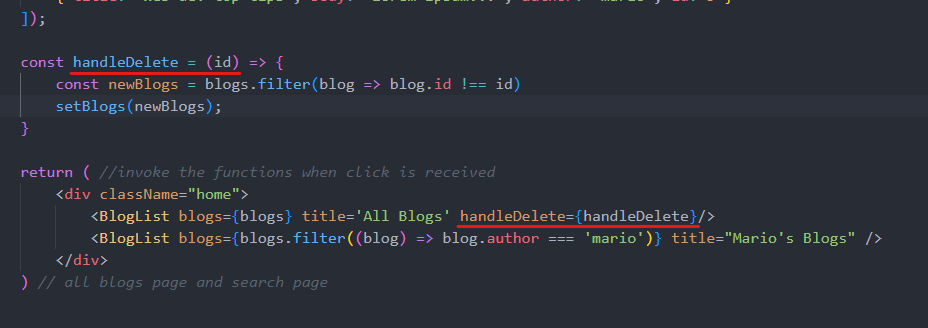
# Reusing Components

Text

Description automatically generated

# Functions as Props (handle delete)

Define the function in the parent component and pass it in as a prop



Text

Description automatically generated

# useEffect hook

run the function every render() of the component (page reload, etc.), and when the state changes (useful for fetching data for example)

Text

Description automatically generated with medium confidence

* No return values

# useEffect Dependencies

e.g. when we only need it to run at certain times, not every time the state changes - -

- empty array: useEffect only run in the initial render

Text

Description automatically generated

* Variable array: useEffect only run initially, and every time the variables’ state change

Text

Description automatically generated

# Using JSON server

Create db.json

Json-server: a fake REST API server: npx json-server –watch data/db.json –port 8000

* Watch the db.json file and host it on port 8000

Endpoints



# Fetching data with useEffect

Fetch().then(return js object).then(do stuff with JS object)

Text

Description automatically generated

Conditional template

Graphical user interface, website

Description automatically generated

* The one on the right will be render only if the left is true

# Conditional loading message

A loading screen while the data is being fetched

Text

Description automatically generated

Another conditional template

Text

Description automatically generated

# Handling errors

When the connection to the server failed: catch method

When the response is not as expected:

* res.ok method
* throw Error(<err msg>)

Text

Description automatically generated

# Making a custom hook

A custom useeffect we wrote above, can be used everywhere

Custom hook name starts with use: useFetch.js

useFetch return:

Text

Description automatically generated

Use it in Home.js

Text

Description automatically generated

# React Router

Install the package: react-router-dom

Import router package in App.js

Text

Description automatically generated

* router, routes (switch) and route
* navbar is not in the switch because it will be there in every page
* the path will show their respective page

# Exact Match Route

Create a new page: Create.js and put a route in App.js

Text

Description automatically generated with medium confidence

# Router Links

Link component in react-router-dom: intercept requests and handle the content changes in the browser, instead of sending fresh requests (HTML request to the server every time)

* If you click “Home” in the home page, Link won’t send the request back to the server, but <a href> will send back every time you click “Home”

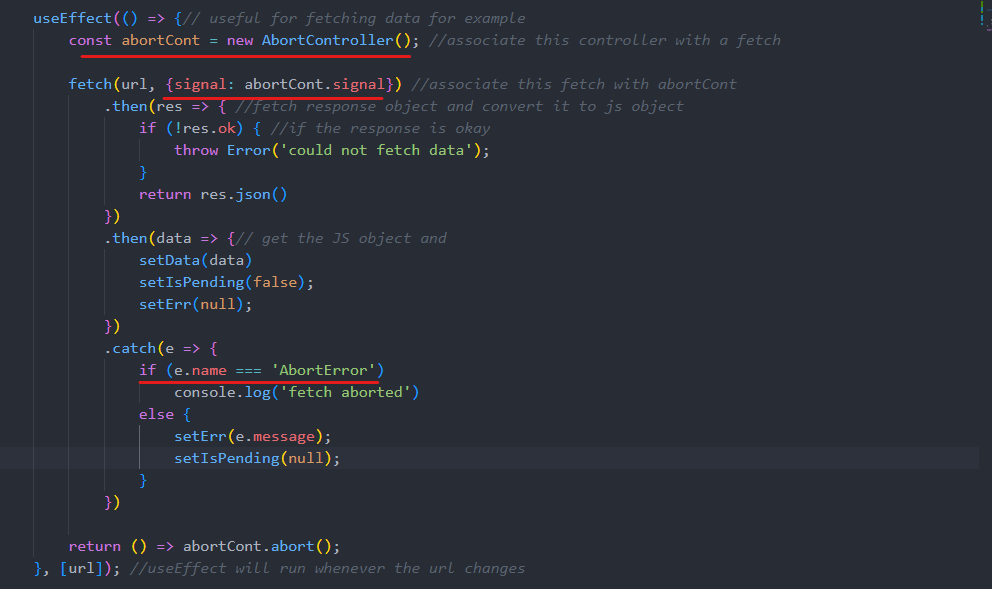
Text

Description automatically generated

# useEffect cleanup

To prevent memory leak (quickly switch to another page while the current page is still fetching data), and other stuffs

Implement an “abort controller” – use it inside useEffect function (useFetch.js)



# Route Parameters

Dynamic routes to the URLs (e.g. blog detail page): name it after “:”



