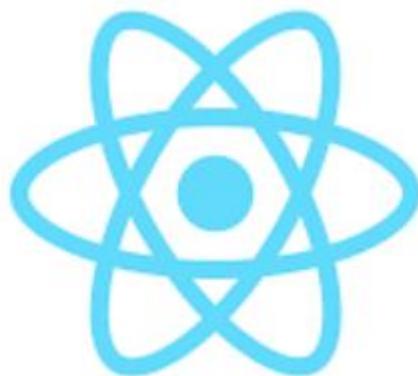


REACT JS

MANUAL BOOK (FREE VERSION)



By

Tin Mai Zaw

Northern City

published in 2024

အမှာစာ

†††††††††

ဤစာအပ်သည် React JS ကို အခြေခံမှစတင် လေ့လာမည့် သူများအတွက် လက်ခွဲစာအပ်ဖြစ်ပါသည်။ ကွန်ပျိုတာကျောင်း သူကျောင်းသားများ၊ အင်ဂျင်နီယာအိုင်တီ ကျောင်းသူကျောင်းသားများကို တစ်ဖက်တစ်လမ်း အထောက်အကူပြုစေရန်အတွက် ရည်ရွယ် ထုတ်ဝေရခြင်းလည်းဖြစ်ပါတယ်။ အိုင်တီနဲ့ အသက်မွေးဝမ်းကြောင်းပြုလိုသူများ၊ software programmer ဖြစ်ချင်သူများ ကဗ္ဗာကျော် software application တွေတိတွင်ပြီး millionaire သူငြေးဖြစ်ဖို့ စိတ်ကူးအိမ်မက်ရှိသူ မည်သူမဆို လွယ်လွယ်ကူကူ လေ့လာနိုင်အောင် လေ့ကျင့်ခန်း ဥပမာများ၊ သင်ခန်းစာ video tutorial များနဲ့ တွဲပြီးတော့ ပြည့်ပြည့်စုစုပေါင်း ရေးသားဖန်းတီးထားတဲ့ စာအပ်ဖြစ်ပါတယ်။



စာရေးသူ ကိုယ်ရေးအကျဉ်း

ဤစာအပ်ကိုရေးသာပြုစုသူကတော့ ကျနော် ဆရာတင်မိုင်အော် ဖြစ်ပါတယ်။ ကျနော်က မြစ်ကြီးနားမြို့ အထက (c) မှာ အထက်တန်းအောင်မြင်ခဲ့ပါတယ်။ ပြီးတော့ ကျနော်က ဖိလိပိုင်နိုင်း University of the Philippines (Los Banos) မှာ B.Sc Computer Science ကိုဆက်လက်ပညာသင်ယူခဲ့ပါတယ်။ ၂၀၀၄ မှာ အိုင်တီနဲ့ ဘွဲ့ရကျောင်းပြီးခဲ့ပါတယ်။ ၂၀၀၅ မှာ မြန်မာပြည်ပြန်လာပြီး မြစ်ကြီးနား ဘတိမြို့မှာ Northern City Computer Training Center ကွန်ပျိုတာသင်တန်းကျောင်းကို စတင်တည်ထောင်ဖွင့်လှစ်ခဲ့ပါတယ်။ လေ့လာဆည်းပူးခဲ့တဲ့ programming IT ဘာသာရပ်တွေကို သင်ကြားပိုချုံတာ ဖြစ်ပါတယ်။ ၂၀၀၉ မှာ မလေးရှားနိုင်း Kualalumpur မှာ zepto it solution လို့ခေါ်တဲ့ အိုင်တီ Company မှာ Software Developer (programmer) အဖြစ် ၃ နှစ်တာ အလုပ်လုပ်ခဲ့ပါတယ်။ ကျန်းမာရေးအခြေနေကြောင့် ရန်ကုန်မြို့ကို ပြန်လာပြီး ဆေးကုသရင်း ရန်ကုန်မြို့မှာပဲ ValueStar Computer Institute မှာ ၆ နှစ်တာ IT Lecturer အနေနဲ့ရော IT Department Head အနေနဲ့ရော ဆက်လက်ခြေချွဲပါတယ်။ ၂၀၁၇ မှာ ကိုယ်ပိုင် အိုင်တီသင်တန်းကျောင်းအဖြစ် Northern City Center နာမည်နဲ့ အရင်က မြစ်ကြီးနားမှာ တည်ထောင်ခဲ့ဖူး တဲ့ နာမည်ကို ပြန်လည်အသုံးပြုပြီး ဖွင့်လှစ်တည်ထောင်ခဲ့တာဖြစ်ပါတယ်။

စာရေးသူ၏ အိုင်တီအတွေ့အကြံ၊ မှတ်တမ်းများ -

Popular Web projects –

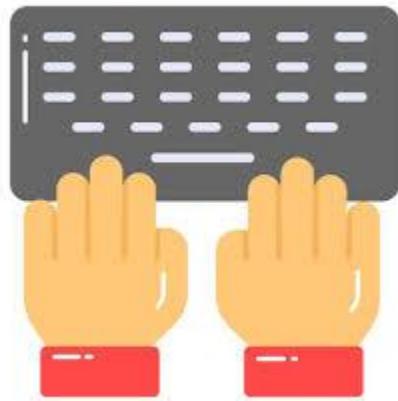
- Aya Internet Banking
<https://www.ayaibanking.com/>
- Malaysia Minimart
<https://familystore.com.my>
- Japan Used Car Sales & Show room
<https://www.sbtjapan.com/sbt-myanmar/>
- Myanmar Traditional Boxing
<https://www.myanmartraditionalboxing.com.mm>
- Myanmar Agriculture Machinery & Products
<https://ptyeeshinn.com.mm>
- Real Estate
<https://www.saikhungnoung.com>
- China Products & Fashion Sales
<https://youhome.space/>
- Online Payment System
<https://ucapay.com.mm>

Popular Point of Sales Application Projects –

- Malaysia Family Store Desktop Application and Mobile Application
- Myitkyina iGu Café & Gallery Desktop Application
- Myitkyina Hospital Blood Bank Desktop Application
- Myitkina Fuji Food & Drinks Desktop Application
- Yangon Joyzone Stock Controller Desktop Application and Mobile Application
- Yangon Malihka Restaurant Desktop Application and Mobile Application
- Yangon Yuri Café Mobile Application and Mobile Application
- Yangon Gracevines Agarwood Desktop Application and Mobile Application

Introduction

 React Video Lesson 1



I. Introduction

a) React JS ဆိုတာဘာလဲ?

React JS သည် web application user interface တွေကို ဖန်တီးရန်အတွက် တို့ထွင်ဖန်တီးထားတဲ့ JavaScript library ဖြစ်ပါတယ်။ React ကို Facebook မှ ဖန်တီးထုတ်လုပ်ထားတာ ဖြစ်ပြီး React library ကို စဉ်ဆက်မပြတ် update လုပ်ဆောင်နေပြီးတော့ error bug များကို ပြင်ဆင်ရင်းနဲ့ component အသစ်များကို ထပ်ထည့်ပြီး ပိုမိုကောင်းမွန်အောင် ကျယ်ကျယ် ပြန့်ပြန်အသုံး ပြုလာနိုင်အောင် ပြင်ဆင်လုပ်ဆောင်နေခဲ့တာ အခါဆို react version 20 အထိရောက်ရှိလာခဲ့ပြီ ဖြစ်ပါတယ်။ သည်စာအုပ်ထဲမှာ React ရဲ့ project setup installation ပြုလုပ်ခြင်း၊ React အခြေခံရေးသားခြင်း၊ component ဖန်တီးခြင်း၊ JSX ကနေ စတင်ပြီး state management ၊ form programming ၊ routing ကဲ့သို့ React JS ရဲ့ အဆင့်မြင့် သဘောတရားများကို example တွေနဲ့ step by step လေ့လာရမှာ ဖြစ်ပါတယ်။ သည်စာအုပ်ကို အတွေ့အကြုံရှိပြီးသား web developer တွေလည်း ကိုကား အသုံးပြုနိုင်သလို အခုမှ စပြီး React ကို လေ့လာနေတဲ့ beginner တွေလည်း အသုံးပြုနိုင်ပါတယ်။ React JS ကို လေ့လာမယ်ဆိုရင် HTML၊ CSS၊ Vanilla JavaScript ဒါမှုမဟုတ် Pure JavaScript နဲ့ programming OOP လောက်အထိ လေ့လာထားဖို့လိုပါမယ်။ Absolute beginner တွေအတွက် မသင့်တော်ပါဘူး။ React JS ကို အသုံးပြုထားတဲ့ popular web application တွေကတော့ Facebook ၊ Instagram ၊ Netflix နဲ့ reddit အစရှိတဲ့ popular platform တွေပဲဖြစ်ပါတယ်။

b) React JS ကို ဘာကြောင့် အသုံးပြုသင့်သလဲ?

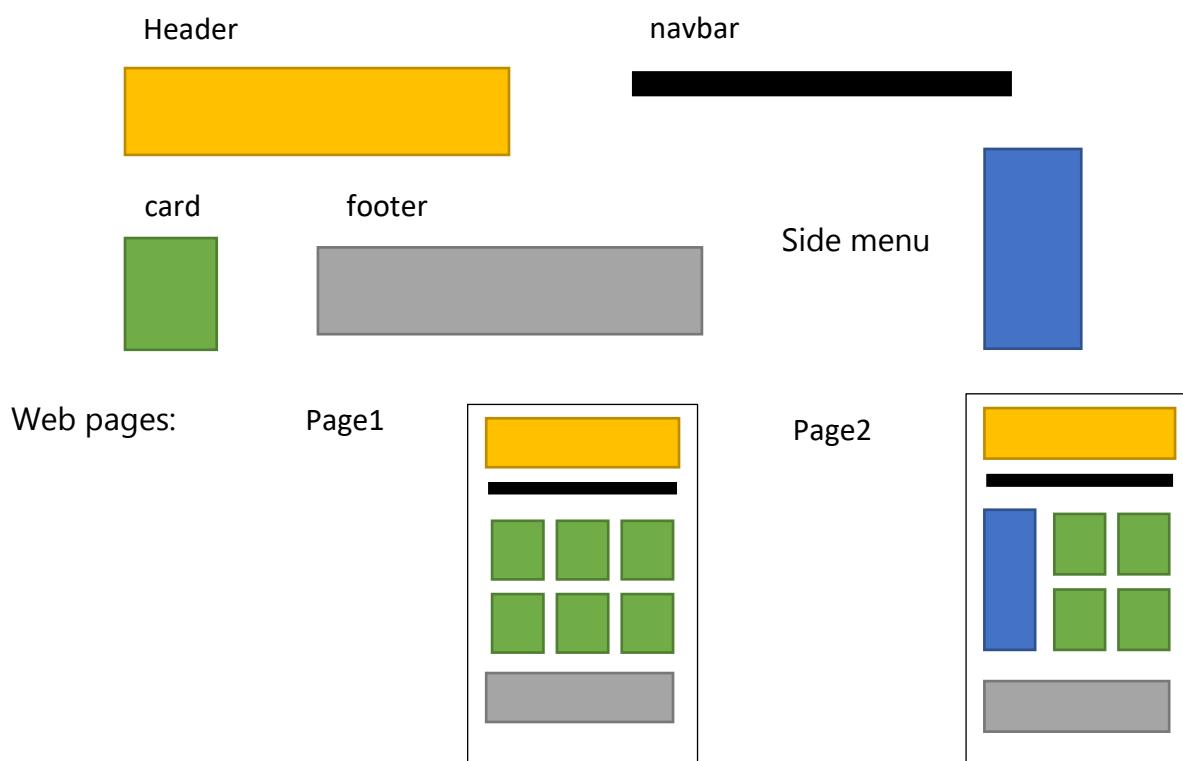
React JS ကို ဘာကြောင့်သုံးသင့်သလဲဆိုတဲ့ အကြောင်းအရင်းကတော့ -

- a) Component based ဖြစ်ခြင်း
 - b) App performance speed ပိုမိုမြန်ဆန်လာစေခြင်း
 - c) Code maintenance လုပ်ဖို့ ပိုမိုလွယ်ကူစေခြင်း
- တို့ကြောင့်ဖြစ်ပါတယ်။

a) Component based ဖြစ်ခြင်း

Web App တစ်ခု develop လုပ်တော့မယ်ဆိုရင် component အများကြီး ဆောက်ရပါမယ်။ Component တစ်ခုချင်းစီဟာ သူ့အစိတ်အပိုင်းနဲ့သူ independent ဖြစ်နေကြပြီး coding logic တွေကိုလည်း သူ့ component နဲ့သူ့ သီးသန့်ရေးရပါတယ်။ ဥပမာ Header component မှာ ရေးထားတဲ့ code logic တွေဟာ side menu component ထဲမှာ ရှိတဲ့ code နဲ့ design တွေအပေါ် effect သက်ရောက်မှုမရှိပဲ သီးသန့် လွှတ်လွှတ်လပ်လပ် ရေးသားနိုင်မှာ ဖြစ်ပါတယ်။

Components :



b) App Performance ပိုမိုမြန်ဆန်ခြင်း

Virtual DOM ကတေသာ React ရဲ့ အရေးပါဆုံး concept တစ်ခုလို့ ပြောလို့ရပါတယ်။ DOM ရဲ့အရှည်ကောက်က Document Object Model ဖြစ်ပါတယ်။ Real DOM က ပုံမှန်အားဖြင့် web page တစ်ခုရဲ့ actual HTML structure ကို ကိုယ်စားပြထားပါတယ်။ Developer ကရေးလိုက်တဲ့ code တွေကို web browser ပေါ်မှာ ထွက်လာအောင် render လုပ်တဲ့ အခါအချိန်ယူရပါတယ်။

React သည် update လုပ်ထားတဲ့ virtual DOM ကို real DOM နှင့် နိုင်းယှဉ်ပြီး real DOM ကို လျှော့ညီစွာ update ပြန်ပြန်လုပ်ပေးပါတယ်။ သည်လို update လုပ်ပေးတဲ့ ဖြစ်စဉ်ကို reconciliation လို့ ခေါ်ပါတယ်။ ဒီလို လုပ်ဆောင်ခြင်း နည်းစနစ်ကို Differing Algorithm ဒါမှမဟုတ် heuristic algorithm လို့ ခေါ်ပါတယ်။

Document Object Model (DOM)

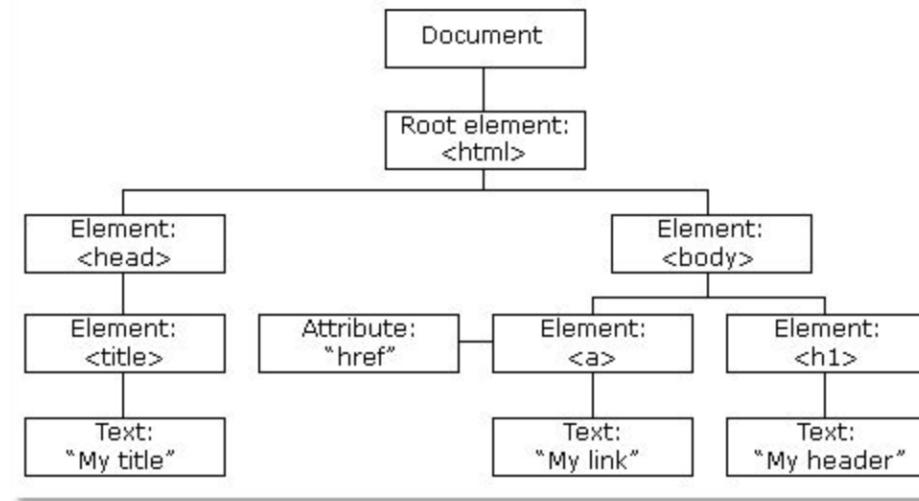


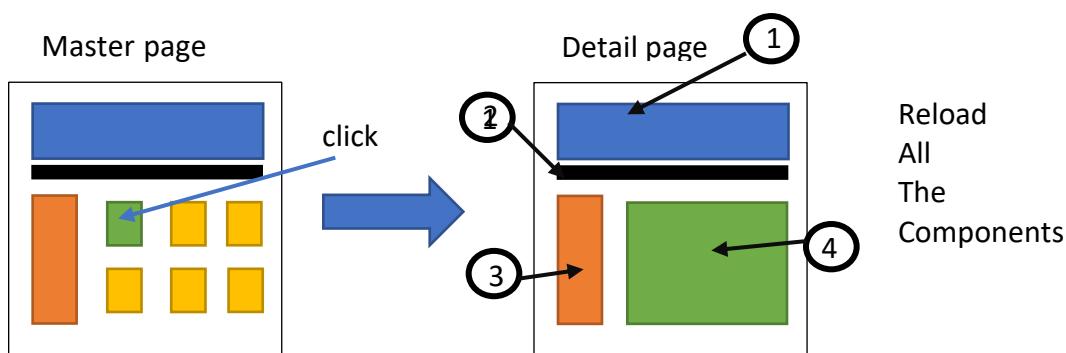
Fig: DOM of a Webpage

ပိုမိုမြင်သာအောင်ဖော်ပြရမယ်ဆိုရင် Developer က code ကို update လုပ်ပြီး Page တစ်ခုလုံးကို render ဆောင် ပြုလုပ်ပြီး updated output တွေကို ကြည့်မယ်ဆိုရင် စိတ်တိုင်းကျ result တစ်ခုရဖို့ဆိုရင် DOM manipulation က အရမ်းကို slow ဖြစ်သွားမယ်။ Memory လည်း waste ဖြစ်ပါမယ်။ ဒီ problem တွေကို ဖြေရှင်းဖို့ React က virtual DOM ဆိုတဲ့ နည်းစနစ်တစ်ခု

