

DAY 21 : REACT 101

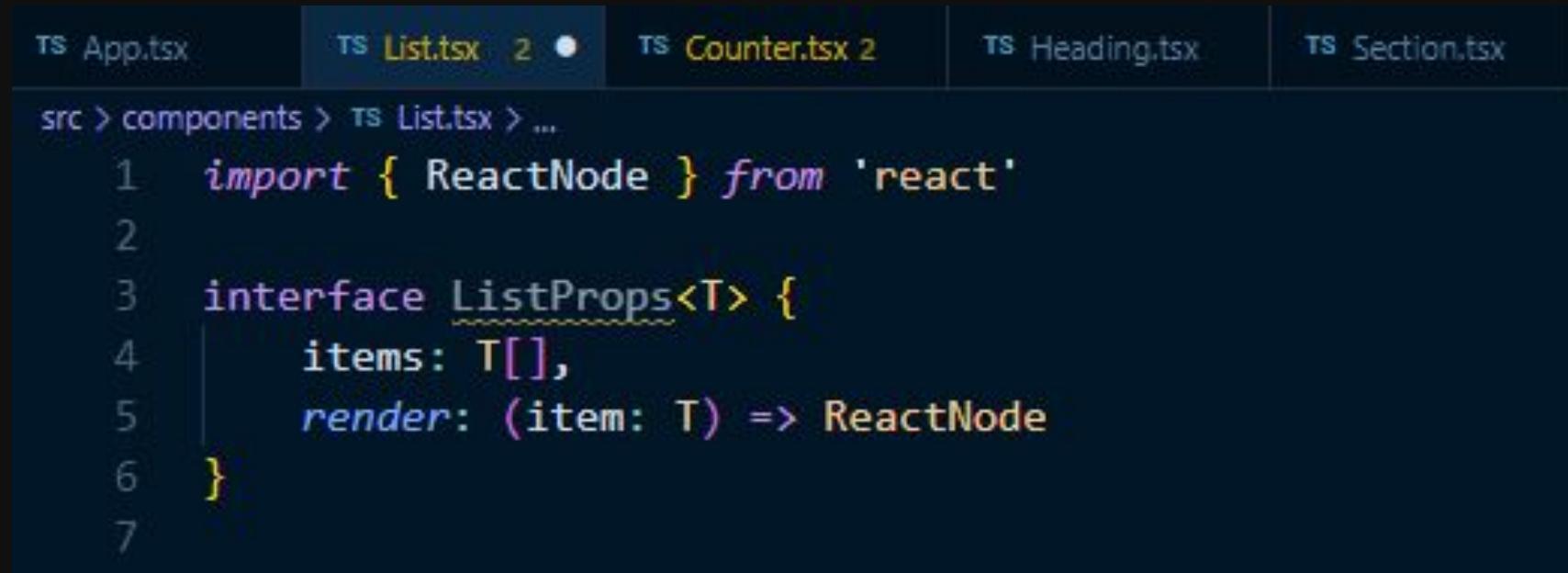
Road to Front-end Developer Bootcamp

Generic List

สร้างไฟล์ List.tsx

```
TS App.tsx          TS List.tsx 2 ● TS Counter.tsx 2      TS Heading.tsx      TS Section.tsx
src > components > TS List.tsx > ...
1 import { ReactNode } from 'react'
2
3 interface ListProps<T> {
4     items: T[],
5     render: (item: T) => ReactNode
6 }
7
```

สร้าง Interface ขึ้นมา

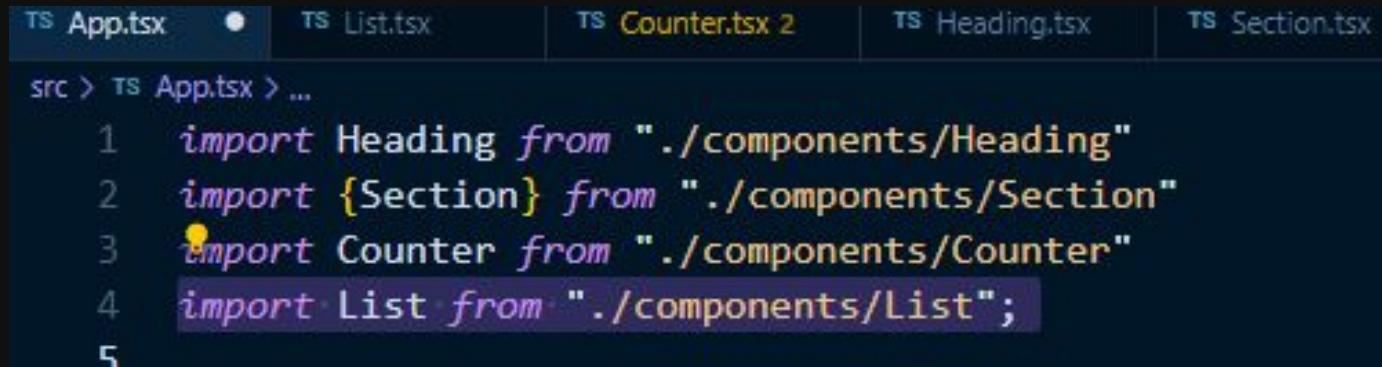


```
src > components > List.tsx > ...
1 import { ReactNode } from 'react'
2
3 interface ListProps<T> {
4     items: T[],
5     render: (item: T) => ReactNode
6 }
7
```