* Use a hook to grab route parameter to the url: useParams

Text

Description automatically generated

Configure the blog list (BlogList.js)

* Wrap each blog item list inside a link

# Reusing Custom Hooks

Reusing the useFetch.js for fetching blog details

Text

Description automatically generated

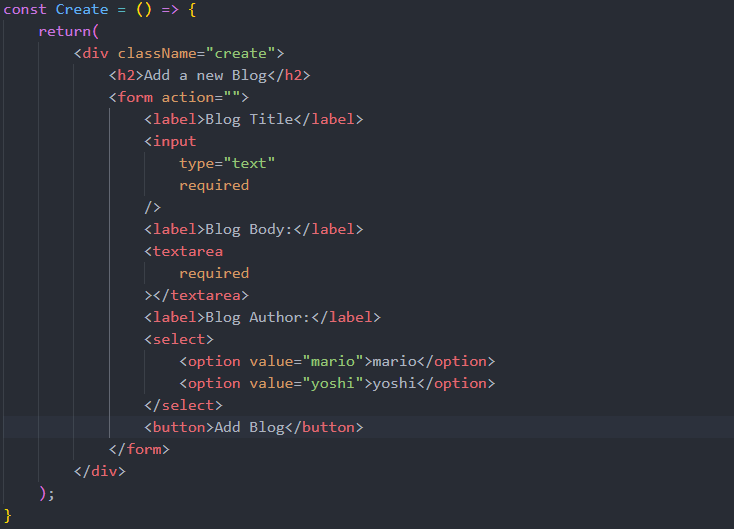
* The <article>tag specifies independent, self-contained content: foum post, blog post, news story ,…

# Control inputs (forms)

Control input is implemented in input field and forms to track its value, and then store it in some type of state

Try it out in Create.js

* Create a input form



* Track input value and store it into state

Text

Description automatically generated

* + Assign “input” tag value to the title state
  + Whenever the input value change, it invoke the event change “e”, when it will set the title “setTitle()” to the target’s (this input tag) value
  + The same goes to body and author

# Submit a form

A new function to handle submit (Create.js)

* <form onSubmit={handleSubmit}>: call the function after pressing the button

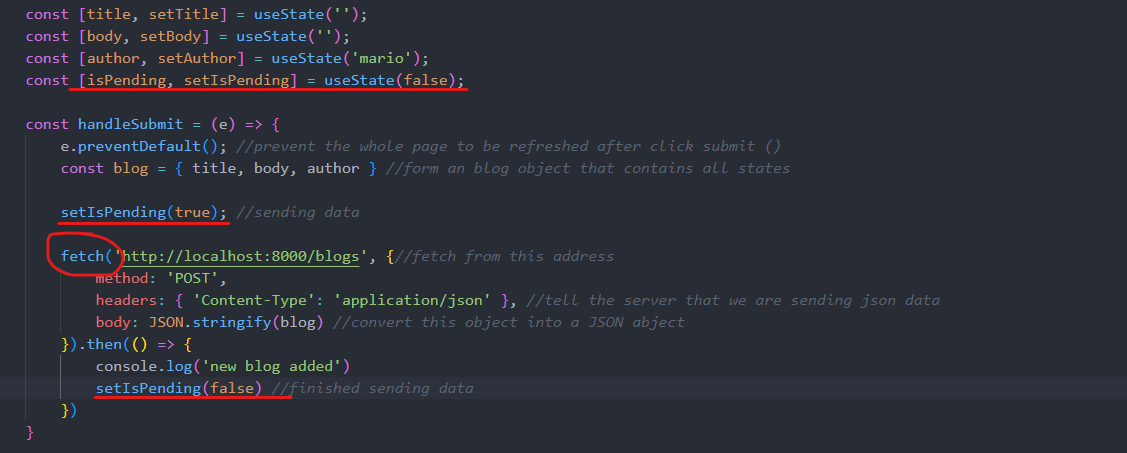
Text

Description automatically generated

* + Event.preventDefault prevent a page from reloading after submit

# Making a POST request

* A request to a server: fetch



* Changing add button’s state when sending data to the server

Text

Description automatically generated

# Programmatic Redirect

Send the user back to the homepage after adding a blog (Create.js)

Hook: useNavigate (router-dom v6)



Text

Description automatically generated

# Deleting Blogs

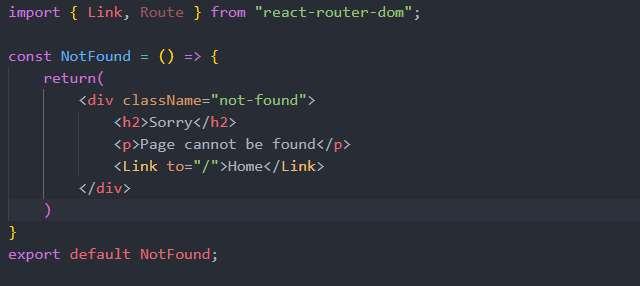
A fetch request to the server with method DELETE (blogDetails.js)

Text

Description automatically generated

# 404 Page

NotFound.js



Set up a match-all route (if not match other defined routes) in App.js (should be at the very bottom of the routes list)



ReactJS Tut Official website

Components

Remember the action: the component use “state”: constructor: this.setState({prop: value})

It is good to store states in parent components