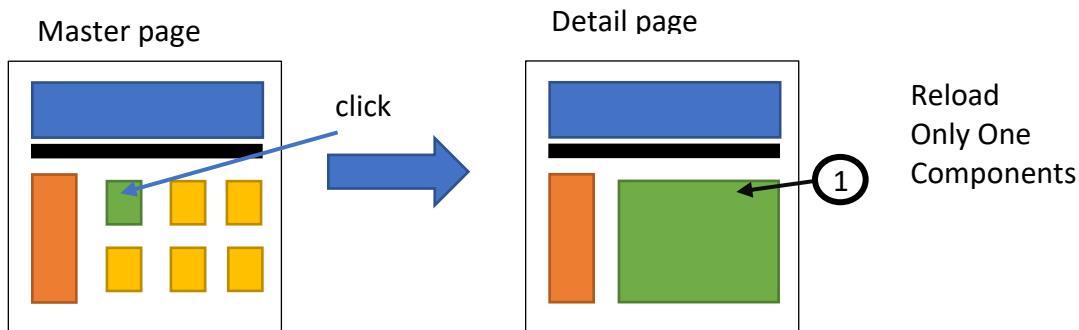
ဖန်တီးအသုံးပြုလာတာ ဖြစ်ပါတယ်။ Virtual DOM ကို အသုံးပြုခြင်းအားဖြင့် DOM manipulation က လွယ်သွားမယ်။ Page တစ်ခုလုံးကို ခဏေကာ render လုပ်စရာမလိုပဲ လိုအပ်တဲ့ component ကို ပဲ update လုပ်မယ်။ ပြီးတော့ render လုပ်မယ်ဆိုရင် memory waste မဖြစ်တော့ဘူးပေါ့။ Update process လည်းအရမ်းမြန်သွားပါမယ်။

ဥပမာ တစ်ခု ထပ် ပြောကြည့်ရအောင်။ သာမန် web app တွေမှာဆို web page တစ်ခုရဲ့အစိတ်အပိုင်း တစ်ခု show details button ကိုနိုင်လိုက်တယ်ဆိုပါစွာ။ Detail Page ကို load လုပ်တဲ့အခါ Detail Page မှာ ရှိသမျှ component တွေအားလုံးကို reload လုပ်ပေးပါမယ်။ အဲဒါကြောင့် process က နဲ့ slow ဖြစ်သွားပါတယ်။ React နဲ့ရေးထားတဲ့ app တွေမှာတော့ real DOM မှာရှိပြီးသား component အားလုံးကို reload လုပ်မှာ မဟုတ်ပါဘူး။ Updated component တစ်ခုထဲကို ပဲ reload လုပ်ပေးမှာမို့ application ကို သုံးတဲ့ user တွေအနေနဲ့ application performance speed ကို ကောင်းကောင်း ထိတွေ့ခံစားရမှာ ဖြစ်ပါတယ်။

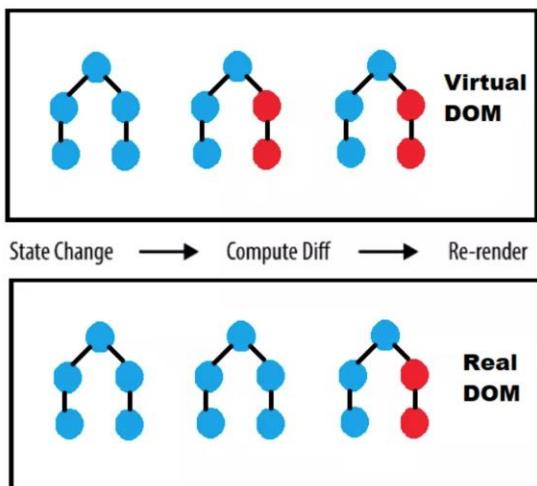
Normal Rendering



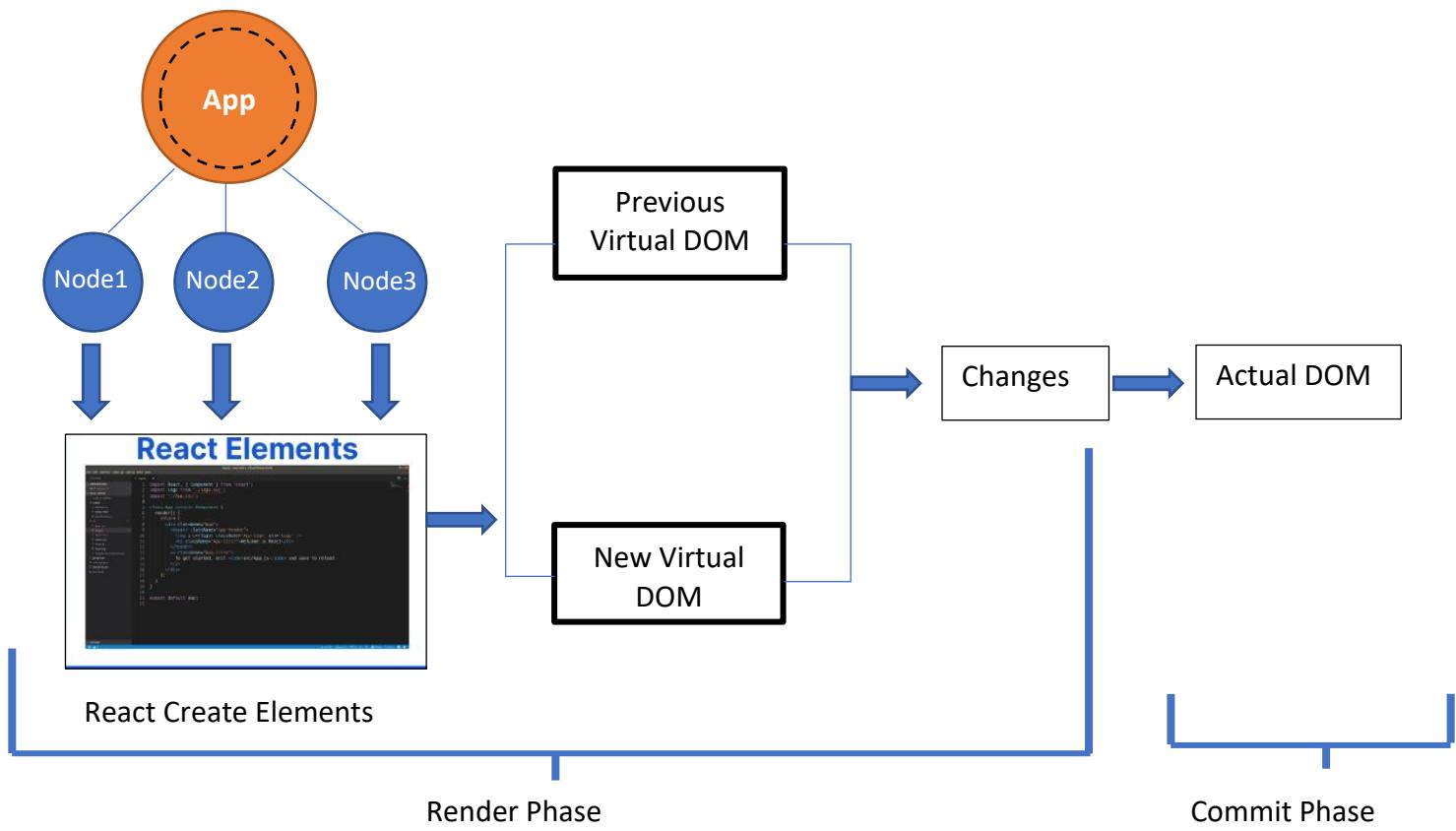
React Rendering



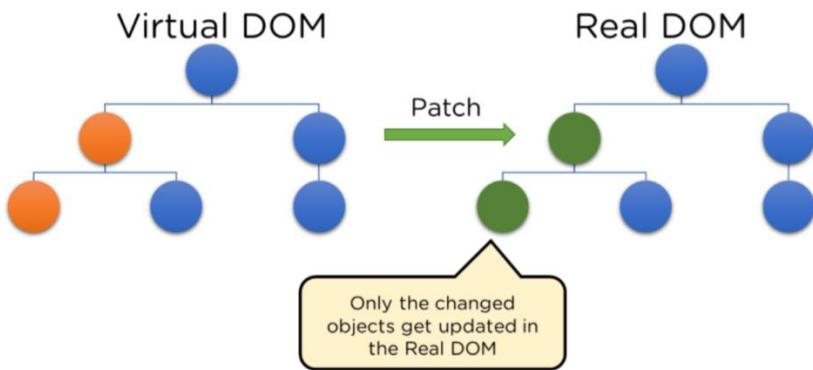
Virtual DOM & Real DOM



React Details Process

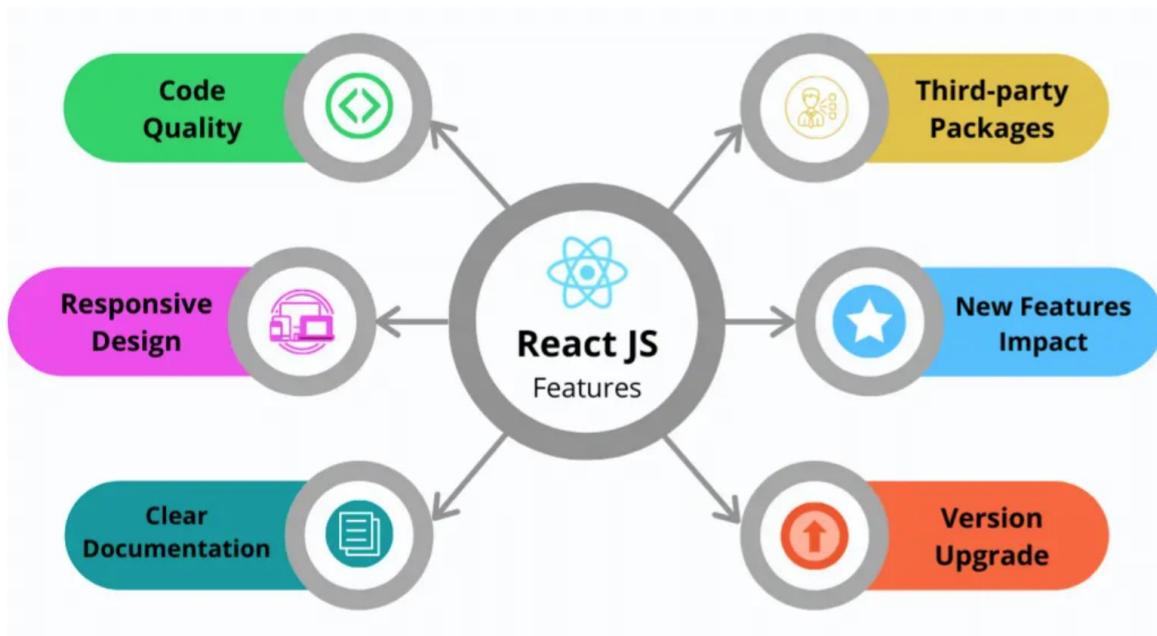


Virtual Document Object Model (DOM)



c) Code Maintenance ပိုမိုလွယ်ကူလာစေခြင်း

ရေးပြီးသား Component code တွေအားလုံးကို ကြိုက်တဲ့ application projects ။ ကြိုက်တဲ့ project စာမျက်နှာတွေမှာ ထပ်ခါထပ်ပါ ပြန်လည် အသုံးပြနိုင်တာ ဖြစ်တဲ့ အတွက် Quality Code တွေကို maintain လုပ်ရတာ အင်မတန်မှ အဆင်ပြေပါတယ်။ အချိန်နဲ့အမျှ Quality ရှိတဲ့ Code တွေကြောင့် application performance နဲ့ application quality တွေကို တိုက်ရှိက အကျိုးပြုမှာ ဖြစ်ပါတယ်။ App Component တွေအားလုံးက independently ရေးရတာ ဖြစ်တဲ့အတွက် ပြန်လည် refurbish လုပ်ရမယ့် အစိတ်အပိုင်းတွေ update လုပ်ရမယ့် အပိုင်းတွေကို အခြားသော application component အစိတ်အပိုင်းတွေကို ထိခိုက်မှုမရှိစေပေး သိုးသန့် ပြင်ဆင် ဆောင်ရွက်နိုင်ပါတယ်။ ပြီးတော့ new feature component ကို လည်း လွယ်ကူစွာ ထပ်ထည့် နိုင်ပါတယ်။



Installation and Setup

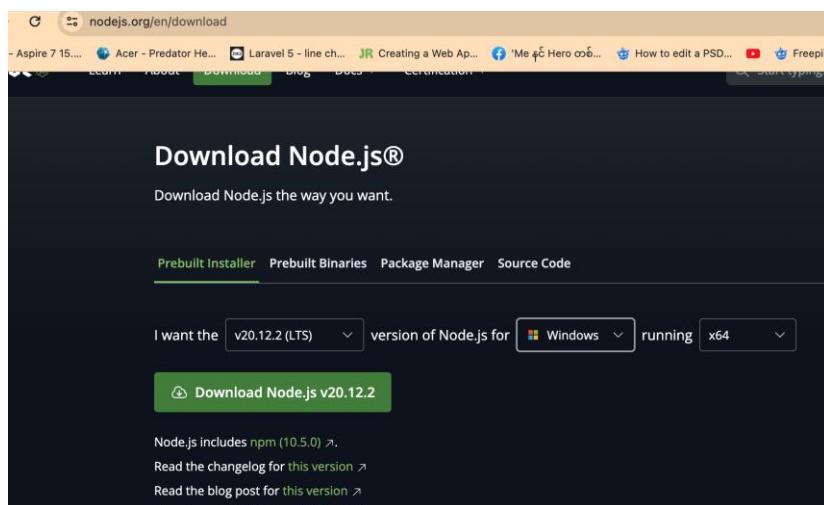
🎬 React Video Lesson 2



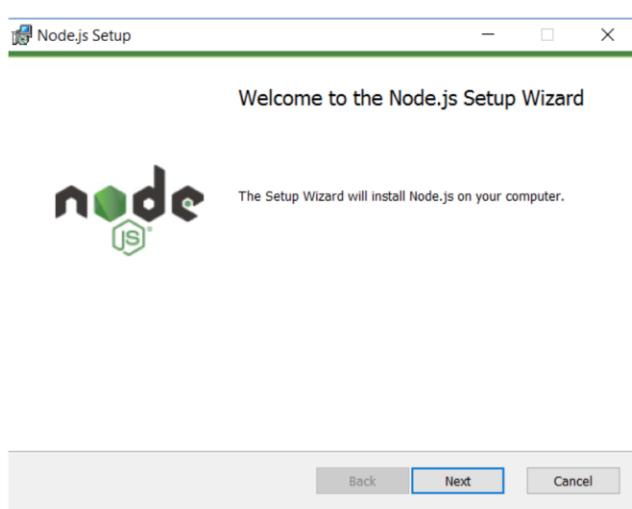
II. Installation and setup

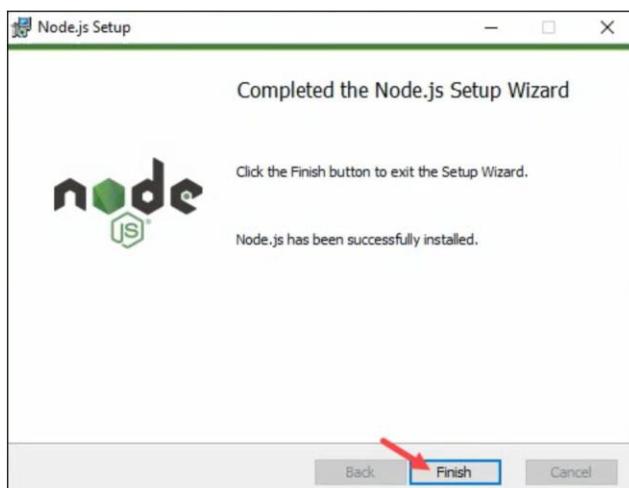
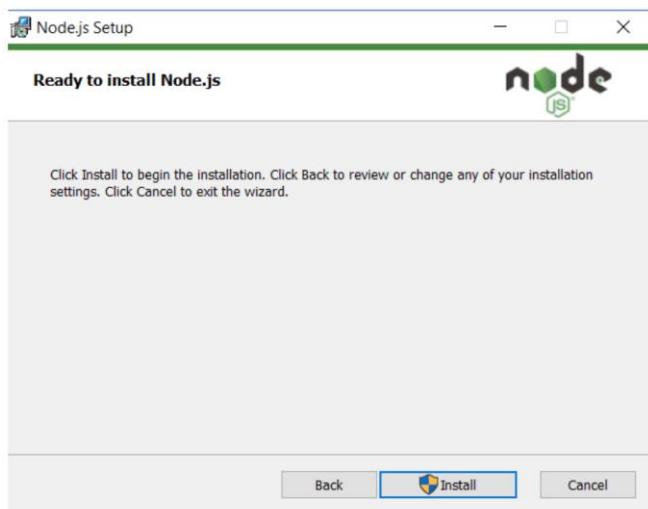
React application ကိုရေးဖို့ရာ ကိုယ့်ရဲ့ကွန်ပျူးတာမှာ Node JS ကို အရင် install လုပ်ရပါမယ်။ အောက်က download link ကို သွားပြီး Node JS ကို ဒေါင်းလုပ်ဆွဲရပါမယ်။ ကိုယ်အသုံးပြုနေတဲ့ ကွန်ပျူးတာမှာ Node JS install လုပ်ပြီးသားဆိုရင် တော့ ဒါ installation step ကိုကျော်လိုက်ပါ။

www.nodejs.org/en/download



Download ဆွဲပြီးသွားရင် Computer ရဲ့ download folder ကိုသွားပြီး Node JS setup file ကို download click ခေါက်ပြီး အောက်ပါအတိုင်း installation လုပ်ရပါမယ်။





Node JS installation ပြီးသွားရင် Node JS version နဲ့ npm version ကို command mode မှာ အောက်ပါအတိုင်း ကြည့်လို့ရပါတယ်။

```
Command Prompt
Microsoft Windows [Version 10.0.17758.1]
(c) 2018 Microsoft Corporation. All rights reserved.
```

A screenshot of a Microsoft Windows Command Prompt window. The title bar says "Command Prompt". The window shows the text "Microsoft Windows [Version 10.0.17758.1]" and "(c) 2018 Microsoft Corporation. All rights reserved.".