ต่อมาทำการสร้าง List ขึ้นมา

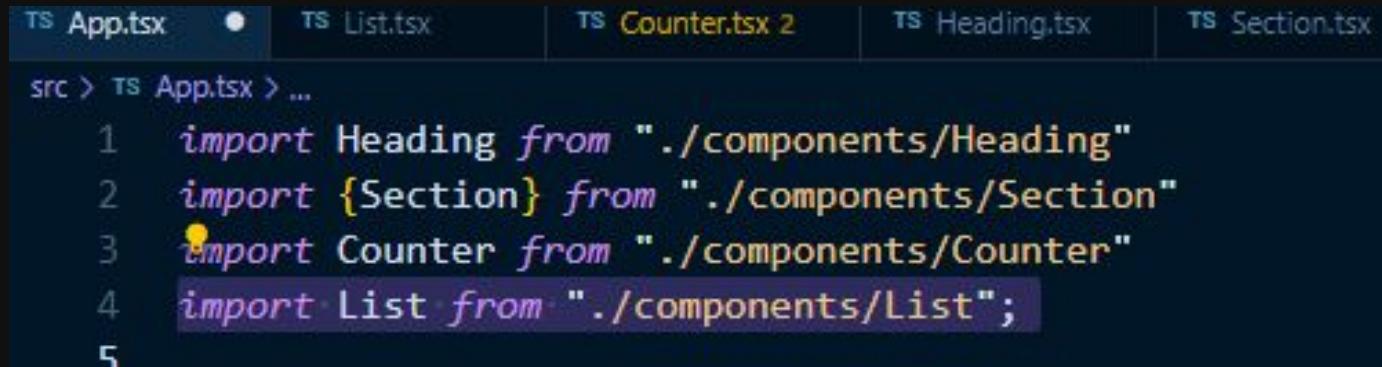
```
src > components > ts List.tsx > ...
1  import { ReactNode } from 'react'
2
3  interface ListProps<T> {
4      items: T[],
5      render: (item: T) => ReactNode
6  }
7
8  const List = <T,>({ items, render }: ListProps<T>) => {
9      return (
10         <ul>
11             <li key={index}>
12                 {render(item)}
13             </li>
14         )})
15     </ul>
16 )
17 }
18 }
19
20 export default List
```

นำเข้า List มาใน App.tsx



```
src > TS App.tsx > ...
1 import Heading from "./components/Heading"
2 import {Section} from "./components/Section"
3 import Counter from "./components/Counter"
4 import List from "./components/List";
5
```

นำเข้า List มาใน App.tsx



```
src > TS App.tsx > ...
1 import Heading from "./components/Heading"
2 import {Section} from "./components/Section"
3 import Counter from "./components/Counter"
4 import List from "./components/List";
5
```

แก้ไขฟังก์ชัน App ในการ Return และ แสดงผล List ที่สร้างไว้

```
// function App() {
//   const [count, setCount] = useState<number>(1);

//   return (
//     <>
//       <Heading title="Hello It's me"/>
//       <Section> My Section</Section>
//       <Counter setCount={setCount}>Count = {count}</Counter>
//       <List items={[["★ Star", "🍴 Fork", "🔥 Fire"]} render={(item) => <b>{item}</b>}></List>
//     </>
//   )
// }

export default App
```



Conditional Rendering



Conditional Rendering

คือ เงื่อนไขที่เราสามารถกำหนดได้ ก่อนจะทำการ Render หน้าเว็บ

```
1 import React from 'react'  
2  
3  
4 let isLogin:boolean = false;  
5  
6 const Dashboard = () => {  
7   return (  
8     <div>Dashboard</div>  
9   )  
10 }  
11  
12 export default Dashboard
```

Quiz : ถ้า Login เป็น True ให้ขึ้น Dashboard ถ้า False ให้ขึ้น Not Login

```
1 import React from 'react'  
2  
3  
4 let isLogin:boolean = false;  
5  
6 const Dashboard = () => {  
7   return (  
8     <div>Dashboard</div>  
9   )  
10 }  
11  
12 export default Dashboard
```



Not Login

ให้เวลาประมาณ 3 นาที



เราสามารถกำหนดเงื่อนไขในลักษณะนี้
กับ Component อื่น ๆ ก็ได้เช่นกัน

Login แล้วหรือยัง ?

หน้า Login

หน้า Dashboard

Quiz : ลองสร้าง หน้า Login แยกออกมา และ ตรวจสอบเงื่อนไขว่า Login อยู่หรือไม่ ถ้าใช่ให้ Render Dashboard ถ้าไม่ใช่ ให้เป็นหน้า Login



กรณีที่เราต้องการ Interactive Form ณ. เวลาหนึ่ง ๆ เลย ..

ກຳກາຮສ້າງ State ໃຊ້ອັບ User / Pwd

```
1 import React from 'react'
2 import { useState } from 'react'
3
4 const Login = () => {
5     const [Username, setUsername] = useState<string>();
6     const [Password, setPassword] = useState<string>();
7
8     return (
9         <div>Login</div>
10    )
11 }
12
13 export default Login|
```

สร้างฟังก์ชันสำหรับรองรับเมื่อผู้ใช้งาน ได้เปลี่ยน Username หรือ รหัสผ่าน

```
const onChangeUsername = (event: React.ChangeEvent<HTMLInputElement>) => {
    setUsername(event.target.value)
    console.log(Username)
}
```

```
const onChangePassword = (event: React.ChangeEvent<HTMLInputElement>) => {
    setPassword(event.target.value)
    console.log(Password)
}
```

ฟังก์ชันที่จะทำงานเมื่อมีการ Submit

```
const onSubmit = (event: React.FormEvent<HTMLFormElement>) => {  
  event.preventDefault()  
  console.log("Submitted")  
}
```

สิ่งที่เราจะทำการสร้างในฟอร์ม

```
return (
  <>
  <div>
    <form onSubmit={onSubmit}>
      <label htmlFor="name">Name</label>
      <input type="text" id="username" name="name" onChange={onChangeUsername}/>
      <input type="password" id="password" name="name" onChange={onChangePassword}/>
    </form>
  </div>
  </>
)
}

export default Login|
```

Tips : กรณีแปลง Str > number เราสามารถใส่ + ด้านหน้าได้เลย

```
const [Username, setUsername] = useState<number>();
const [Password, setPassword] = useState<number>();
const onChangeUsername = (event: React.ChangeEvent<HTMLInputElement>) => {
    setUsername(+event.target.value)
    console.log(Username)
}

const onChangePassword = (event: React.ChangeEvent<HTMLInputElement>) => {
    setPassword(+event.target.value)
    console.log(Password)
}
```

Exactly like in JavaScript, you can use the [parseInt](#) or [parseFloat](#) functions, or simply use the unary `+ operator`:

2250

```
var x = "32";
var y: number = +x;
```

All of the mentioned techniques will have correct typing and will correctly parse simple decimal integer strings like `"123"`, but will behave differently for various other, possibly expected, cases (like `"123.45"`) and corner cases (like `null`).

x	parseInt(x)	parseFloat(x)	Number(x)	xx	-x	x>>0	isNaN(x)
"123"	123	123	123	123	123	123	false
"+123"	123	123	123	123	123	123	false
"-123"	-123	-123	-123	-123	-123	-123	false
"123.45"	123	123.45	123.45	123.45	123	123	false
"-123.45"	-123	-123.45	-123.45	-123.45	-123	-123	false
"12e5"	12	1200000	1200000	1200000	1200000	1200000	false
"12e-5"	12	0.00012	0.00012	0.00012	0	0	false
"0123"	123	123	123	123	123	123	false
"0000123"	123	123	123	123	123	123	false
"0111"	0	0	7	7	7	7	false
"0b10"	0	0	8	8	8	8	false
"0xBABE"	47806	0	47806	47806	47806	47806	false
"4294967295"	4294967295	4294967295	4294967295	4294967295	-1	4294967295	false
"123456789012345678"	123456789012345680	123456789012345680	123456789012345680	123456789012345680	-1506741424	2798225872	false
"12e999"	12	Infinity	Infinity	Infinity	0	0	false
--	NaN	NaN	0	0	0	0	false
"123foo"	123	123	NaN	NaN	0	0	true
"123.45foo"	123	123.45	NaN	NaN	0	0	true
" 123 "	123	123	123	123	123	123	false
"foo"	NaN	NaN	NaN	NaN	0	0	true
"12e"	12	12	NaN	NaN	0	0	true
"0b567"	0	0	NaN	NaN	0	0	true
"09999"	0	0	NaN	NaN	0	0	true
"0xFUZZ"	15	0	NaN	NaN	0	0	true
"-0"	0	0	0	0	0	0	false
"Infinity"	NaN	Infinity	Infinity	Infinity	0	0	false
"-Infinity"	NaN	Infinity	Infinity	Infinity	0	0	false
"-Infinity"	NaN	-Infinity	-Infinity	-Infinity	0	0	false
null	NaN	NaN	0	0	0	0	false
undefined	NaN	NaN	NaN	NaN	0	0	true
true	NaN	NaN	1	1	1	1	false
false	NaN	NaN	0	0	0	0	false
Infinity	NaN	Infinity	Infinity	Infinity	0	0	false
NaN	NaN	NaN	NaN	NaN	0	0	true
()	NaN	NaN	NaN	NaN	0	0	true
{valueOf: function(){return 42}}	NaN	NaN	42	42	42	42	false
{toString: function(){return "56"}}	56	56	56	56	56	56	false

Quiz : จากตัวอย่างก่อนหน้า ลองรับข้อมูลตัวเลข 2 ตัว มาคำนวณหา BMI และ แสดงในรูปแบบ Console

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

const Login = () => {
    const [Username, setUsername] = useState<string>();
    const [Password, setPassword] = useState<string>();
    const onChangeUsername = (event: React.ChangeEvent<HTMLInputElement>) => {
        setUsername(event.target.value)
        console.log(Username)
    }

    const onChangePassword = (event: React.ChangeEvent<HTMLInputElement>) => {
        setPassword(event.target.value)
        console.log(Password)
    }

    const onSubmit = (event: React.FormEvent<HTMLFormElement>) => {
        event.preventDefault()
        console.log("Submitted")
    }

    return (
        <>
        <div>
            <form onSubmit={onSubmit}>
                <label htmlFor="name">Name</label>
                <input type="text" id="username" name="name" onChange={onChangeUsername} />
                <input type="password" id="password" name="pwd" onChange={onChangePassword} />
            </form>
        </div>
        </>
    )
}

export default Login
```

ให้เวลาประมาณ 5 นาที

Quiz 2 : จากตัวอย่างก่อนหน้า ลองรับข้อมูลตัวเลข 2 ตัว มาคำนวณหา BMI และ แสดงในรูปแบบ Text จาก State

25.781249999999996

Name 1.66

น้ำหนัก / (ส่วนสูง ^ 2)

ให้เวลาประมาณ 10 นาที



การทำ React ควบคู่กับ Form ทั่วไป

https://react-hook-form.com

Search ... V5/V6

Home Get Started API TS Advanced FAQs Tools ▾ Resources ▾ Releases ❤

34,936

React Hook Form

Performant, flexible and extensible forms with easy-to-use validation.

Demo Get Started ►

The screenshot shows the React Hook Form website. At the top, there's a search bar, a version selector (V5/V6), and a navigation menu with links like Home, Get Started, API, TS, Advanced, FAQs, Tools, Resources, and Releases. A social sharing section with icons for LinkedIn, Twitter, GitHub, and a notification count of 34,936 follows. The main heading is "React Hook Form" with a pink clipboard icon. Below it is a subtitle: "Performant, flexible and extensible forms with easy-to-use validation." Two buttons are present: "Demo" and "Get Started ►". The central part of the page features a dark-themed code editor window titled "App.jsx - my-react-app" containing sample code for a form. To the right of the code editor is a mobile device simulation for an iPhone 12 Pro Max running iOS 14, displaying a form with fields for First Name and Last Name, a counter labeled "Render: 1", and a pink "SUBMIT" button. The mobile screen also shows the URL "localhost:632". The bottom of the page includes a video player bar with a play button, a progress bar from 0:00 to 0:30, and several other controls. At the very bottom, there's footer text in Thai: "สงวนลิขสิทธิ์ เมื่อหาก็งหมด หรือ ส่วนหนึ่งส่วนใด - บริษัท บอร์นกูเดฟ จำกัด".

ถ้าใน Windows ไม่ต้อง sudo

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL AZURE

bash + × ⊞ ⊖ ... ^ X

Kittikorns-MacBook-Pro:app-name kittikornprasertsak\$ sudo npm install react-hook-form

เริ่มจาก Import ตัว react-hook-form

```
2 import { useForm } from 'react-hook-form'  
3  
4 type Inputs = {  
5   email: string,  
6   password: string  
7 }
```

ต่อมาเขียนเนื้อหาใน

```
9  const MyForm = () => {
10    const { register, handleSubmit } = useForm<Inputs>()
11
12    const onSubmit = (data:any) => console.log(data)
13
14    return (
15      <form onSubmit={handleSubmit(onSubmit)}>
16        <input {...register('email')} />
17        <input {...register('password')} />
18        <input type='submit' />
19      </form>
20    )
21  }
22
23  export default MyForm
```

Dashboard

dasd	112233	Submit
------	--------	--------

ต่อมาเขียนเนื้อหาใน

```
9  const MyForm = () => {
10    const { register, handleSubmit } = useForm<Inputs>()
11
12    const onSubmit = (data:any) => console.log(data)
13
14    return (
15      <form onSubmit={handleSubmit(onSubmit)}>
16        <input {...register('email')} />
17        <input {...register('password')} />
18        <input type='submit' />
19      </form>
20    )
21  }
22
23  export default MyForm
```

และ สุดท้ายให้เราทำการใส่ไว้ใน App.tsx

```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5 import Dashboard from './components/Dashboard'
6 import Login from './components/Login'
7 import MyForm from './components/MyForm'
8
9 function App() {
10
11   let isLoggedIn:boolean = true
12   if(isLoggedIn)
13   {
14     return (<><Dashboard/> <MyForm/></>)
15   }
16   else{
17     return (<Login/>)
18   }
19 }
20
21 export default App
```

และ สุดท้ายให้เราทำการใส่ไว้ใน App.tsx

```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5 import Dashboard from './components/Dashboard'
6 import Login from './components/Login'
7 import MyForm from './components/MyForm'
8
9 function App() {
10
11   let isLoggedIn:boolean = true
12   if(isLoggedIn)
13   {
14     return (<><Dashboard/> <MyForm/></>)
15   }
16   else{
17     return (<Login/>)
18   }
19 }
20
21 export default App
```

⭐ การลิงก์ไปยังหน้าต่าง ๆ ของ React

React Router

The screenshot shows the React Router v6 documentation page at <https://reactrouter.com/en/main>. The page has a dark theme with white text and features several sections:

- Getting Started**: Includes links to Feature Overview, Tutorial, Examples, FAQs, and Main Concepts.
- Upgrading**: Includes links to Migrating to RouterProvider, Upgrading from v5, and Migrating from @reach/router.
- Routers**: Lists various router components: Picking a Router, createBrowserRouter, createHashRouter, createMemoryRouter, createStaticHandler, createStaticRouter, RouterProvider, and StaticRouterProvider.
- Router Components**: Lists components: BrowserRouter, HashRouter, MemoryRouter, NativeRouter, Router, and StaticRouter.
- Route**: Lists components: Route, action, and errorElement.

The main content area contains four cards:

- What's New in 6.4?**: Describes v6.4 as the most exciting release yet with new data abstractions for reads, writes, and navigation hooks to easily keep your UI in sync with your data. It includes a link to the Feature Overview.
- I'm New**: Introduces the tutorial, which quickly introduces primary features like configuring routes, loading and mutating data, and pending and optimistic UI. It includes a link to the tutorial.
- I'm on v5**: Provides a migration guide to help users migrate incrementally and keep shipping along the way. It includes a link to the migration guide.
- I'm Stuck!**: Offers common questions about React Router v6, with a link to explore them.

At the bottom, there is a footer with links to © Remix Software, Inc., Brand, Docs and examples CC 4.0, and a GitHub icon.

React Router

main ↺ ↻ Made by Remix ↻

Tutorial

Welcome to the tutorial! We'll be building a small, but feature-rich app that lets you keep track of your contacts. We expect it to take between 30-60m if you're following along.

On this page

- Setup
- Adding a Router
- The Root Route
- Handling Not Found Errors
- The Contact Route UI
- Nested Routes
- Client Side Routing
- Loading Data
- Data Writes + HTML Forms
- Creating Contacts
- URL Params in Loaders
- Updating Data
- Updating Contacts with FormData
- Mutation Discussion
- Redirecting new records to the edit page
- Active Link Styling
- Global Pending UI
- Deleting Records
- Contextual Errors
- Index Routes
- Cancel Button
- URL Search Params and GET Submissions
- GET Submissions with Client Side Routing
- Synchronizing URLs to Form State
- Submitting Forms onChange
- Adding Search Spinner
- Managing the History Stack
- Mutations Without Navigation
- Optimistic UI
- Not Found Data
- Pathless Routes
- JSX Routes

👉 Every time you see this it means you need to do something in the app!

The rest is just there for your information and deeper understanding. Let's get to it.

Setup

NOTE If you're not going to follow along in your own app, you can skip this section

We'll be using Vite for our bundler and dev server for this tutorial. You'll need Node.js installed for the `npm` command line tool.

👉 Open up your terminal and bootstrap a new React app with Vite:

```
npm create vite@latest name-of-your-project -- --template react
# follow prompts
```

ทำการติดตั้ง React Router ถ้าใน Windows ไม่ต้องใส่ sudo

```
sudo npm install react-route
```

การทำ React Router เราจะต้องทำการ **export “wrapped”** App component

ทำการ import HashRouter

```
1 import React from 'react'  
2 import ReactDOM from 'react-dom/client'  
3 import { HashRouter } from 'react-router-dom'  
4 import App from './App.tsx'  
5 import './index.css'
```

ครอบส่วน App ด้วย HashRouter