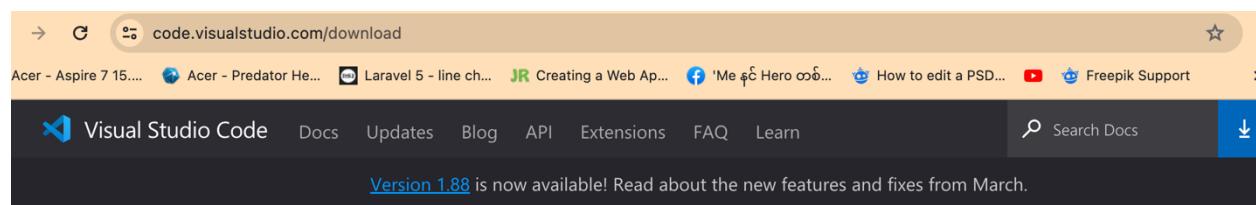
```
C:\Users\KB>node -v
v20.10.0

C:\Users\KB>npm -v
10.2.3

C:\Users\KB>
```

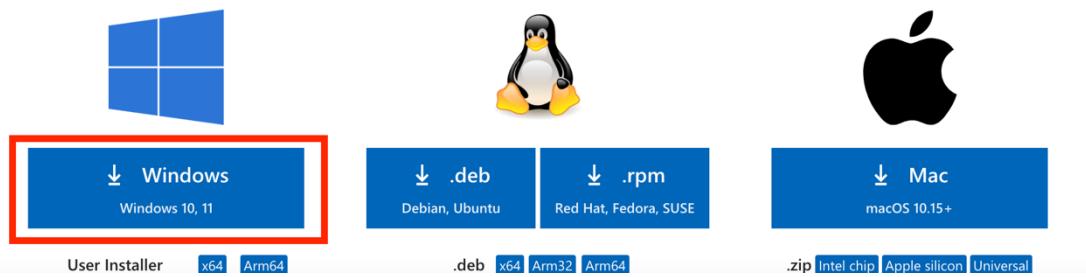
Node JS download လုပ်ပြီး installation ပြီးသွားရင် တော့ vs code ကို install လုပ်ရပါမယ်။

<https://code.visualstudio.com/download>

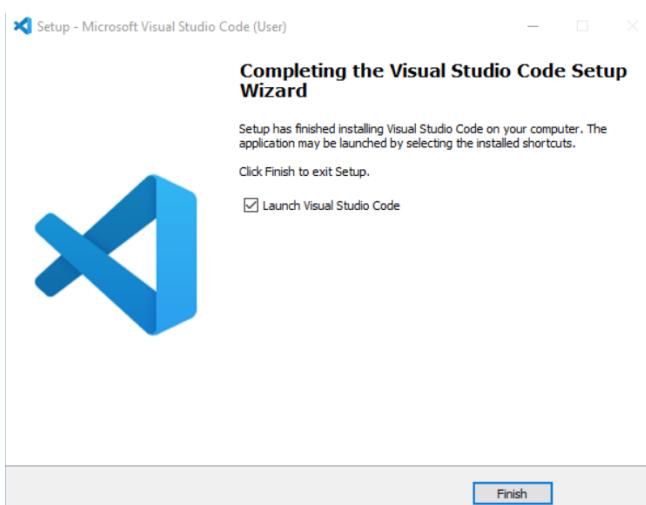
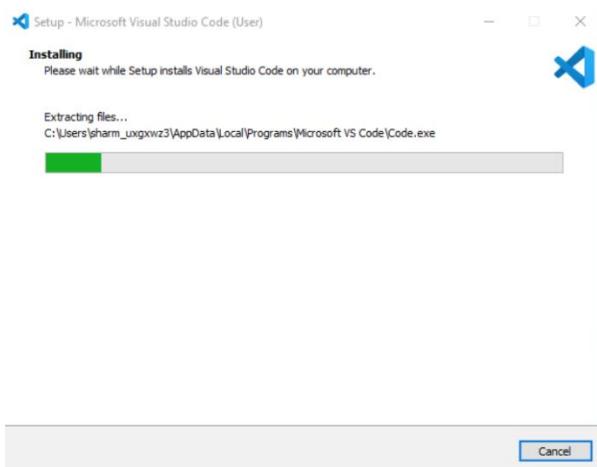
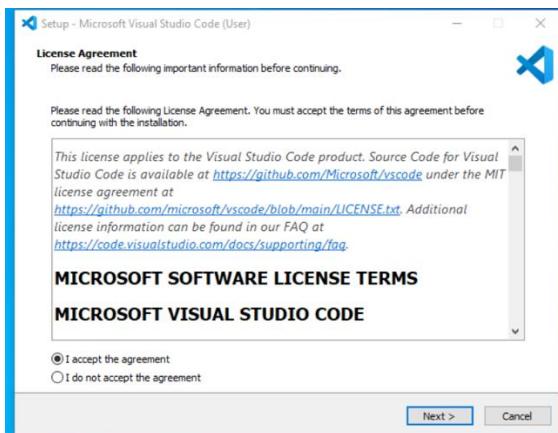


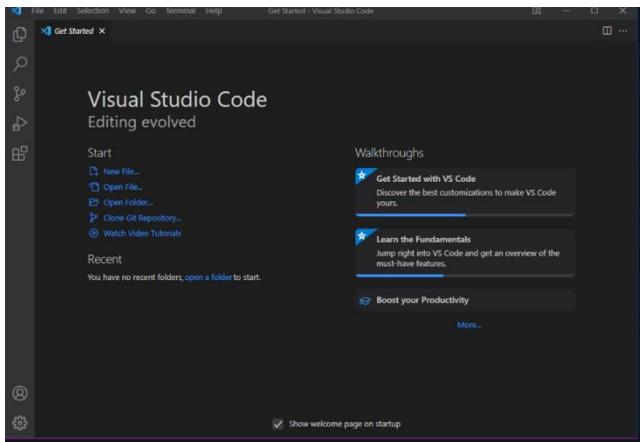
Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



VS Code download ဆဲပြီးသွားရင် အောက်ပါအတိုင်း install လုပ်ပါ။





လိုအပ်တဲ့ software Installation တွေပြီးသွားပြီဆိုရင်တော့ react application ကို စရေးလို ရပါပြီ။

III. Practical Lessons

Running the first application

React Environment setup ကို အောက်ပါအတိုင်း ၂ မျိုး တည်ဆောက်နိုင်ပါတယ်။

- i) Webpack နဲ့ babel သုံးပြီး Manual setup ပြုလုပ်နိုင်ပါတယ်။
- ii) create-react-app command နဲ့ auto setup ပြုလုပ်နိုင်ပါတယ်။

ဒီ စာအုပ်ထဲက lesson တွေမှာ တော့ create-react-app command ကို အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။
auto setup ဖြစ်တဲ့အလျောက် အချိန်ကုန်သက်သာပြီး configuration လုပ်ရတာ လွယ်ကူသွားပါမယ်။
ဒီ command ကို သုံးတော့မယ်ဆို Node JS ကို install လုပ်ထားဖို့တော့ လိုပါတယ်။

Command:

```
npx create-react-app my-react-app
```

```
[6] 4806
● htoinanbrang@htoinans-MacBook-Pro react % npx create-react-app my-react-app
Creating a new React app in /Users/htoinanbrang/Desktop/react/my-react-app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
added 1495 packages in 4m
258 packages are looking for funding
  run 'npm fund' for details
Initialized a git repository.

Installing template dependencies using npm...
npm WARN EBADENGINE Unsupported engine{
  name: '@esbuild/library/dom',
  required: { node: '^>=18' },
  current: { node: '^v16.14.2', npm: '8.5.0' }
}
npm WARN EBADENGINE
npm WARN EBADENGINE
added 67 packages in 18s
```

```
Inside that directory, you can run several commands:
  npm start
    Starts the development server.
  npm run build
    Bundles the app into static files for production.
  npm test
    Starts the test runner.
  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!
We suggest that you begin by typing:
  cd my-react-app
  npm start

Happy hacking!
npm notice
npm notice New major version of npm available! 8.5.0 -> 10.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.5.2
npm notice Run npm install -g npm@10.5.2 to update!
npm notice
htoinanbrang@htoinans-MacBook-Pro react %
```

Command:

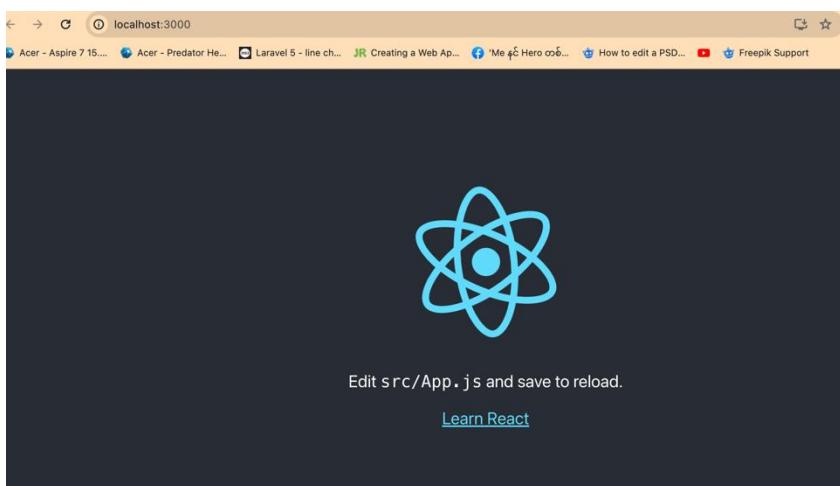
```
cd my-react-app
```

Command:

```
my-react-app> npm start
```

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project named 'my-react-app' under the 'REACT' category. The terminal tab is active, displaying the following output:

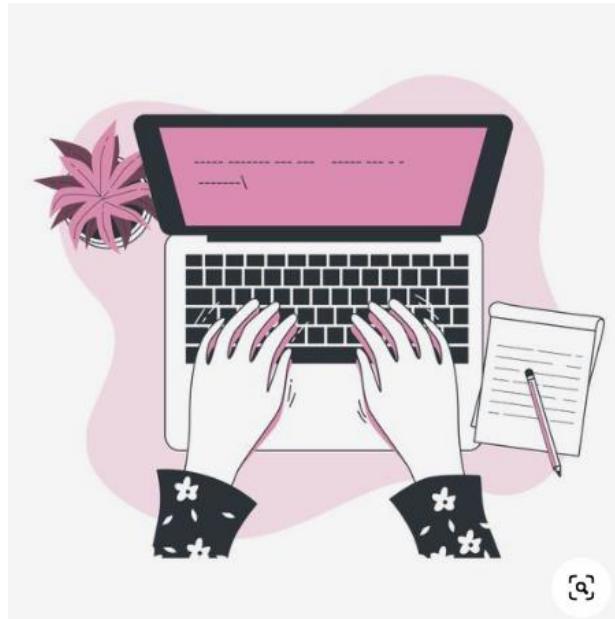
```
Compiled successfully!
You can now view my-react-app in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.100.41:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
```



Congratulation የ|| Running the first React Application አለንምኝች፡በታች፤ ዘዴገፍዎች፤ React Lesson ጥሩበት ተቻይሶች፤ ተቻይሶች፤ ስርዓት የሚያስፈልግ ይችላል፤

Discussion on Project structure

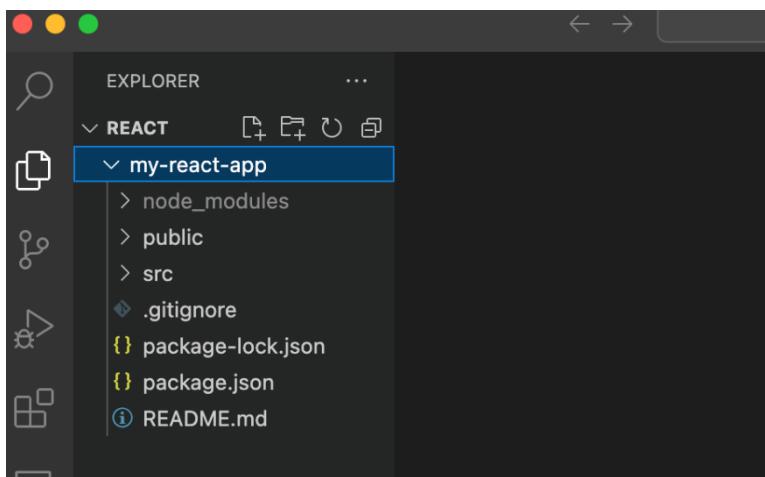
🎬 React Video Lesson 3



Project Structure

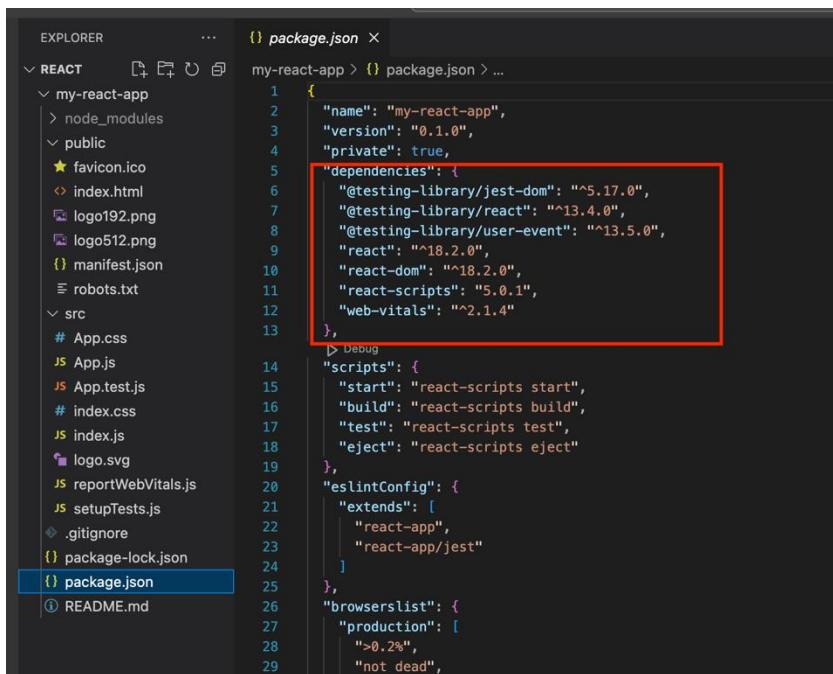
Project structure နဲ့ ပတ်သက်လို့ အောက်ပါအစိတ်အပိုင်းများကို အသေးစိတ်ဆွေးနွေးကြမှာ ဖြစ်ပါတယ်။ အဲဒီအစိတ်အပိုင်းတွေကတော့ -

- i) package.json file
- ii) node_modules folder
- iii) src folder
- iv) public folder စုံဖြစ်ပါတယ်။



- i) package.json file

ကျနော်တို့ရေးမယ့် application မှာ သုံးမယ့် dependency library စာရင်းကို ဒီဖိုင်မှာ ကြည့်ရှုနိုင်ပါတယ်။ ဒီ dependency library list ကို ကြည့်ပြီး လိုအပ်မယ့် library တွေထပ်ထည့်မယ်။ မလိုတော့တဲ့ library တွေကို ဖြုတ်ပလိုက်မယ်ပေါ့။