```
7  ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(  
8    <React.StrictMode>  
9      <HashRouter>  
10        <App />  
11      </HashRouter>  
12    </React.StrictMode>,  
13  )
```

หรือในกรณีเราสามารถนำไปใส่ใน App.tsx ได้

src >  App.tsx >  App

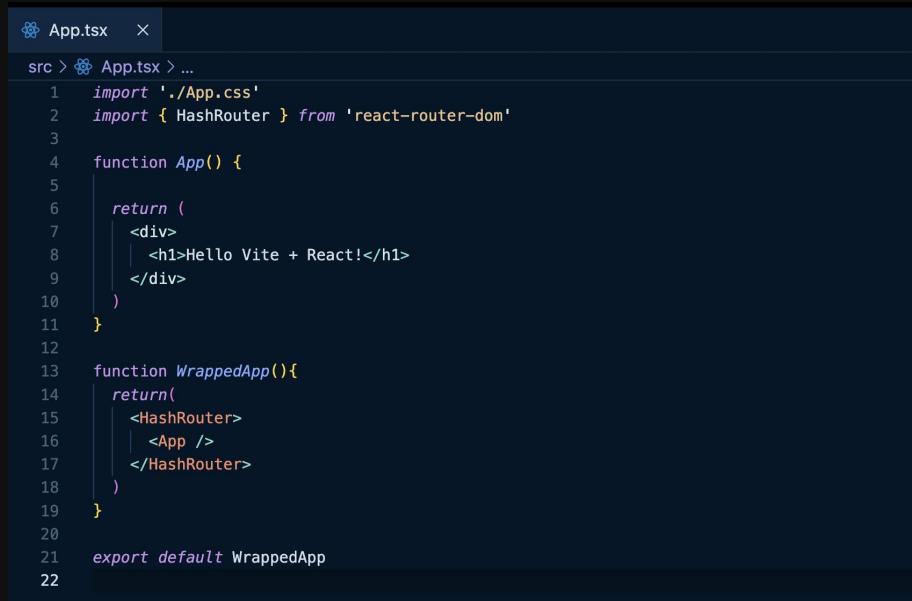
```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5 import { HashRouter } from 'react-router-dom'
6
```

และหากเราใส่ใน App.tsx เรา ก็สามารถสร้างฟังก์ชัน ขึ้นมาเพื่อ Wrap ตัว App ได้ เช่น กัน

```
function WrappedApp(){
  return(
    <HashRouter>
      <App />
    </HashRouter>
  )
}
```

```
export default WrappedApp
```

Code หลักของเราง่ายได้ลักษณะประมาณนี้



```
App.tsx
src > App.tsx > ...
1 import './App.css'
2 import { HashRouter } from 'react-router-dom'
3
4 function App() {
5
6   return (
7     <div>
8       <h1>Hello Vite + React!</h1>
9     </div>
10  )
11}
12
13 function WrappedApp(){
14  return(
15    <HashRouter>
16      <App />
17    </HashRouter>
18  )
19}
20
21 export default WrappedApp
22
```

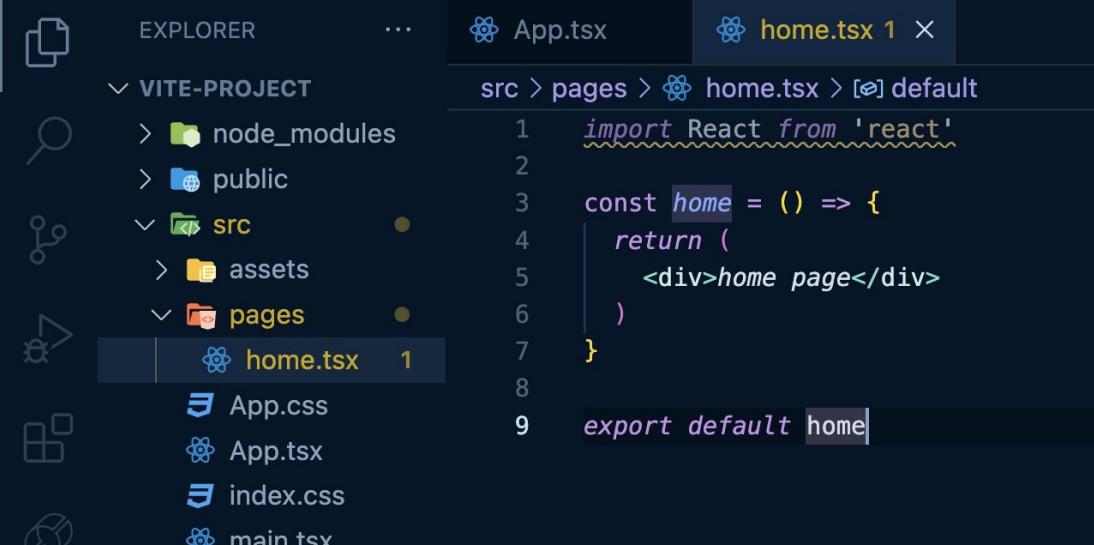
ขั้นตอนต่อมา เราจะมาทำการสร้าง Routes หรือ ตัวที่ไว้ระบุ Path ต่าง ๆ ในหน้าเว็บเรา แต่อย่าลืม ! import ของที่ต้องใช้มาด้วยจ้า

```
src > ⚛ App.tsx > 📂 WrappedApp
1  ✓ import './App.css'
2    import { HashRouter, Route, Routes } from 'react-router-dom'
3
4  ✓ function App() {
5
6    ✓ return (
7      ✓ <div>
8        <h1>Hello Vite + React!</h1>
9        ✓ <Routes>
10          ✓ <Route path="/" />
11        ✓ </Routes>
12      ✓ </div>
13    )
14 }
```

ສໍາຮັບ Path = "/" ຈະເປັນ root ຂອງ ໄຫ້ຮັກນິ້ນເອງ

```
src > App.tsx > WrappedApp
1  import './App.css'
2  import { HashRouter, Route, Routes } from 'react-router-dom'
3
4  function App() {
5
6    return (
7      <div>
8        <h1>Hello Vite + React!</h1>
9        <Routes>
10          <Route path="/" />
11        </Routes>
12      </div>
13    )
14 }
```

ก่อนที่เราจะสร้าง Route เพื่อลิงก์กับเราไปสร้างหน้าใหม่กันก่อน ให้เราสร้างใน โฟลเดอร์ **pages/home.tsx** กรณีไม่มีสร้างใหม่ได้เลย



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

- EXPLORER** sidebar: VITE-PROJECT > node_modules, public, src (expanded), assets, pages (expanded), home.tsx (selected).
- EDITOR**:
 - File: App.tsx
 - File: home.tsx (selected)
- Code Editor Content (home.tsx):**