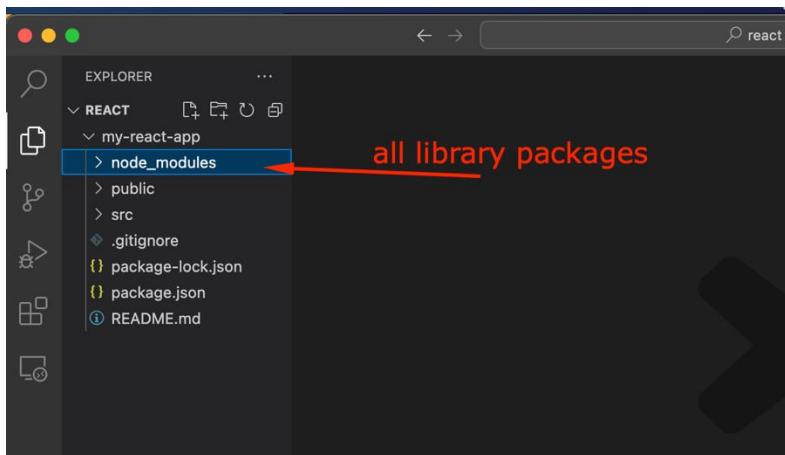
```

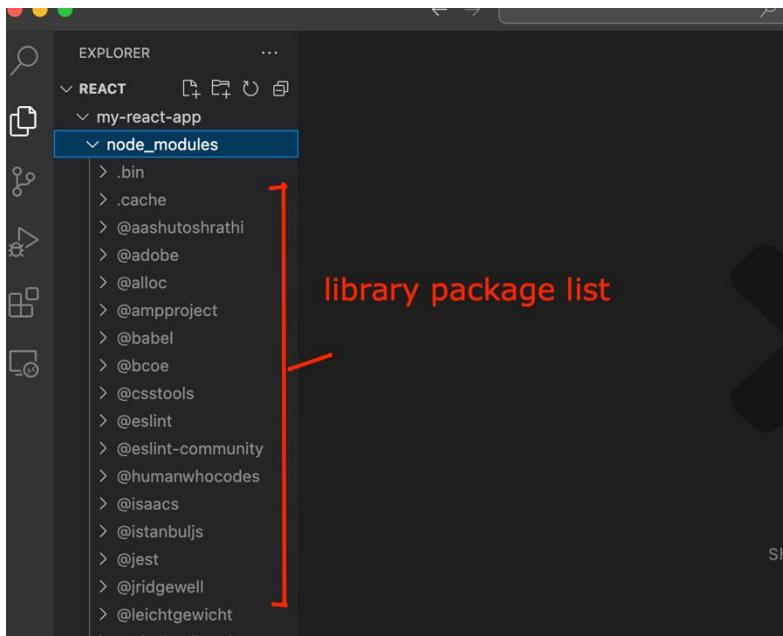
{
  "name": "my-react-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead"
    ]
  }
}

```

ii) node_modules folder

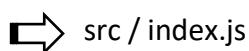
application project အတွက်လိုအပ်တဲ့ core library packages တောားလုံး ၃ folder အထဲမှာ ရှိပါတယ်။





iii) src folder

Application နှုပါတ်သက်တဲ့ logic function မှန်သမျှ ဒီ folder ထဲမှာပဲရေးကြမှာဖြစ်ပါတယ်။
များသောအားဖြင့် application development အချိန်တွေကို ဒီ src folder ထဲမှာပဲကုန်ဆုံးရမှာဖြစ်ပါတယ်။



src folder ထဲက index.js file ကတော့ react application တစ်ခုလုံးရဲ့ စတင်တဲ့ နေရာဖြစ်ပါတယ်။

Virtual DOM မှာရှိနေတဲ့ logical structure JavaScript elements တွေကို render လုပ်ပေးတဲ့ file ဖြစ်ပါတယ်။ ကျောင်းသားတွေ မကြာခဏ မေးနေကြ မေးခွန်းကတော့ src/index.js file နဲ့ public/index.html နဲ့ ဘယ်လို ဆက်စပ်မှုရှိသလဲဆိုတာပါပဲ။ ဖြေရှင်းချက်ကတော့ node_models folder ထဲက node_modules/react-scripts/config/paths.js file ကို သွားဖွင့်ကြည့်မယ်ဆိုရင်

`appHtml: resolveApp('public/index.html')` ဆိတဲ့ code ကို publicly accessible ဖြစ်အောင် သတ်မှတ်ထားတာ တွေပါမယ်။

ဒါ code လေးက ကျနော်တို့ react application ကို စု run လိုက်တာနဲ့ entry point ဖြစ်တဲ့ index.js ဖိုင်က စပြီးအလုပ်လုပ်ပါတယ်။ index.js ဖိုင်က index.html ဖိုင်ကို ရှာပြီး output တွေကို index.html file မှာ render လုပ်ပေးပါတယ်။ node_modules package ထမာ config folder အထဲက path.js file က index.html file ကို application setup ထမာ publicly accessible ဖြစ်အောင် သတ်မှတ်ရေးသားထဲ့အတွက် src/index.js file နဲ့ public/index.html file တွေက ဆက်စပ်ပြီး အလုပ်လုပ် နေရခြင်းဖြစ်ပါတယ်။

➡ src/index.js

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure under "REACT" (my-react-app). The "src" folder is expanded, showing files like App.css, App.js, App.test.js, index.css, and index.js. The "index.js" file is currently selected and highlighted with a blue border.
- CODE EDITOR:** The tab bar shows "JS index.js". The code editor displays the contents of index.js:

```

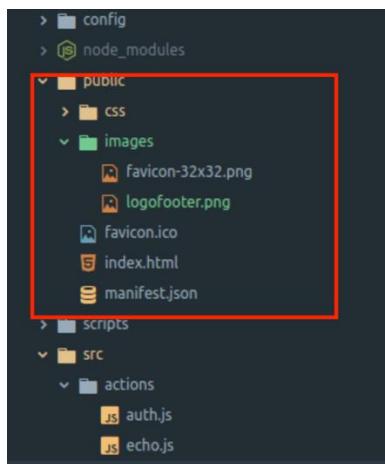
my-react-app > src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10  |   <App />
11  </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18

```

- SEARCH:** A search bar at the top right contains the text "react".

iv) public folder

public folder ထဲမှာတော့ static file ပေါ် images ပေါ် CSS file တွေနဲ့ အခြားသော asset file တွေကို ထည့်ဖို့ဖြစ်ပါတယ်။



public/index.html (default structure)

React application တစ်ခုလုံးရဲ့ functional component တွေအာလုံးကို index.html file ထဲက root structure block ထဲမှာ index.js file ထဲက JavaScript ReactDOM code နဲ့ render လုပ်ပါတယ်။

How React render HTML

Index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Index.html

```
<div id="root">
</div>
```

ဒီနေရာမှာ ReactDOM အကြောင်းနည်းပြောခြင်ပါတယ်။ Application project ဘယ်လောက်ပဲ
ကြိုးကြိုး၊ ဘယ်လောက်ပဲ သေးသေး

```

<html lang="en">
<head>
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
        Notice the use of %PUBLIC_URL% in the tags above.
        It will be replaced with the URL of the 'public' folder during the build.
        Only files inside the 'public' folder can be referenced from the HTML.

        Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
        work correctly both with client-side routing and a non-root public URL.
        Learn how to configure a non-root public URL by running 'npm run build'.
    -->
    <title>React App</title>
</head>
<body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
        This HTML file is a template.
        If you open it directly in the browser, you will see an empty page.

        You can add webfonts, meta tags, or analytics to this file.
        The build step will place the bundled scripts into the <body> tag.

        To begin the development, run 'npm start' or 'yarn start'.
        To create a production bundle, use 'npm run build' or 'yarn build'.
    -->
</body>
</html>

```

File Structure & Code Optimization

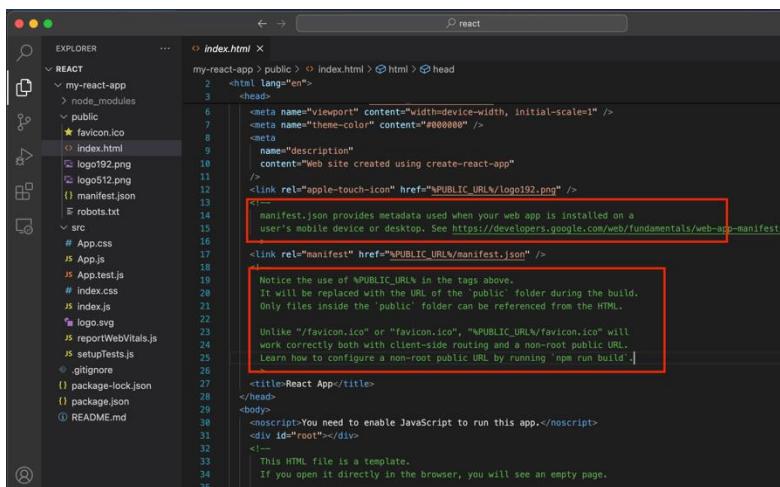
🎬 React Video Lesson 4



File Structure & Code Optimization

React Application တစ်ခုရဲ့ file structure နဲ့ function relationships တွေကို ကောင်းမျိုးစွာ လည်နှင့်အောင် file structure နဲ့ code တွေကို cleaning လုပ်ပါမယ်။ ရှိုးရှင်း ပြီး ဘာမှ မရှိတဲ့ project အနေအထားတစ်ခုကနေ complex project အနေအထားတစ်ခုသို့ရောက်အောင် တစ်ဆင့်ပြီးတစ်ဆင့် လေ့လာကြမှာ ဖြစ်ပါတယ်။

ဦးစွာ index.html file ထဲက မလိုအပ်တဲ့ comment အပိုင်းတွေကို ဖျက်လိုက်ပါမယ်။



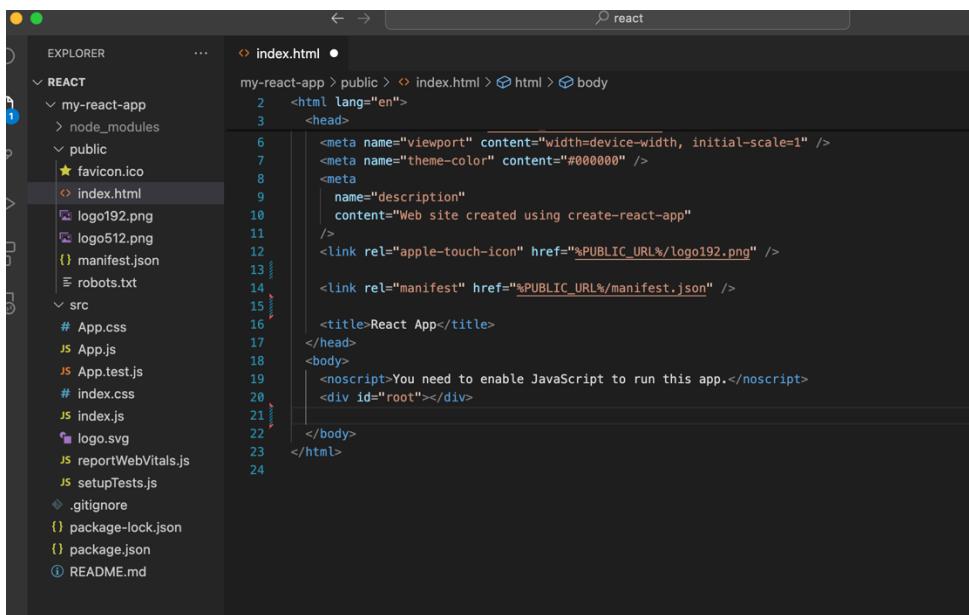
The screenshot shows the VS Code interface with the index.html file open. A red box highlights a block of comments at the top of the file:

```

<!--
  Notice the use of "%PUBLIC_URL%" in the tags above.
  It will be replaced with the URL of the "public" folder during the build.
  Only files inside the "public" folder can be referenced from the HTML.

  Unlike "favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
  work correctly both with client-side routing and a non-root public URL.
  Learn how to configure a non-root public URL by running 'npm run build'.
-->

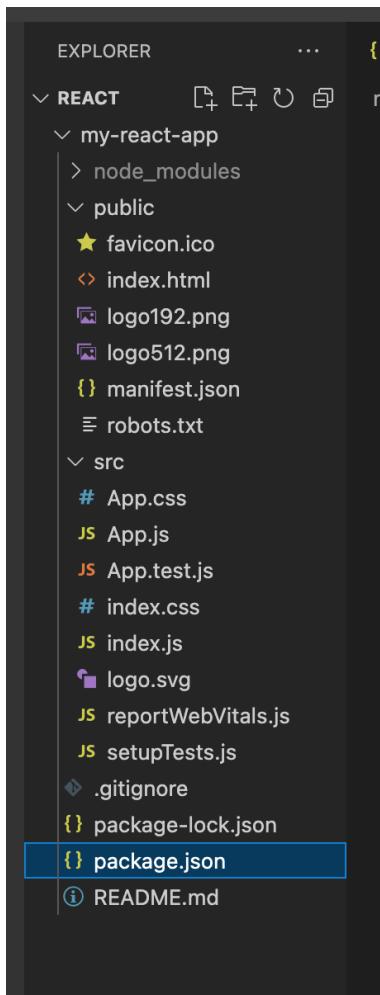
```

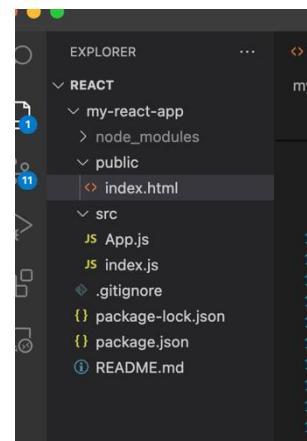
The screenshot shows the VS Code interface with the index.html file open. The previously highlighted comments have been removed, resulting in a cleaner file structure.

မလိုအပ်တဲ့ ဖိုင်တွေကို လည်း delete လုပ်လိုက်ပါ။ အောက်ပါအတိုင်း public folder ထဲမှာ index.html file ပဲ ရှိနေမယ်။ src folder ထဲမှာ app.js file နဲ့ index.js file ပဲ ကျွန်ုပ်မယ်။

Before deleting files



After deleting files



Index.js file ကို လည်း အောက်ပါအတိုင်း code cleaning လုပ်ပါမယ်။ React ဘယ်လိုအလုပ်လုပ်တယ်ဆိုတာကို ပိုမိုနားလည်အောင် file ထွေ့၊ code ထွေ့ကို အခဲလို ရှင်းလင်းရတာဖြစ်ပါတယ်။

Before deleting codes

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```



After deleting codes

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

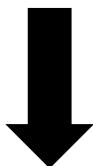
 App.js

Before deleting codes

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```



After deleting codes

```
function App() {
  return (
    <h1> React App </h1>
  );
}

export default App;
```

Default Code တွေကို update လုပ်ပြီးရင် အောက်ပါအတိုင်း ပြန် run ကြည့်ပါ။

```
cd my-react-app
```

```
my-react-app> npm start
```

Compiled successfully!

You can now view my-react-app in the browser.

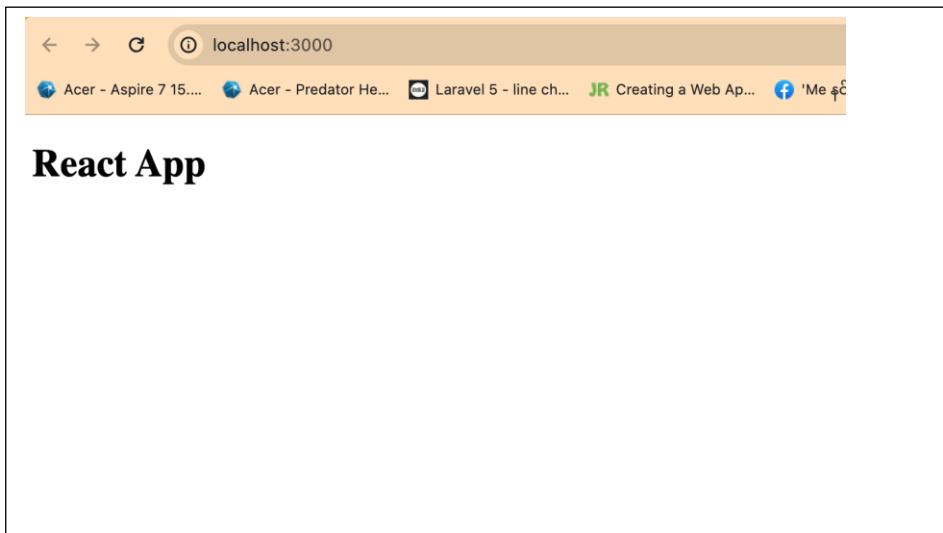
Local: http://localhost:3000

On Your Network: http://192.168.100.41:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

➡ Output:



Congratulation එකිනෙ File structure සහ code optimization සංස්කරණයට පෙන්වා ඇතුළුව නොවේ
React Component මෙහෙයුම් තොගී ලදු ලාභපිශිතයි।

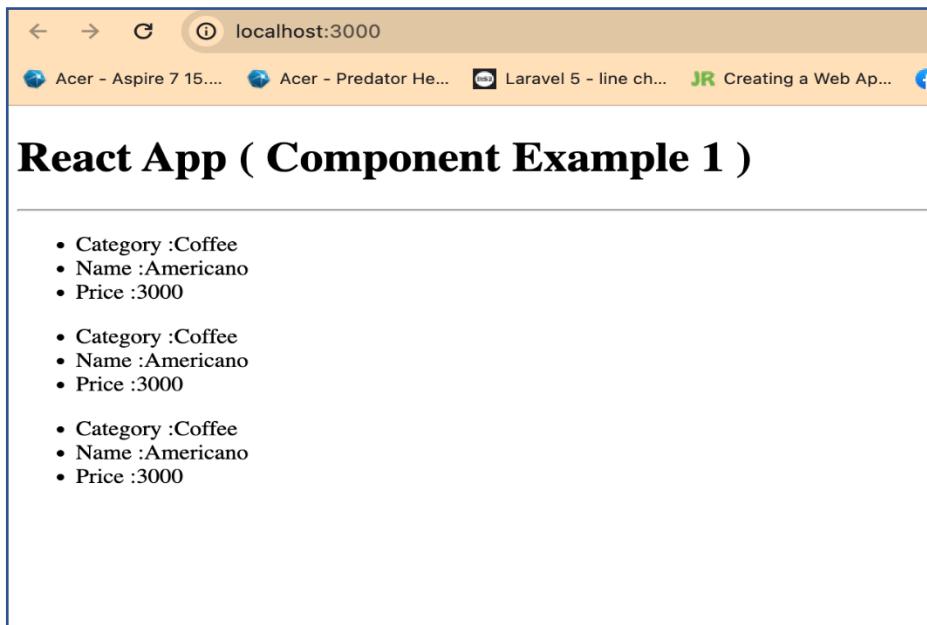
React Components

🎬 React Video Lesson 5



Components

➡ Example 1



➡ src/app.js

```
const Item=()=>{
  return (
    <ul>
      <li>Category :Coffee </li>
      <li>Name :Americano </li>
      <li>Price :3000 </li>
    </ul>
  );
}

const App=()=>{
  return (
    <>
      <h1>React App ( Component Example 1 )</h1>
      <hr/>
      <Item/>
    </>
  );
}
```

```
<Item/>
<Item/>
</>

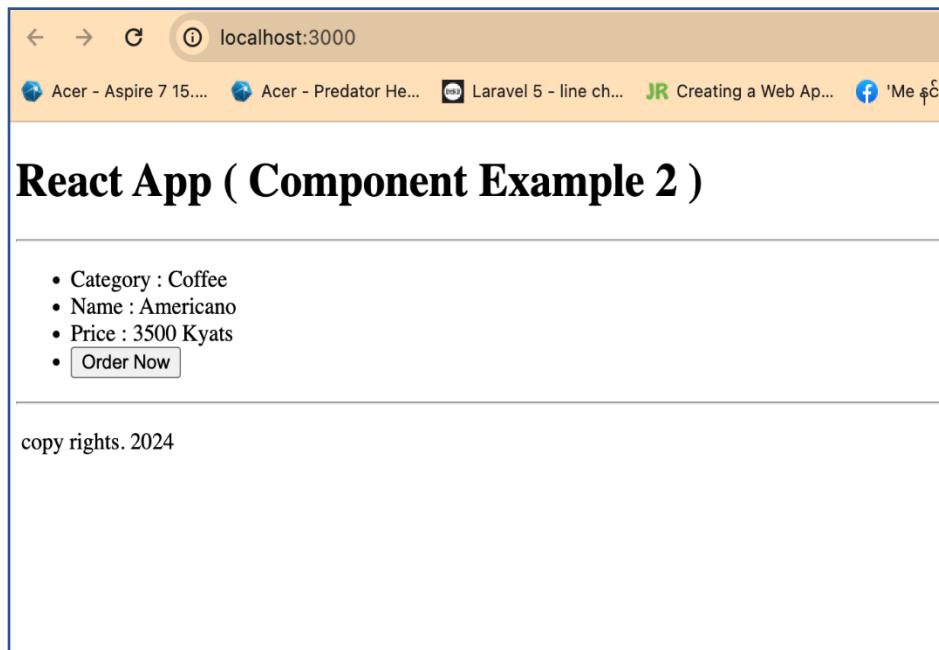
};

export default App;
```

Example 2

```
- src
  - Header.jsx
  - Item.jsx
  - Footer.jsx
  - App.js
```

Output:



» App.js

```
import Header from './Header'  
import Item from './Item'  
import Footer from './Footer'  
  
const App=()=>{  
  return (  
    <>  
    <Header/>  
    <hr/>  
    <Item/>  
    <hr/>  
    <Footer/>  
  
    </>  
  );  
}  
  
export default App;
```

» Header.jsx

```
const header=()=>{  
  return (  
    <h1>React App ( Component Example 2 ) </h1>  
  );  
}  
export default header;
```

» Item.jsx

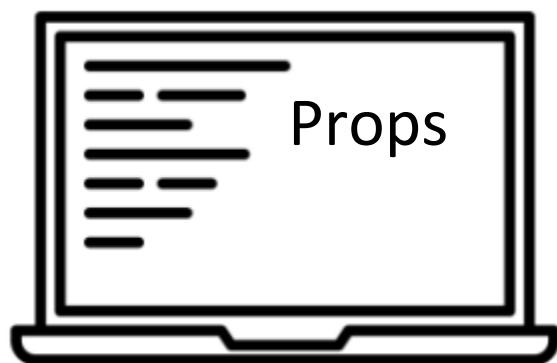
```
const Item=()=>{
  return (
    <>
    <ul>
      <li> Category : Coffee </li>
      <li> Name : Americano </li>
      <li> Price : 3500 Kyats </li>
      <li> <button>Order Now</button> </li>
    </ul>
    </>
  );
}
export default Item;
```

» Footer.jsx

```
const Item=()=>{
  return (
    <>
    <ul>
      <li> Category : Coffee </li>
      <li> Name : Americano </li>
      <li> Price : 3500 Kyats </li>
      <li> <button>Order Now</button> </li>
    </ul>
    </>
  );
}
export default Item;
```

Props

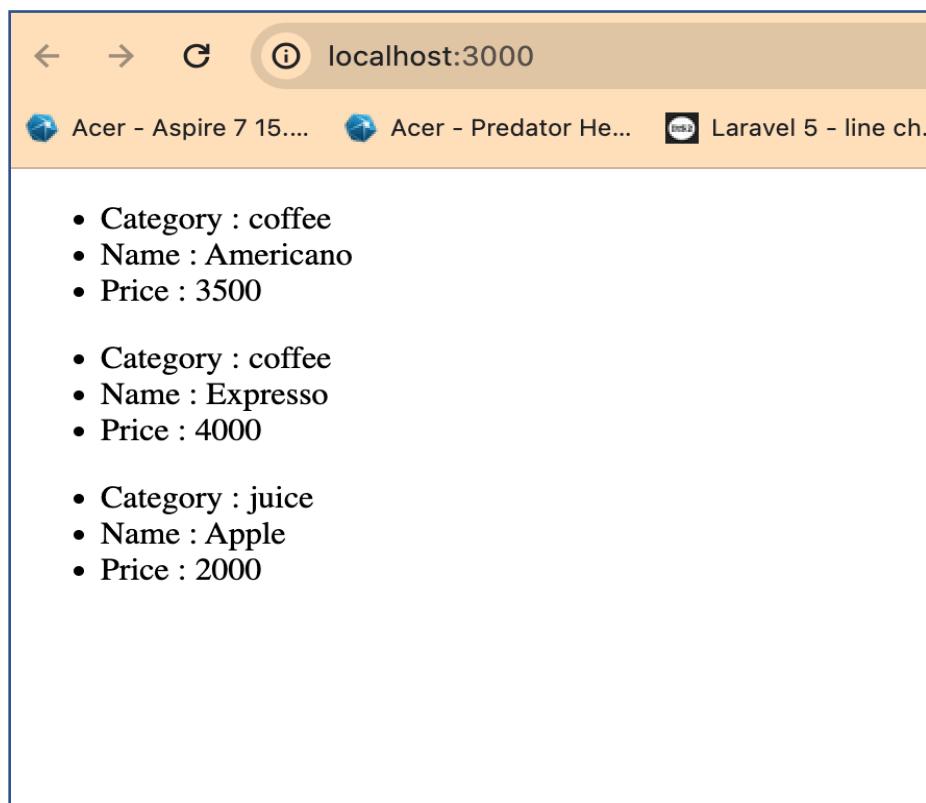
🎬 React Video Lesson 6



Props

➡ Example 1

➡ Output:





```
const Item=(props)=>{
  return(
    <>
    <ul>
      <li> Category : {props.category} </li>
      <li> Name : {props.name} </li>
      <li> Price : {props.price} </li>
    </ul>
    </>
  );
}

const App=()=>{
  return (
    <>
    <Item category='coffee' name='Americano' price={3500} />
    <Item category='coffee' name='Expresso' price={4000} />
    <Item category='juice' name='Apple' price={2000} />
    </>

  );
}

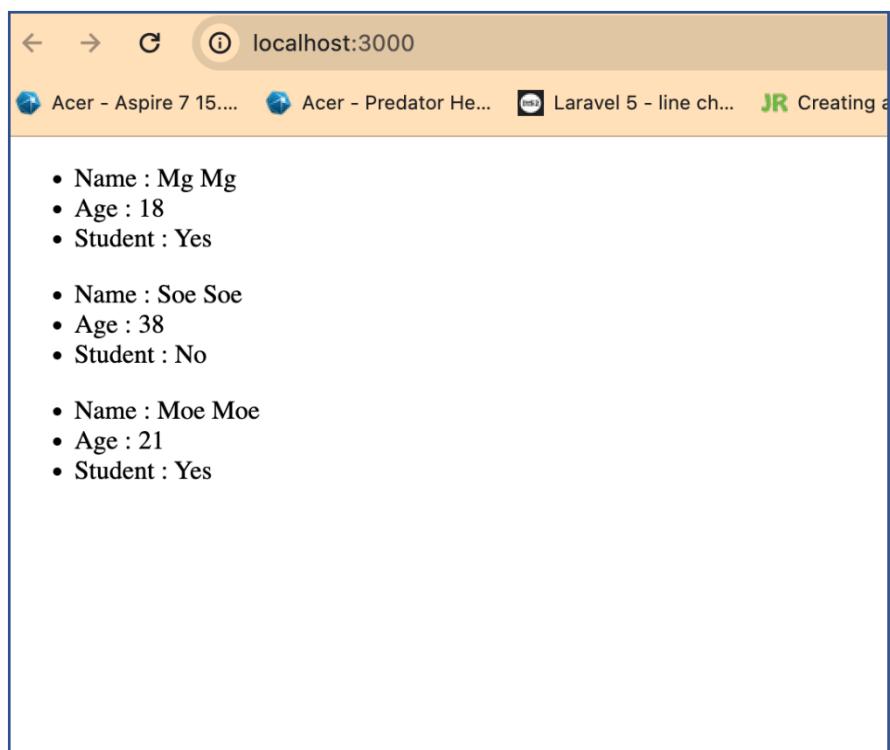
export default App;
```

➡ Example 2

Project files

```
- src
  - - App.js
  - - Student.jsx
```

➡ Output:



A screenshot of a web browser window titled "localhost:3000". The browser has several tabs open, including "Acer - Aspire 7 15....", "Acer - Predator He...", "Laravel 5 - line ch...", and "JR Creating a". The main content area displays three bullet-pointed lists, each representing a student's information:

- Name : Mg Mg
- Age : 18
- Student : Yes

- Name : Soe Soe
- Age : 38
- Student : No

- Name : Moe Moe
- Age : 21
- Student : Yes

💻 Codes:

➡ App.js.

```
import Student from './Student'

const App=()=>{
  return (
    <>
    <Student name={'Mg Mg'} age={18} isStudent={true} />
    <Student name={'Soe Soe'} age={38} isStudent={false} />
    <Student name={'Moe Moe'} age={21} isStudent={true} />
    </>

  );
}

export default App
```

➡ Student.jsx

```
const Student=(props)=>{
  return(
    <>
    <ul>
      <li> Name : {props.name} </li>
      <li> Age : {props.age} </li>
      <li> Student : {props.isStudent ? "Yes" : "No"} </li>
    </ul>
    </>
  );
}

export default Student
```

Conditional Rendering

🎬 React Video Lesson 7

CON ? T : F

Conditional Rendering

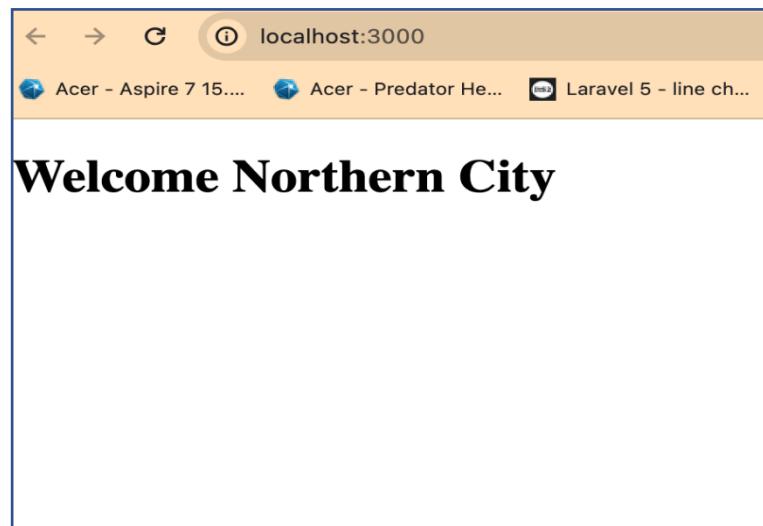
Example 1

Project files

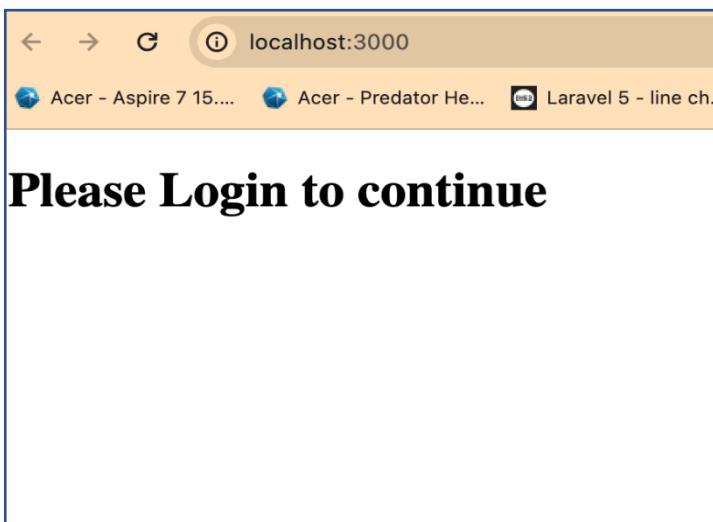
```
- src
  - - App.js
  - - Greetings.jsx
```

If isLoggedIn is true

Output:



If isLoggedIn is false
→ Output:



Source Codes:

» App.js

```
import Greetings from './Greetings'

const App=()=>{
  const login=false;
  const username="Northern City"
  return (
    <>
    <Greetings isLoggedIn={login} name={username} />
    </>

  );
}

export default App
```

» Greetings.jsx

```
const Greeting=(props)=>{  
  
    if(props.isLogin)  
        return <h1> Welcome {props.name} </h1>  
  
    return <h1> Please Login to continue </h1>  
}  
  
export default Greeting
```

➡ Update 1

```
const Greeting=(props)=>{  
  
    const welcome_prompt=<h1> Welcome {props.name} </h1>  
    const login_prompt=<h1> Please Login to continue </h1>  
  
    return (props.isLogin ? welcome_prompt : login_prompt);  
  
}  
  
export default Greeting
```



Update 2

```
import PropTypes from 'prop-types'

const Greeting=(props)=>{

    const welcome_prompt=<h1> Welcome
{props.username} </h1>
    const login_prompt=<h1> Please Login to continue
</h1>

    return (props.isLogin ? welcome_prompt :
login_prompt);

}

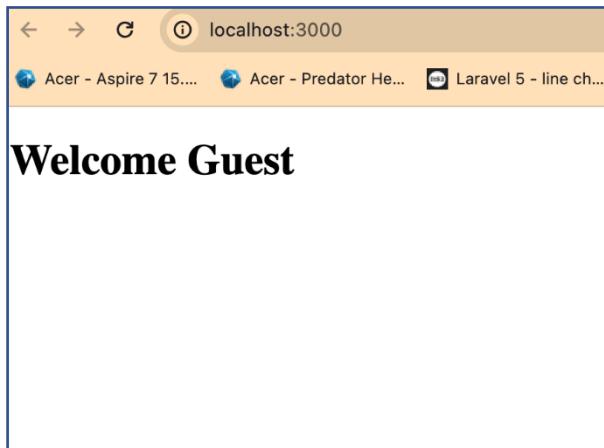
Greeting.propTypes={
    isLogin : PropTypes.bool,
    username: PropTypes.string,
}

Greeting.defaultProps={
    isLogin:false,
    username:"Guest",
}

export default Greeting
```