```
src > pages > home.tsx > default
1 import React from 'react'
2
3 const home = () => {
4   return (
5     <div>home page</div>
6   )
7 }
8
9 export default home
```

และ ทำการสร้าง Route ได้เลย โดยใน element ให้เราใส่หน้าที่ต้องการไว้

```
src > App.tsx > ...
1 import './App.css'
2 import { HashRouter, Route, Routes } from 'react-router-dom'
3 import Home from './pages/Home'
4
5 function App() {
6
7   return (
8     <div>
9       <h1>Hello Vite + React!</h1>
10      <Routes>
11        <Route path="/" element={<Home/>}/>
12      </Routes>
13    </div>
14  )
15}
```

กรณีที่มีหน้าอื่น ๆ เช่น หน้า Error แล้วเราจะจัดการก็สามารถ Custom เองได้ลักษณะนี้ และ ลองสร้างหน้า NotFound ได้ในลักษณะเดียวกัน

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure:

- VITE-PROJECT
 - node_modules
 - public
 - src
 - assets
 - pages
 - Home.tsx
 - NotFound.tsx (selected)
- App.css

The main editor area shows the content of `NotFound.tsx`:

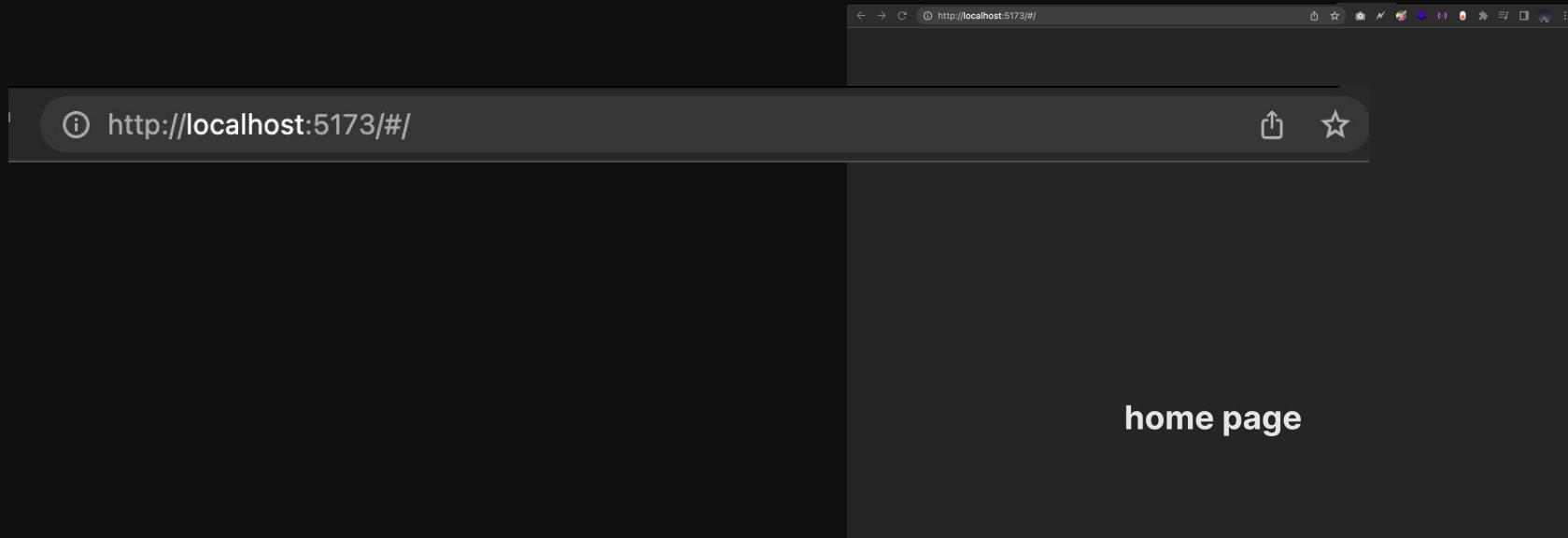
```
src > pages > NotFound.tsx > [!] NotFound
1 import React from 'react'
2
3 const NotFound = () => {
4   return (
5     <div>NotFound</div>
6   )
7 }
8
9 export default NotFound
```

ແຕ່ຈະແຕກຕ່າງກັນທີ ຮະບຸເປີນ * ແກ່ນໜີ້ເອງ

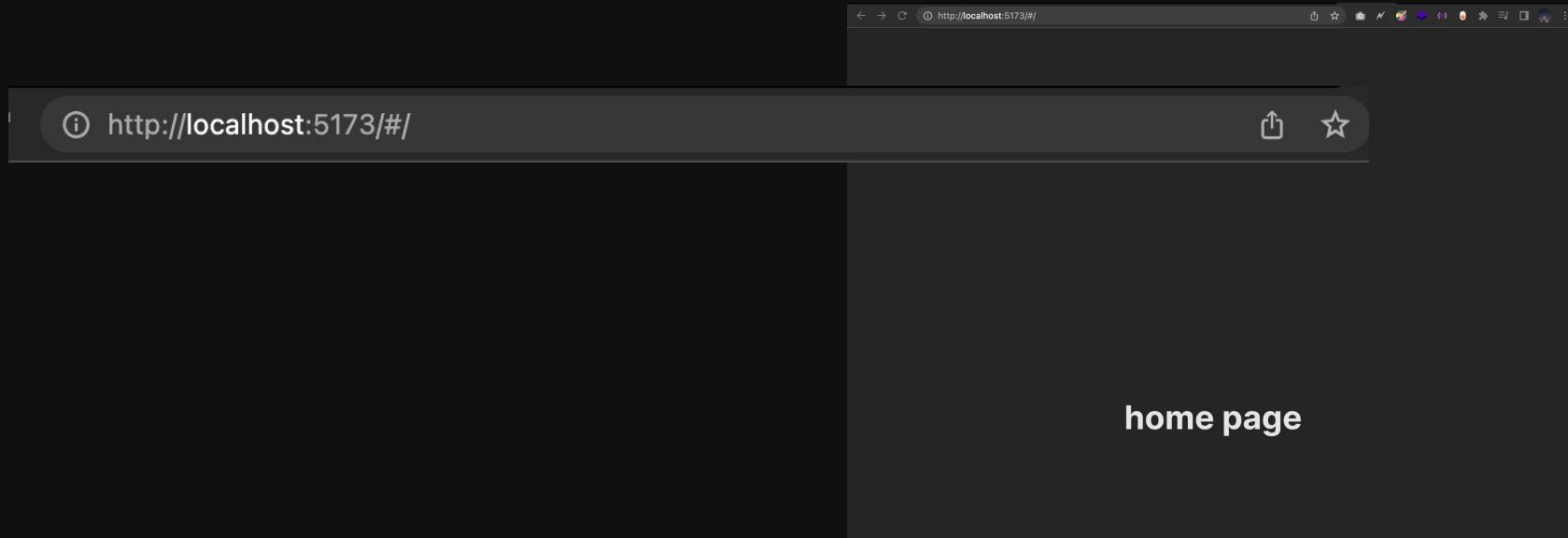
```
1 import './App.css'
2 import { HashRouter, Route, Routes } from 'react-router-dom'
3 import Home from './pages/Home'
4 import NotFound from './pages/NotFound'
5
6 function App() {
7
8     return (
9         <div>
10            <h1>Hello Vite + React!</h1>
11            <Routes>
12                <Route path="/" element={<Home/>}/>
13                <Route path='*' element={<NotFound/>} />
14            </Routes>
15        </div>
16    )
17 }
```

แล้วลองทดสอบกันดู

គារន៍ចិងគីតូទូនខ្លាងជាមុន /#/ កំណត់បន្ថែម



ตัวนี้จะเรียกว่า HashRouter



ปกติแล้วเราจะ用 Router หลัก ๆ อยู่ 2 ตัว ก็คือ HashRouter กับ BrowserRouter

BrowserRouter

243



+50



It uses [history API](#), i.e. it's unavailable for legacy browsers (IE 9 and lower and contemporaries). Client-side React application is able to maintain clean routes like `example.com/react/route` but needs to be backed by web server. Usually this means that web server should be configured for single-page application, i.e. same `index.html` is served for `/react/route` path or any other route on server side. On client side, `window.location.pathname` is parsed by React router. React router renders a component that it was configured to render for `/react/route`.

Additionally, the setup may involve server-side rendering, `index.html` may contain rendered components or data that are specific to current route.

HashRouter

It uses URL hash, it puts no limitations on supported browsers or web server. Server-side routing is independent from client-side routing.

Backward-compatible single-page application can use it as `example.com/#/react/route`. The setup cannot be backed up by server-side rendering because it's / path that is served on server side, `#/react/route` URL hash cannot be read from server side. On client side, `window.location.hash` is parsed by React router. React router renders a component that it was configured to render for `/react/route`, similarly to `BrowserRouter`.

Most importantly, `HashRouter` use cases aren't limited to SPA. A website may have legacy or search engine-friendly server-side routing, while React application may be a widget that maintains its state in URL like `example.com/server-side/route#/react/route`. Some page that contains React application is served on server side for `/server-side/route`, then on client side React router renders a component that it was configured to render for `/react/route`, similarly to previous scenario.

Ref :

<https://stackoverflow.com/questions/51974369/what-is-the-difference-between-hashrouter-and-browser-router-in-react>

BrowserRouter

Uses everything after the domain(eg .com, .in, .io etc) as the path. For example, if the url is `localhost:3000/home`, then `/home` is the path/route. This router uses the [HTML 5 History API](#) to keep the UI in sync with the the path.

BrowserRouter is used for doing client side routing with URL segments. You can load a top level component for each route. This helps separate concerns in your app and makes the logic/data flow more clear. Let's see how this can be implemented in code using `react-router`:

```
1 import { BrowserRouter, Route } from 'react-router-dom';
2
3 function App() {
4   return (
5     <BrowserRouter>
6       <Route path="/home" exact component={Home} />
7     </BrowserRouter>
8   );
9 }
```

[browser_router_example.ts](#) hosted with ❤ by GitHub

[view raw](#)

We'll look at the advantages and disadvantages when we compare the routers later.

HashRouter

This is similar (visually) to the browser router, with the only difference being that a `#` is added between the domain and the path. Let's say the url is `localhost:3000/#/home`, then the `/home` is the route. Basically, it uses everything after the `#` as the path. Now, we'll see how to implement this:

```
1 import { HashRouter, Route } from 'react-router-dom';
2
3 function App() {
4   return (
5     <HashRouter>
6       <Route path="/home" exact component={Home} />
7     </HashRouter>
8   );
9 }
```

hash_router_example.ts hosted with ❤ by GitHub

[view raw](#)

We have an `App()` component in which we are returning a `HashRouter` with a route. If you run our little app in the browser, you will find out that react router automatically adds a `#` before the `path`.

① `localhost:3000/#/`

ແລະ ເລາຈຣິງ ຫຼື ຍັງມີອີກ 2 ຕັ້ງ

MemoryRouter

It keeps the history of your “URL” in memory (does not read or write to the address bar). It’s useful in tests and non-browser environments like [React Native](#).

```
1 import { MemoryRouter, Route } from 'react-router';
2
3 function App() {
4     return (
5         <MemoryRouter>
6             <Route path="/home" exact component={Home} />
7         </MemoryRouter>
8     );
9 }
```

memory_router_example.ts hosted with ❤ by GitHub

[view raw](#)

The component will change when you click on a link but the url will remain the same

StaticRouter

This is useful in server-side rendering scenarios when the user isn't actually clicking around, so the location never actually changes. Hence, the name:

`static`. It's also useful in simple tests when you just need to plug in a location and make assertions on the render output.

• • •

Comparing Routers

- Browser Router

The browser router is often very complicated to deploy to servers. It uses [history API](#), so it's unavailable for legacy browsers (IE 9 and lower and contemporaries.. but who cares about that those, we are modern folks here 😎).

Client-side React application is able to maintain clean routes like `example.com/react/route` but needs to be backed by a web server. A traditional web server evaluates requests like this:



In case of a single page application, it's just a single `index.html` file which requires some extra configuration on the hosting server.

However, these days a lot of hosting providers assume that we use this kind of routing and simplify the process.

We can also use server side rendering with this router, `index.html` may contain rendered components or data that are specific to current route.

- Hash Router

Enough of browser routers, let's talk hash routers. We can quite easily configure our server to ignore the `#` and serve the entire html file regardless of the route. This hash will be used by our routes to show the appropriate content.

Also, `HashRouter` use cases aren't limited to SPA. A website may have legacy or search engine-friendly server-side routing, while React application may be a widget that maintains its state in URL like `example.com/server side/route#/react/route`.

- Memory Router

Memory routers will refresh/change the component but no the url. This probably will leave the user hanging and he or she might think that his computer has hanged or there is an issue with the site which is certainly not pleasant for the user.

On the other hand, this is great for environments in which the url is not shown like React Native, Testing Environments etc

- Static Router

Static routing can be used to define an exit point from a router when no other routes are available or necessary which is called a `default route`. Think of this as a fallback route(404s). Static routing can be used for small networks that require only one or two routes.

กรณีที่เราต้องการสร้างหน้าคัดไป ?

ให้เราสร้าง About.tsx ใน pages เมื่อตอนเดิม

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

- EXPLORER** sidebar: Shows a project structure with **vite-project**, **VITE-P...**, **node_modules**, **public**, **src** (expanded), **assets**, **pages** (expanded), **About.tsx**, **Home.tsx**, and **NotFound...**. The **About.tsx** file is selected.
- EDITOR**: The **About.tsx** file is open, showing the following code:

```
import React from 'react'
const About = () => {
  return (
    <div>About</div>
  )
}
export default About
```

A yellow lightbulb icon is positioned above the opening brace of the `return` block, indicating a potential issue or suggestion.
- STATUS BAR**: Shows the path `src > pages > About.tsx` and the text `[e] About`.

ຕ່ອມາໃນ App.tsx ກີ່ກຳໃນລັກຜະນະເດີມນັ້ນເອງ

```
import './App.css'
import { HashRouter, Route, Routes } from 'react-router-dom'
import Home from './pages/Home'
import NotFound from './pages/NotFound'
import About from './pages/About'

function App() {

    return (
        <Routes>
            <Route path="/" element={<Home/>} />
            <Route path="*" element={<NotFound/>} />
            <Route path="/about" element={<About/>} />
        </Routes>
    )
}
```

MUI

The screenshot displays the Material-UI (MUI) website homepage. At the top, there's a navigation bar with the MUI logo, links for Products, Docs, Pricing, About us, and Blog, and a search bar. To the right of the search bar are icons for dark mode, refresh, and settings.

The main content area features a large blue header with the text "Move faster with intuitive React UI tools". Below this, a section describes MUI's comprehensive suite of UI tools, mentioning Material UI, a design system, and production-ready components. It includes a "Get started" button and a command-line interface snippet for installing MUI.

On the right side, there are several examples of MUI components:

- A modal window titled "March 25th" containing a code editor icon, the text "Check the docs for getting every component API", and a user profile for "Michael Scott" with a progress bar at 60%.
- A calendar for "May 2023" showing the days of the week and specific dates like the 25th.
- A media player component titled "Ultraviolet" showing a thumbnail of a person sitting under a tree, playback controls, and a track list.
- A grid component showing a 2x2 grid of items with labels like "React", "TypeScript", "CSS", and "JavaScript".
- A slider component showing a vertical scale from 25°C to 50°C.

At the bottom, there are logos for Spotify, Amazon, NASA, Netflix, Unity, and Shutterstock, followed by a testimonial: "The world's best product teams trust MUI to deliver an unrivaled experience for both developers and users."

วิธีการใช้งาน MUI

1. สร้างโปรเจกต์ด้วย Vite
2. npm install vite
3. ลองทดสอบ npm run dev
4. ใช้คำสั่ง npm install @mui/material
@emotion/react @emotion/styled
5. ทดสอบกับโค้ด Demo (หน้าคัดไป)

```
import { useState } from 'react';
import reactLogo from './assets/react.svg';
import './App.css';
import { Button } from '@mui/material';

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

  return [
    <div className="App">
      <div>
        <a href="https://vitejs.dev" target="_blank">
          
        </a>
        <a href="https://reactjs.org" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React + + Typescript + MUI 5</h1>
      <Button color="secondary">Secondary</Button>
      <Button variant="contained" color="success">
        Success
      </Button>
      <Button variant="outlined" color="error">
        Error
      </Button>

      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.tsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </div>
  ];
}

export default App;
```

ควรดื่มน้ำวันละเท่าไหร่ ?

x มล.

น้ำหนักของคุณ (กิโลกรัม)



ทำโปรแกรมคำนวณปริมาณน้ำที่ต้องดื่มต่อวันโดยคำนวณทันทีเมื่อกรอกข้อมูลลงไป และ สามารถสลับรีมได้

ควรดื่มน้ำวันละเท่าไหร่ ?

x มล.

น้ำหนักของคุณ (กิโลกรัม)

คำนวณ

🌙 Dark ▽

ทำโปรแกรมคำนวณปริมาณน้ำที่ต้องดื่มต่อวันโดยคำนวณ **เมื่อくだปุ่ม** และ สามารถลับรีมได้

ควรดื่มน้ำวันละเท่าไหร่ ?

x มล.

น้ำหนักของคุณ (กิโลกรัม)



Light



ทำโปรแกรมคำนวณปริมาณน้ำที่ต้องดื่มต่อวันโดยคำนวณทันทีเมื่อกรอกข้อมูลลงไป และ สามารถสลับรีมได้

ควรดื่มน้ำวันละเท่าไหร่ ?

x มล.

น้ำหนักของคุณ (กิโลกรัม)



นำ Project Upload ขึ้น Repository ของ GitHub
แล้วส่งลิงก์มาในส่งการบ้านได้เลย *อย่าลืมเปิด Public*