Updated Output:



 Notes:

Application တွေကြီးထွားလာသည့်နှင့်အမျှ typechecking နဲ့ bug တွေကို ကျဖော်တို့ ရှင်းလင်းကြပါတယ်။ အချို့သော application တွေမှာ type checking ပြုလုပ်ရန် Flow သို့မဟုတ် TypeScript ကဲ့သို့သော JavaScript extension များကို အသုံးပြုကြတယ်။ ဒါပေမယ့် အဲဒါတွေကို အသုံးမပြုရင်တောင် React မှာ build-in typechecking ရှိပါတယ်။ အစိတ်အပိုင်းတစ်ခုအတွက် props များပေါ်မှာ typechecking လုပ်ဆောင်ဖို့ propTypes ကို အသုံးပြုကြခြင်းဖြစ်ပါတယ်။

React Component default values တွေကို ကြိုတင် သတ်မှတ်ချင်တဲ့အတွက် defaultProps method ကို အသုံးပြုထားတာ ဖြစ်ပါတယ်။

List

🎬 React Video Lesson 8



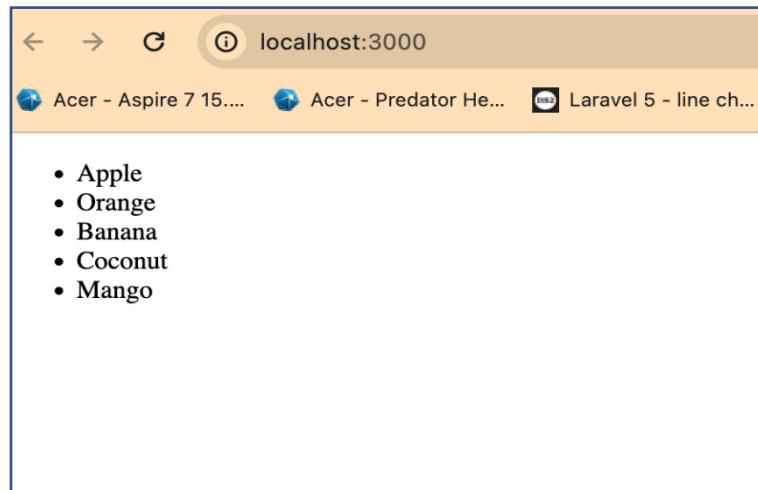
List

Example 1.

Project files

```
- src
  - - App.js
  - - List.jsx
```

Sample Output



» App.js

```
import List from './List'

const App=()=>{

  return (
    <>
    <List />
    </>

  );
}

export default App
```

» List.jsx

```
const List=()=>{
  const fruits=["Apple","Orange","Banana","Coconut","Mango"];

  const list_items=fruits.map((fruit)=><li>{fruit}</li>)

  return (<ul> {list_items} </ul>);

};

export default List
```

» List.jsx (updated)

```
const List=()=>{
  const fruits=["Apple","Orange","Banana","Coconut","Mango"];
  fruits.sort() // or fruit.reverse()

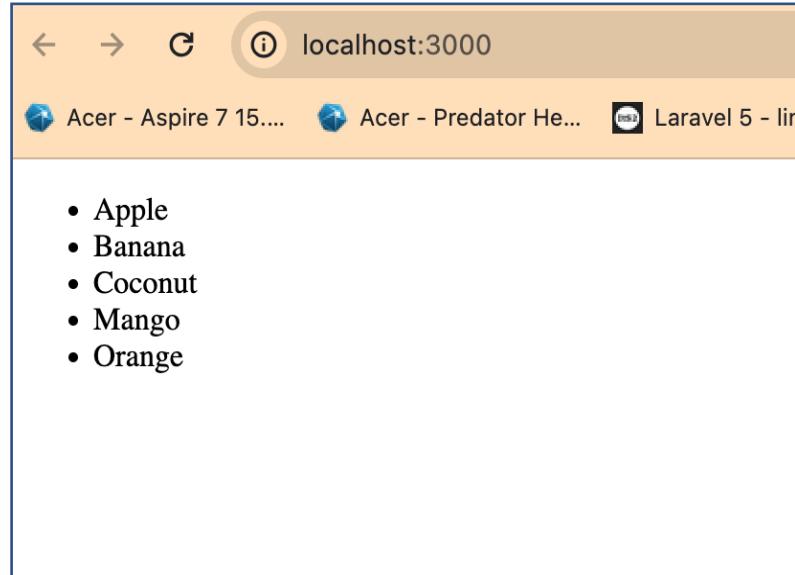
  const list_items=fruits.map((fruit)=><li>{fruit}</li>)

  return (<ul> {list_items} </ul>);

};

export default List
```

» Sorted output:

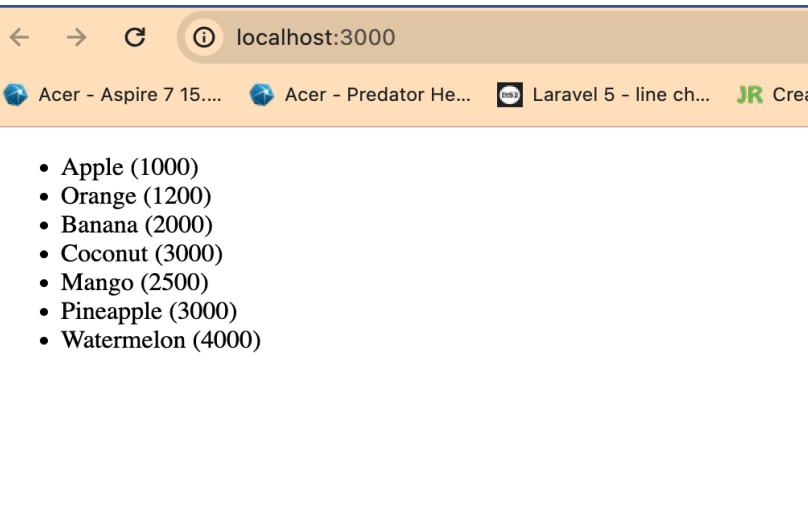


➡ Example 2.

Project Files

```
- src
  - - App.js
  - - List.jsx
```

➡ Sample Output:



» App.js

```
import List from './List'

const App=()=>{

  return (
    <>
    <List />
    </>

  );
}

export default App
```

>> List.jsx

```
const List=()=>{
  const fruits=[ {"id":1,"name":"Apple","price":1000},
    {"id":2,"name":"Orange","price":1200},
    {"id":3,"name":"Banana","price":2000},
    {"id":4,"name":"Coconut","price":3000},
    {"id":5,"name":"Mango","price":2500},
    {"id":6,"name":"Pineapple","price":3000},
    {"id":7,"name":"Watermelon","price":4000}
  ];
  //display all fruits
  const list_items=fruits.map((fruit)=><li key={fruit.id}>{fruit.name}
  {fruit.price}</li>)

  return (<ul> {list_items} </ul>);

};

export default List
```

>> List.jsx
 (Sort by Name)

```
const List=()=>{
  const fruits=[ {"id":1,"name":"Apple","price":1000},
    {"id":2,"name":"Orange","price":1200},
    {"id":3,"name":"Banana","price":2000},
    {"id":4,"name":"Coconut","price":3000},
    {"id":5,"name":"Mango","price":2500},
    {"id":6,"name":"Pineapple","price":3000},
    {"id":7,"name":"Watermelon","price":4000}
  ];

  //Alphabetical Sort
  fruits.sort((a,b)=>a.name.localeCompare(b.name))

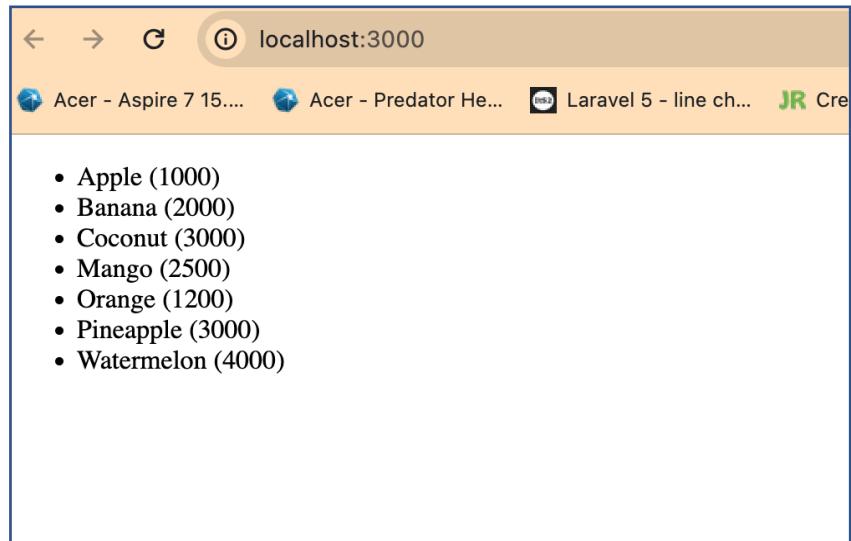
  //display all friuts
  const list_items=fruits.map((fruit)=><li
key={fruit.id}>{fruit.name} ({fruit.price})</li>

  return (<ul> {list_items} </ul>);

};

export default List
```

Output:



>> List.jsx
(Try all sorting)

```
//Alphabetical Sort
//fruits.sort((a,b)=>a.name.localeCompare(b.name))

//Alphabetical Reverse Sort
//fruits.sort((a,b)=>b.name.localeCompare(a.name))

//Numeric Sort by id
//fruits.sort((a,b)=>a.id - b.id)

//Numeric Sort by id reverse
//fruits.sort((a,b)=>b.id - a.id)

//Expensive Fruits (price is greater than 2000)
//const expensiveFruits=fruits.filter((f)=>f.price>2000)

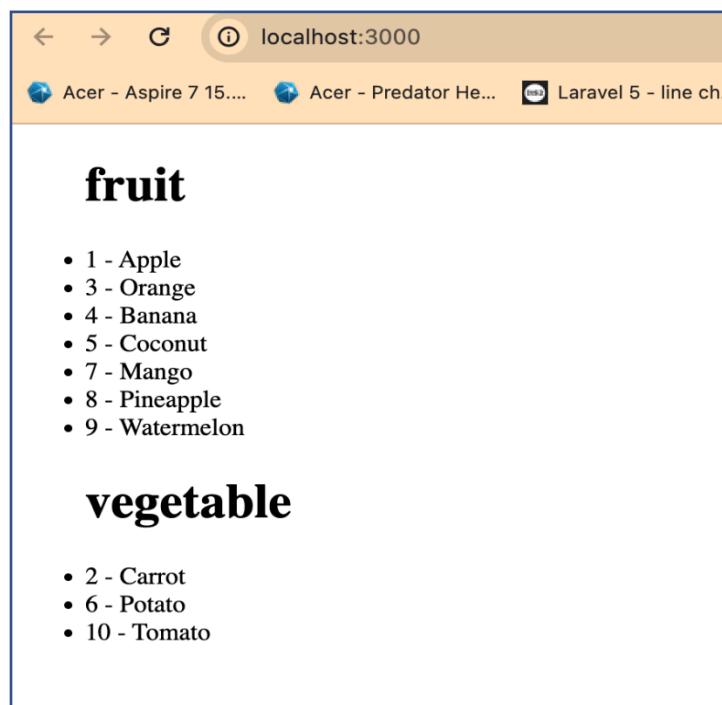
//Cheap Fruits (price is less than or equal to 2000)
//const cheapFruits=fruits.filter((f)=>f.price<=2000)
```



Example 3.

Project Files
 - src
 -- App.js
 -- List.jsx

» Sample Output:



» List.jsx

```

const List=(props)=>{
  const category=props.category
  const items=props.items;

  const list_items=items.map((item)=>
    item.category==category ? <li key={item.id}>{item.id}- {item.name} </li>: null )

  return (<ul>
    <h1>{category}</h1>
    {list_items}
  </ul>);

}

export default List

```

>> App.js

```
import List from './List'

const App=()=>{

  const items=[ {"id":1,"name":"Apple","price":1000,"category":"fruit"},  
    {"id":2,"name":"Carrot","price":1200,"category":"vegetable"},  
    {"id":3,"name":"Orange","price":1200,"category":"fruit"},  
    {"id":4,"name":"Banana","price":2000,"category":"fruit"},  
    {"id":5,"name":"Coconut","price":3000,"category":"fruit"},  
    {"id":6,"name":"Potato","price":1500,"category":"vegetable"},  
    {"id":7,"name":"Mango","price":2500,"category":"fruit"},  
    {"id":8,"name":"Pineapple","price":3000,"category":"fruit"},  
    {"id":9,"name":"Watermelon","price":4000,"category":"fruit"},  
    {"id":10,"name":"Tomato","price":1000,"category":"vegetable"}  
];

  return (  
    <>  
    <List items={items} category="fruit" />  
    <List items={items} category="vegetable" />  
    </>  
  );  
}

export default App
```

 Notes:

Ternary operator အကြောင်းနည်းနည်းပြောပြချင်တယ်၊ သာမှန် conditional statement ပုံစံကို single line condition ပုံစံအဖြစ် ပြောင်းလဲ ထားခိုင်းဖြစ်ပါတယ်။

Traditional Conditional Statement

Syntax:

```
If(con){
    T;
}
else{
    F;
}
```



```
con. ? T : F;
```

ကျနော်တို့ရေးထားတဲ့ အောက်ပါ source code ကို analysis လုပ်ကြည့်ရအောင်

 Code:

```
const list_items= items.map(
    (. item)=>
        item.category==category ? 
            <li key={item.id}>{item.id} - {item.name} </li>
            :
            null
)
```

Ternary operator

အခုနောက်ပိုင်း developer တွေ တွေ့ ternary operator ပုံစံကို logical operator နဲ့ ပြောင်းလဲ ရေးသားတဲ့ ပုံစံကို အောက်ပါအတိုင်းမြင်ရမယ်။

con ? True Block : False Block



con && True Block

လေ့ကျင့်ခန်းထဲက ternary operator ပုံစံကို အောက်ပါအတိုင်း logical and ပုံစံ ပြောင်းရေးပြီး output ကို ကြည့်ကြည့်ပါ။ ရလဒ် အတူတူဖြစ်နေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

Code:

```
const list_items= items.map(
    (. item) =>
        item.category === category &&
        <li key={item.id}>{item.id} - {item.name}</li>

    )
)
```

Component Data type checking နဲ့ default data initialization အကြောင်းဆက်ပြောပြချင်တယ်။
Data Type Checking အတွက် အောက်ပါအတိုင်း ရေးသားနိုင်ပါတယ်။

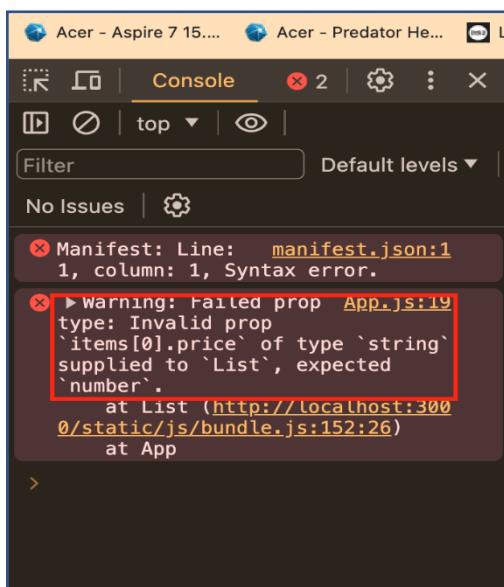
Code:

```
List.propTypes = {
    category: PropTypes.string,
    items: PropTypes.arrayOf(
        PropTypes.shape({
            id: PropTypes.number,
            name: PropTypes.string,
            price: PropTypes.number,
        })
    )
}
```

► Type Checking Test

```
const items=[ {"id":1,"name":"Apple","price" "10", "category":"fruit"},  
 {"id":2,"name":"Carrot","price":1200,"category":"vegetable"},  
 {"id":3,"name":"Orange","price":1200,"category":"fruit"},  
 {"id":4,"name":"Banana","price":2000,"category":"fruit"},  
 {"id":5,"name":"Coconut","price":3000,"category":"fruit"},  
 {"id":6,"name":"Potato","price":1500,"category":"vegetable"},  
 {"id":7,"name":"Mango","price":2500,"category":"fruit"},  
 {"id":8,"name":"Pineapple","price":3000,"category":"fruit"},  
 {"id":9,"name":"Watermelon","price":4000,"category":"fruit"},  
 {"id":10,"name":"Tomato","price":1000,"category":"vegetable"}  
];
```

► Error Report:

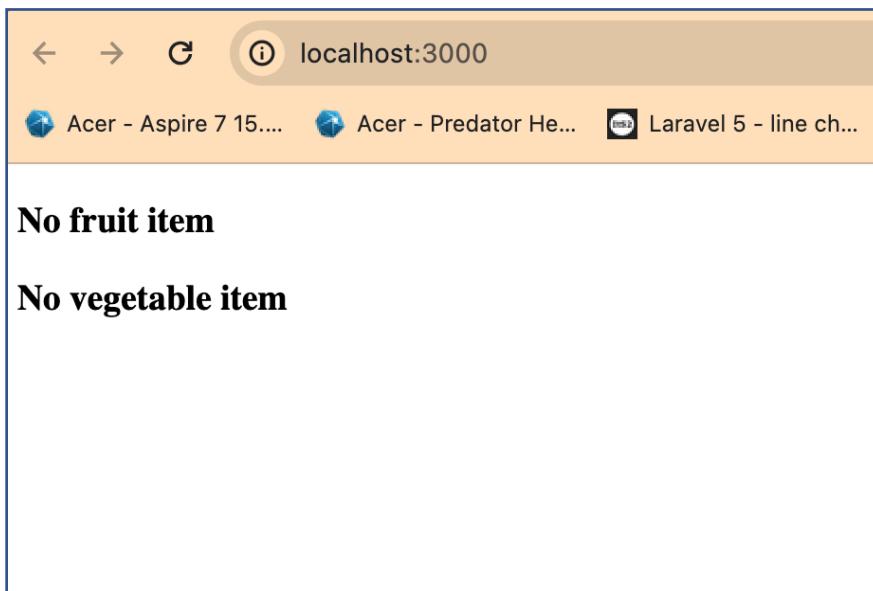


Data Type checking ပြီးသွားရင် တော့ Data type initialization ကို ရေးကြပါမယ်။

💻 Code:

```
List.defaultProps={  
    category:"Category",  
    items:[],  
}
```

Items list ကို empty လုပ်ပြီး အောက်ပါ output ကို ရအောင် try ကြည့်ပါ။



Spread Operator

Coding ရေးတဲ့အခါ Array data တွေကို မကြာခကာ ပေါင်းစည်းခြင်း၊ ဖြတ်တောက်ခြင်း၊ ပြုပြင်ခြင်းတွေကိုလုပ်ရပါတယ်။ ရှိုးရှိုး Array method တွေဖြစ်တဲ့ push၊ pop၊ splice၊ slice တွေအပြင် spread operator (...) တွေကို လည်း အကျမ်းတစ်ဝင်ဖြစ်နေဖို့ လိုပါတယ်။ အောက်ပါ Example တွေကို လိုက်လုပ်ပြီး လေ့လာကြည့်ကြရအောင်နေ။

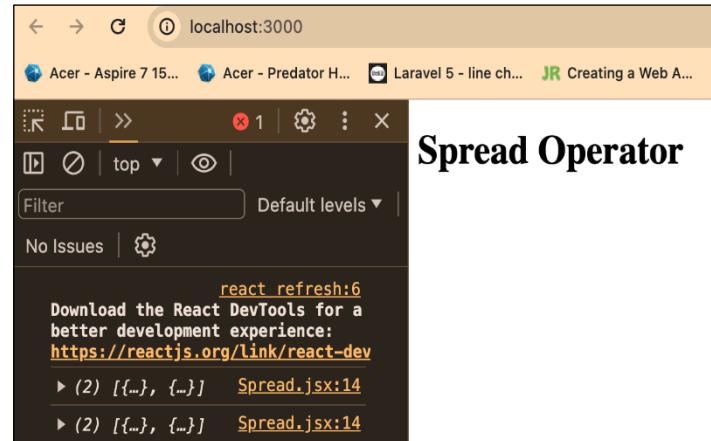
Example 1

Array Source:

```
const student_list = [
    {"id":1,"name":"mg mg","address":"hlaing"},
    {"id":2,"name":"su su","address":"innsein"},
    {"id":3,"name":"moe moe","address":"sanchaung"},
    {"id":4,"name":"thu zar","address":"la tha"},
];
```

Display All Students:

Console.log([...student_list])



Display All Data:

```
student_list.forEach((s)=>{
  console.log('-----')
  console.log(`Id : ${s.id}`)
  console.log(`Name : ${s.name}`)
  console.log(`Address : ${s.address}`)
  console.log('-----')
})
```

Id	Name	Address
1	mg mg	hlaing
2	su su	innsein

Add New Object into student list

Syntax:

[...array, new_object];

Code:

```
const new_student={
  "id":3,
  "name":"aung aung",
  "address":"san chaung"
}
```

```
const new_arr=[...student_list,new_student];
console.log("new arr =",new_arr)
```

Output:

Before adding new student

```
▼ (2) [{}], [{}]
  ► 0: {id: 1, name: 'mg mg', ad
  ► 1: {id: 2, name: 'su su', ad
    length: 2
  ► [[Prototype]: Array(0)]
```

After adding new student

```
Spread.jsx:21
▼ (3) [{}], [{}], [{}]
  ► 0: {id: 1, name: 'mg mg', ad
  ► 1: {id: 2, name: 'su su', ad
  ► 2: {id: 3, name: 'aung aung'
    length: 3
```

Update Array Object by Id

Syntax:

```
{...array,...update_object}
```

Code :

```
const update_student={  
    "id":1,  
    "name":"mg mg",  
    "address":"san chaung update"  
}
```

```
const new_arr=student_list.map((s)=>{  
  
    if(s.id==1){  
        return {...s,...update_student}  
    }  
    else{  
        return s;  
    }  
  
})
```

Output:

```
Spread.jsx:31  
▼ (2) [ {..}, {..} ] i  
▶ 0: {id: 1, name: 'mg mg', address: 'san chaung update'}  
▶ 1: {id: 2, name: 'su su', address: 'innsein'}  
length: 2  
▶ [[Prototype]]: Array(0)
```

Delete and object from array

Syntax:

```
array.filter((obj)=> obj.id!==delete_id);
```

Code:

```
let new_arr=student_list.filter((s)=> s.id!==1);
```

Output:

```
Spread.jsx:16
▼ [...]
  ► 0: {id: 2, name: 'su su', address: 'innsein'}
    length: 1
  ► [[Prototype]]: Array(0)
```

လေ့ကျင့်ရန် -

The screenshot shows a browser window with the URL `localhost:3000`. The page displays a list of students with their names and IDs in English and Burmese. There are four main sections: Original Student List, Add New Student, Update Student, and Delete Student List. Each section contains a list of student entries.

- Original Student List
 - 1 - mg mg (hlaing)
 - 2 - su su (innsein)
 - 3 - moe moe (sanchaung)
 - 4 - thu zar (la tha)
- Add New Student
 - 1 - mg mg (hlaing)
 - 2 - su su (innsein)
 - 3 - moe moe (sanchaung)
 - 4 - thu zar (la tha)
 - 5 - tun tun (innsein new)
- Update Student
 - Update Id : 2
- Delete Student List
 - 1 - mg mg (hlaing)
 - 2 - su su (innsein)
 - 4 - thu zar (la tha)

Hints:

Array Source:

```
const student_list = [
    {"id":1,"name":"mg mg","address":"hlaing"},
    {"id":2,"name":"su su","address":"innsein"},
    {"id":3,"name":"moe moe","address":"sanchaung"},
    {"id":4,"name":"thu zar","address":"la tha"},
];
```

New Object:

```
const newStudent={
    "id":5,
    "name":"tun tun",
    "address":"innsein new"
}
```

Object to be updated:

```
const updateStudent={
    "id":2,
    "name":"su mon",
    "address":"innsein update"
}
```

Display Data:

```
const StudentList = ({students}) => {  
  return (  
    <>  
    <ul>  
      {students.map(s => (  
        <li key={s.id}>  
          {s.id} - {s.name} ( {s.address} )  
  
        </li>  
      ))}  
    </ul>  
  );  
}
```



Click Event

React Video Lesson 0



Event



Example 1

Project Files

```
- src
  - - App.js
  - - Button.jsx
```



Output

A screenshot of a browser developer tools console window titled "Console" at localhost:3000. The window shows a list of log messages. At the top right is a button labeled "Click Me". The log messages are:

```
you click button 1 - Button.jsx:7
1 times
you click button 1 - Button.jsx:7
2 times
you click button 1 - Button.jsx:7
3 times
you click button 1 - Button.jsx:7
4 times
>
```

➡ App.js

```
import Button from './Button'

const App=()=>{

  return (
    <>
    <Button name={'Click Me'}>/>

    </>
  );
}

export default App
```

➡ Button.jsx

```
function Button(props){

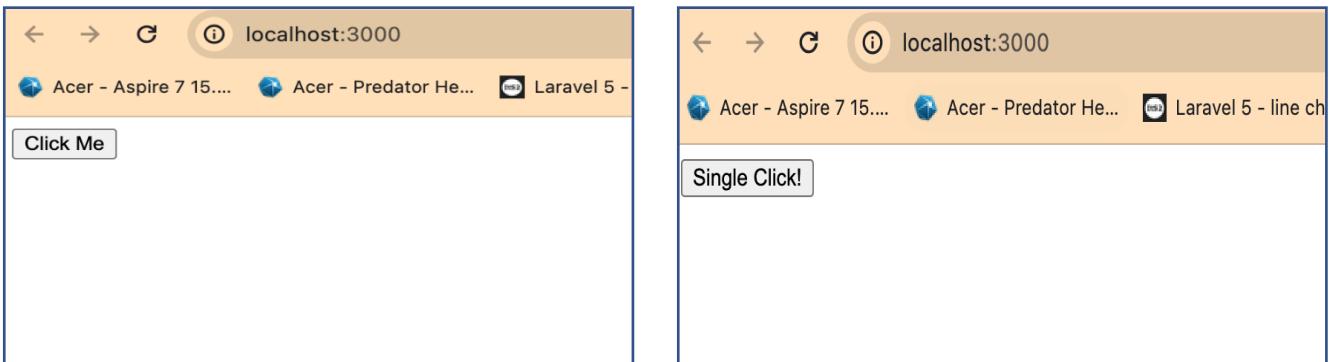
  let count=0;
  const handleClick= ()=> {
    count++;
    console.log(`you click button 1 - ${count} times `)

  }

  return (<button onClick={handleClick}> {props.name} </button>)

}
export default Button
```

↗ Single click



💻 Code:

📄 Button.jsx

```
function Button(props){

  let count=0;
  const handleClick= (e)=> {
    count++;
    count%2==0 ? e.target.textContent="Single Click!" : e.target.textContent=`${props.name}`

  }

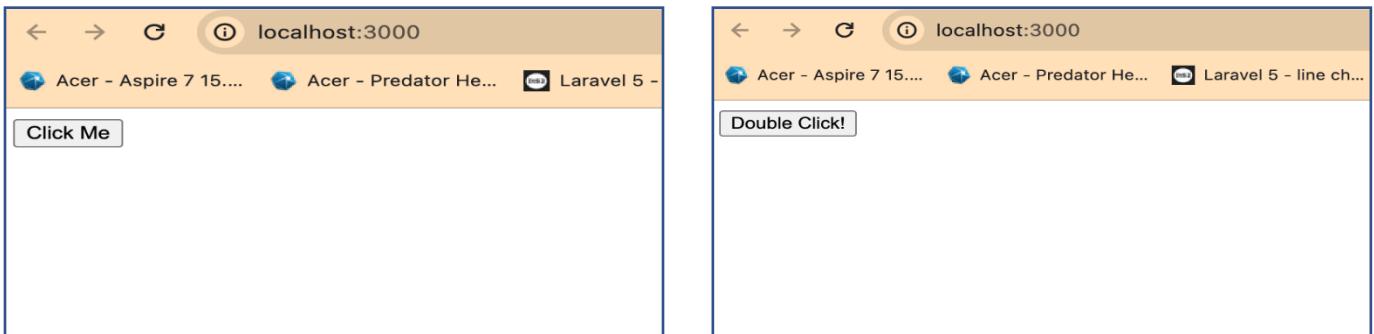
  return (<button onClick={(e)=>handleClick(e)}>{props.name} </button>)
}
```

📄 App.js

```
import Button from './Button'
const App=()=>{

  return (
    <>
    <Button name={'Click Me'} />
    </>
  );
}
export default App
```

👉 Double click



💻 Code :

Button.jsx

```
function Button(props){

    let count=0;
    const handleClick= (e)=> {
        count++;
        count%2==0 ? e.target.textContent="Double Click!" :
        e.target.textContent=`${props.name}`

    }

    return (<button onClick={handleClick}>{props.name}</button>)
}

export default Button
```



useState

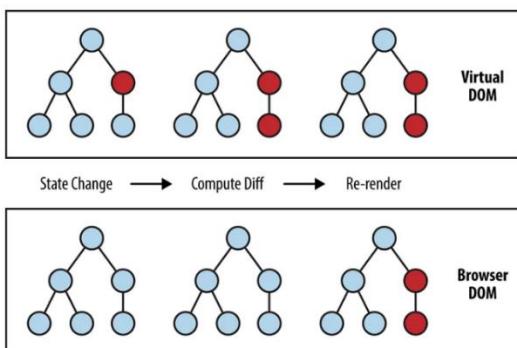
React Video Lesson 10



useState

React မှာ state က JavaScript object တစ်ခုဖြစ်ပြီးတော့ data တွေကိုသိမ်းထားနိုင်ပါတယ်။ state ထဲမှာ သိမ်းထားတဲ့ data တွေကို အချိန်နဲ့အမျှပြုပြင်ပြောင်းလဲနိုင်တယ်။ data တွေကိုအပြောင်းအလဲလုပ်ပြီး new state တွေကို သိမ်းဖို့ အတွက်ကျတော့ setState() built-in method ကို အသုံးပြုပါတယ်။

မေးစရာတစ်ခုရှိတာက ဘာလို့ ရိုးရိုး local variable တွေကိုမသုံးပဲ state variable တွေကို အသုံးပြုရတာလဲ ဆိုတဲ့မေးခွန်းပါ။ React မှာ ရိုးရိုး local variable တွေကို track မလုပ်ပေးပါဘူး။ ဥပမာ Component တစ်ခုမှာ အပြောင်းအလဲတစ်ခုခဲ့ရေးလိုက်တာနဲ့ အဲဒီ state update ရဲ့ရလဒ်ကို React က UI မှာ live update လုပ်ပေးပါတယ်။ useState() method ကို မသုံးပဲ local variable သုံးမယ်ဆို live update ဖြစ်မှာ မဟုတ်ပါဘူး။ ထို့ကြောင့် state variable သည် DOM ကို auto re-render လုပ်ပေးတာဖြစ်ပြီး local variable သည် DOM ကို auto re-render မလုပ်ပေးပါ။ ဒီလို့ React က state အပြောင်းအလဲကို auto detect လုပ်ပြီး DOM structure ကို auto re-render လုပ်ပေးတဲ့ ဖြစ်စဉ်ကို reconciliation လို့ခေါ်ပါတယ်။ Reconciliation ဖြစ်စဉ်က diff algorithm ကို သုံးပြီး DOM structure တွေကို အထိရောက်ဆုံးနဲ့ အမြန်ဆုံး စီမံနိုင်တာဖြစ်ပါတယ်။



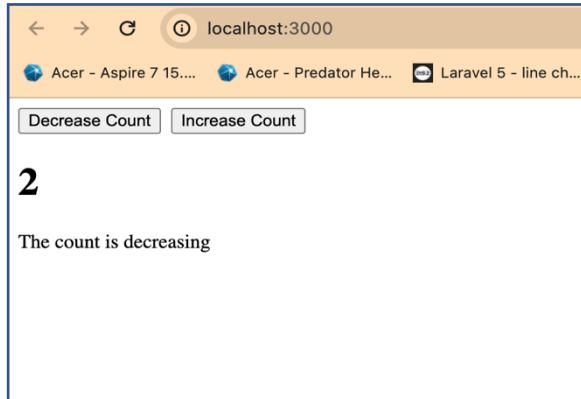
Reconciliation Process

Example Exercise တွေကို တစ်ခုပြီးတစ်ခု လိုက်လုပ်ပြီး state variable တွေကို နားလည်အောင် ကြိုးစားကြည့်ရအောင်နော်။

➡ Example 1

Increase and decrease counter

Sample Output:



Project Files

```
-src
  -- App.js
  -- Button.jsx
  -- CountDisplay.jsx
```

 Code:

App.js

```
import CountDisplay from './CountDisplay'  
import { useState } from 'react';  
const App=()=>{  
  
    const [counter,setCounter]=useState(0)  
  
    const clickHandle1=(prev)=>{  
        prev=prev-1  
        setCounter(prev)  
    }  
  
    const clickHandle2=(next)=>{  
        next=next+1  
        setCounter(next)  
    }  
    return (  
        <>  
        <button onClick={()=>clickHandle1(counter)}> Decrease Count </button>  
        &nbsp;  
        <button onClick={()=>clickHandle2(counter)}> Increase Count </button>  
        <CountDisplay count={counter}/>  
  
        </>  
    );  
}  
  
export default App
```

CountDisplay.jsx

```
import { useState } from 'react';

const CountDisplay = ({ count }) => {
  const [prevCount, setPrevCount] = useState(count);
  const [trend, setTrend] = useState(null);
  if (prevCount !== count) {
    setPrevCount(count);
    setTrend(count > prevCount ? 'increasing' : 'decreasing');
  }
  return (
    <>
      <h1>{count}</h1>
      {trend && <p>The count is {trend}</p>}
    </>
  );
}

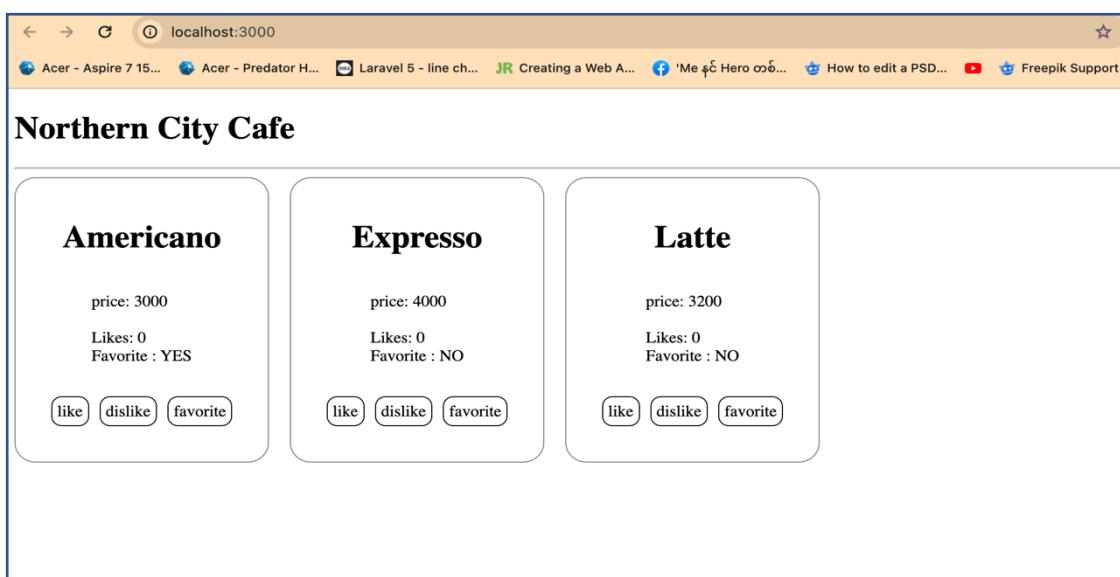
export default CountDisplay
```

Example 2

Project Files

```
- src
  - App.js
  - Item.jsx
  - Item.css
```

Output



App.js

```
import Item from "./Item"
import { useState } from "react"

import "./Item.css"
const App=()=> {

    const [fav,setFav]=useState(1)

    const items= [
        {id:1,name:"Americano",price:3000},
        {id:2,name:"Expresso",price:4000},
        {id:3,name:"Latte",price:3200}
    ];

    const handleFav=(id)=>{
        setFav(id)
    }
    return (
        <>
            <h1>Northern City Cafe </h1>
            <hr/>
            <div className="row">
                {
                    items.map((item) =>
                        <Item key={item.id} onFavorite={handleFav} favorite={fav==item.id} id={item.id} name={item.name} price={item.price} />
                    )
                }
            </div>
        </>
    )
}

export default App
```

Item.jsx

```

import "./Item.css"
import { useState } from "react"

const Item=(props)=>{

    const [likes,setLike]=useState(0)

    const handleLike=()=>{
        const newLikes=likes+1
        setLike(newLikes)
    }

    const handleDislike=()=>{
        const newLikes=likes-1
        setLike(newLikes)
    }

    const handleFav=()=>{
        props.onFavorite(props.id)
    }

    return (
        <>
        <div className="card">
            <h1>{props.name} </h1>
            <p> price: {props.price} <br/><br/>
                Likes: {likes} <br/>
                Favorite : {props.favorite ? "YES" : "NO"} </p>
            <p className="cta">
                <span className="like-btn" onClick={handleLike}>like </span>
                <span className="dislike-btn" onClick={handleDislike}>dislike </span>
                <span className="fav-btn" onClick={handleFav}>favorite </span>
            </p>
        </div>
        </>
    )
}

export default Item

```

Item.css

```
.row{  
    display: flex;  
    flex-direction: row;  
    justify-content: start;  
    align-items: center;  
    gap:20px;  
}  
.card{  
    width:200px;  
    display: flex;  
    flex-direction: column;  
    justify-content: start;  
    align-items: center;  
    padding:20px;  
    border-radius: 20px;  
    border:1px solid gray;  
}  
.cta{  
    display: flex;  
    flex-direction: row;  
    justify-content: center;  
    align-items: center;  
    gap:10px;  
}  
.like-btn,.dislike-btn,.fav-btn{  
    padding:5px;  
    border:1px solid black;  
    border-radius: 10px;  
}  
.like-btn:hover,.dislike-btn:hover,.fav-btn:hover{  
    padding:5px;  
    border:1px solid rgb(249, 66, 16);  
    border-radius: 10px;  
    color:rgb(251, 251, 251);  
    cursor: pointer;  
    background-color: black;  
}
```

Outputs:

Northern City Cafe

Add New Item

test3 price: 2000 Likes: 4 Favorite : YES

like dislike favorite

Latte price: 3000 Likes: 0 Favorite : NO

like dislike favorite

Americano price: 3500 Likes: 0 Favorite : NO

like dislike favorite

Northern City Cafe

Invalid inputs..check your inputs....

Save

Northern City Cafe

Expresso

3500

Save

Northern City Cafe

Add New Item

Expresso price: 3500 Likes: 0 Favorite : NO

Latte price: 3000 Likes: 0 Favorite : NO

Americano price: 3500 Likes: 0 Favorite : NO

Project Files

```
-- src
  --- App.css
  --- App.js
  --- Item.css
  --- Item.jsx
  --- Form.css
  --- Form.jsx
```

App.js

```
import Form from "./Form";
import Item from "./Item";
import "./Form.css";
import "./App.css";
import { useState } from "react";

function App() {
  const [fav, setFav] = useState(false);
  const [showForm, setShowForm] = useState(false);
  const [items, setItems] = useState([
    {
      id: 1,
      name: "Latte",
      price: 3000,
    },
    {
      id: 2,
      name: "Americano",
      price: 3500,
```

```

    },
]);

const handleDeleteItem = (id) => {
  setItems(items.filter((item) => item.id !== id));
};

const handleAddItem = (name, price) => {
  const item = {
    id: window.self.crypto.randomUUID(),
    name: name,
    price: price,
    like: 0,
  };

  setItems([item, ...items]);
  setShowForm(false);
};

const addNewItem = () => {
  setShowForm(true);
};

const handleFav = (id) => {
  setFav(id);
};

let jsx = null;

if (showForm) {
  jsx = <Form onAddItem={handleAddItem}></Form>;
} else {
  jsx = (
    <>
      <button onClick={addNewItem} className="add-btn">
        {" "}
        Add New Item{" "}
      </button>
      {items.map((item) => (
        <Item
          key={item.id}
          {...item}
          onDelete={handleDeleteItem}
          onFavorite={handleFav}
        >
      ))}
    </>
  );
}

return (
  <div>
    {jsx}
  </div>
);

```

```
        favorite={fav === item.id}
        id={item.id}
      ></Item>
    )})
  </>
);
}

return (
<>
<div className="main">
  <h1>Northern City Cafe</h1>
  <hr />

  {jsx}
</div>
</>
);
}

export default App;
```

App.css

```
.main{
  width: 100%;
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
  justify-content: center;
  align-items: center;
}

.add-btn{
  width:300px;
  height:20px;
  padding:20px;
  border-radius: 20px;
  border:0;
  background-color: orange;
  color:white;
```

```
display: flex;  
justify-content: center;  
align-items: center;  
margin: 20px 0;  
  
cursor: pointer;  
  
}
```

Item.jsx

```
import "./Item.css";  
import { useState } from "react";  
  
const Item = (props) => {  
  const [likes, setLike] = useState(0);  
  
  const handleLike = () => {  
    const newLikes = likes + 1;  
    setLike(newLikes);  
  };  
  
  const handleDislike = () => {  
    const newLikes = likes - 1;  
    setLike(newLikes);  
  };  
  
  const handleFav = () => {  
    props.onFavorite(props.id);  
  };  
  return (  
    <>
```

```

<div className="card">
  <ul className="detail">
    <li>
      {" "}
      <h2> {props.name} </h2>{" "}
    </li>
    <li> price: {props.price} </li>
    <li> Likes: {likes} </li>
    <li> Favorite : {props.favorite ? "YES" : "NO"} </li>
  </ul>
  <p className="cta">
    <span className="like-btn" onClick={handleLike}>
      like{" "}
    </span>
    <span className="dislike-btn" onClick={handleDislike}>
      dislike{" "}
    </span>
    <span className="fav-btn" onClick={handleFav}>
      favorite{" "}
    </span>
  </p>
</div>
</>
);
};

export default Item;

```

Item.css

```

.card{
  width:50%;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  padding:20px;
  border-radius: 20px;
  border:1px solid gray;
}

```

```
margin:10px;
}

.cta{
  width:50%;
  display: flex;
  flex-direction: row;
  justify-content: end;
  align-items: end;
  gap:10px;
}

.like-btn,.dislike-btn,.fav-btn{
  padding:5px;
  border:1px solid black;
  border-radius: 10px;
}

.detail{
  width: 50%;
  display: flex;
  justify-content: row;
  align-items: center;
  gap:20px;
  list-style: none;
}

}

.like-btn:hover,.dislike-btn:hover,.fav-btn:hover{
  padding:5px;
  border:1px solid rgb(249, 66, 16);
  border-radius: 10px;
  color:rgb(251, 251, 251);
  cursor: pointer;
  background-color: black;
}
```

Form.jsx

```
import { useState } from "react";

const Form = ({ onAddItem }) => {
```

```

const [name, setName] = useState("");
const [price, setPrice] = useState(0);
const [error, setError] = useState("");

const submitHandle = (e) => {
  e.preventDefault();
  if (name.length < 3 || price === 0) {
    setError("Invalid inputs..check your inputs...");
    return;
  }
  setError("");
  onAddItem(name, price);
  setName("");
  setPrice(0);
};

return (
  <>
  <form onSubmit={submitHandle}>
    {error && <p> {error} </p>}
    <input
      type="text"
      value={name}
      onChange={(e) => setName(e.target.value)}
    />
    <input
      type="number"
      value={price}
      onChange={(e) => setPrice(e.target.value)}
    />
    <button className="submit-btn"> Save </button>
  </form>
  </>
);
};

export default Form;

```

Form.css

```
form{  
    display: flex;  
    flex-direction: column;  
    justify-content: start;  
    align-items: start;  
    gap:10px;  
}
```

```
input{  
    width:250px;  
    height:20px;  
    padding:5px;  
    border-radius: 10px;  
    border-color:silver;  
  
    text-align:center;  
}
```

```
.submit-btn{  
  
    width:270px;  
    height:30px;  
    padding:15px;  
  
    background-color: orange;  
    color:black;  
    border-radius: 20px;  
    border:0;  
  
    margin-top:10px;  
  
    display: flex;  
    justify-content: center;  
    align-items: center;  
  
}
```

```
.submit-btn:hover{  
  
    width:270px;  
    height:30px;  
    padding:15px;  
    cursor: pointer;
```

```
background-color: rgb(65, 46, 12);  
color:white;  
}
```

useEffect

 React Video Lesson 11



useEffect

Application တစ်ခုရဲ့ components တွေမှာ side effects တွေကို လုပ်ဆောင်ရွက်အောင်လို့ useEffect ကိုအသုံးပြုတာဖြစ်ပါတယ်။ side effect ဥပမာ တွေကတွေ API data fetching လုပ်တာတွေ၊ DOM structure တွေကို direct update လုပ်တာတွေ၊ computer battery status တွေ၊ network notification တွေနဲ့ user online status notification control တွေပဲဖြစ်ပါတယ်။ useEffect() method ထဲမှာ parameter ၂ ခုရှိပါတယ်။ ဒုတိယ parameter ကတော့ optional ပါ။

```
useEffect(<function>, <dependency>)
```

useEffect ကို နားလည်ဖို့ဆိုရင် react ရဲ့ lifecycle ၃ ခု ကိုအရင် သိဖို့လိုပါမယ်။

1. Mounting
2. State Changes
3. Unmounting

ဒီ process ၏ ခုထဲက တစ်ခုခု အပြောင်းအလဲဖြစ်တဲ့အခါတိုင်း useEffect() က အလုပ်လုပ်ပါတယ်။ ပိုပြီးမြင်သာအောင်ကြည့်မယ်ဆိုရင် အောက်ပါအတိုင်း lifecycle process နဲ့ တွဲပြီးမြင်လို့ရပါတယ်။

1. Mounting
useEffect(() => {}, [])
2. State Changes
useEffect(() => {}, [state])
3. Unmounting
useEffect(() => {return () = {}}, [])

App page တစ်ခုကို စစချင်း render လုပ်လိုက်တာနဲ့ dependency array က empty ဖြစ်နေမှာပါ။ Page ပေါ်မှာ component action တစ်ခုခုလုပ်လိုက်တာနဲ့ state changes ဖြစ်ပေါ်မှာ ဖြစ်ပြီး dependency အနေနဲ့ အသုံးပြုရမှ ဖြစ်ပါတယ်။ component unmount ဖြစ်သွားပြီးဆိုတာနဲ့ dependency array တွေပြန်ပြီး empty ဖြစ်သွားမှာ ဖြစ်ပါတယ်။ useEffect က react ရဲ့ အရေးကြီးဆုံး hook တစ်ခုဖြစ်ပြီးတော့ cleanup function တွေကို useEffect ထဲမှာပဲ ပြုလုပ်ရပါတယ်။ ဥပမာ - remove active listener ပြုလုပ်တာတို့၊ clear timeout ပြုလုပ်တာ တို့ပေါ့။

ဆက်လက်ပြီး အသေးစိတ်ကို Example တွေလုပ်ရင်း လေ့လာကြည့်ရအောင်နော် -

Example 1

```
import { useState, useEffect } from "react";

function App() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  }, [count]);

  return (
    <div>
      <button onClick={() => setCount((prevCount) => prevCount + 1)}>
        Click {count} times{" "}
      </button>
    </div>
  );
}
export default App;
```

Output:



Example 2

```
import { useState, useEffect } from "react";

function UseEffectExample2() {
  const [userData, setUserData] = useState(null);

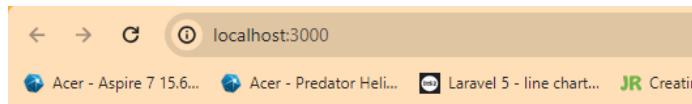
  useEffect(() => {
    fetch('https://random-data-api.com/api/users/random_user')
      .then(response => response.json())
      .then(data => setUserData(data));
  }, []);

  return (
    <div>
      {userData && (
        <div>
          <h2>User Information</h2>

          <p>
            Name:
            {userData.first_name}
            {userData.last_name}
          </p>
          <p>
            Email: {userData.email}
          </p>
          {/* Add more user data fields as needed */}
        </div>
      )}
    </div>
  );
}

export default UseEffectExample2;
```

Output:



User Information

Name: MieshaOberbrunner
Email: miesha.oberbrunner@email.com

Example 3

```
import { useEffect, useState } from "react";

const useEffectPhoto = () => {
  const [imageURL, setImageURL] = useState(null);

  useEffect(() => {
    fetch("https://api.thedogapi.com/v1/images/search")
      .then((response) => response.json())
      .then((response) => setImageURL(response[0].url))
      .catch((error) => console.error(error));
  }, []);

  return (
    imageURL &&
    <>
      <h1>An image</h1>
      <img src={imageURL} alt={"placeholder text"} />
    </>
  );
};

export default useEffectPhoto;
```

Output:



An image



Practice

Generate Random User from api

Api endpoint: https://random-data-api.com/api/v2/users?response_type=json

localhost:3000



Branden Zulauf
Chief IT Assistant
Email:branden.zulauf@email.com
Phone:+358 1-533-133-4109
x610
Location:Townetown

Get Random User

localhost:3000



Edwardo Lind
Legacy Designer
Email:edwardo.lind@email.com
Phone:+229 133-023-5148
x46729
Location:New Damon

Get Random User

End of Free Version

Premium Version (Facebook private group)

(You will be provided source codes & video lessons)

1. Node JS fundamentals
2. Node JS Server
3. Mongo Database
4. Node JS API & Postman
5. React JS CRUD
6. React JS JWT authentication
7. React JS Payment Checkout
8. Email OTP Verification
9. Social Media Login
10. Git & Github
11. Docker
12. Project Deployment to web server
13. Project Codes and Video Lessons
 - o Movieland App
 - o Food Delivery App
 - o Ecommerce App

Contact :

- **Phone: 09449008972, 09425026458**
- **Email: northerncitycenter@gmail.com**