ROUTING & STATE MANAGEMENT IN REACT

ROUTING

LOADING A PAGE IN A BROWSER

Browser

HTML

Other Resources

http://creativecommons.org

HTTP GET

```
http://creativecommons.org
<a><span id="home-button">
</span></a>
<div id="logo">
 <span>
   Creative Commons
 </span>
```

HTTP GET

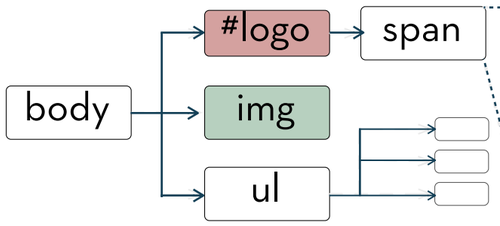
```
cforms.js
//Collap
String.p
function
  return
this.rep
```

```
creativecommons.css
topbar #home-button{
 position: relative;
 float: lef
 display: b
 height: 40
 width: 150
```

cc-logo.png

Rendered Page

Document Object Model (DOM)




```
topbar span {
  float: left;
  display: block;
  height: 40px;
  width: 150px;
  cursor:
pointer;
  z-index: 1;
  top: 0;
```

WHAT IS ROUTING

In SPAs (Single-Page Applications), routing is used for **changing the display** when the URL changes

Use **History API** to update the URL and change the active component to match the URL

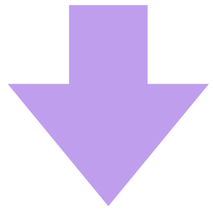
No real page reload

Gives the users the illusion of **navigating between multiple pages**

Users can **share direct URLs** to app states

http://creativecommons.org/

Home Component



http://creativecommons.org**/about**

About Component

REACT ROUTER

1. Install React Router to your React project

```
> npm install react-router-dom
```

2. Use components from the API

- `BrowserRouter`
- `Switch`
- `Route`
- `Link`

https://reactrouter.com/web/guides/quick-start

BrowserRouter

Uses History API to keep your UI in sync with the URL

```
<BrowserRouter>
  <App />
</BrowserRouter>
```

Wrap your your whole app on the top level to use!

https://reactrouter.com/web/api/BrowserRouter

Switch and Route

```
<Switch>
  <Route exact path="/">
    <Home />
  </Route>
  <Route path="/about">
    <About />
  </Route>
  <Route path="/dashboard">
    <Dashboard />
  </Route>
</Switch>
```

Use them together to declare your routes

https://reactrouter.com/web/api/Switch

Link

```
<Link to="/">Home</Link>
<Link to="/dashboard">Dashboard</Link>
<Link to="/about">About</Link>
```

Use it to create references to your routes

https://reactrouter.com/web/api/Link

Basic Routing

Example

URL Params

Example

Nested Routing

Example

HOOKS

HOOKS

*A Hook is a special function that lets you "**hook into**" React features.*

*Hooks let you use **state** and **other React features without writing a class**. You can also **build your own Hooks** to share reusable stateful logic between components.*

MOTIVATION

1

It’s hard to reuse stateful logic between components

2

Complex components become hard to understand

3

Classes confuse both people and machines

STATE HOOK: useState

```
 1:  import React, { useState } from 'react';
 2:
 3:  function Example() {
 4:    const [count, setCount] = useState(0);
 5:
 6:    return (
 7:      <div>
 8:        <p>You clicked {count} times</p>
 9:        <button onClick={() => setCount(count + 1)}>
10:         Click me
11:        </button>
12:      </div>
13:    );
14:  }
```

EFFECT HOOK: `useEffect`

```jsx
import React, { useState, useEffect } from 'react';

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

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

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

RULES

Only Call Hooks at the Top Level

Don’t call Hooks inside loops, conditions, or nested functions.

This ensures that Hooks are called in the same order each time a component renders.

Only Call Hooks from React Functions

Don’t call Hooks from regular JavaScript functions. Instead, you can:

- Call Hooks from React function components.
- Call Hooks from custom Hooks

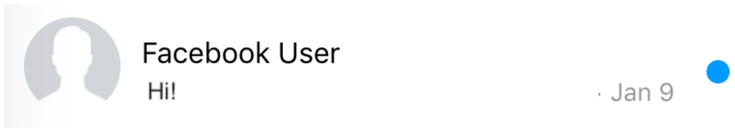
This ensures that all stateful logic in a component is clearly visible

STATE MANAGEMENT

STATE MANAGEMENT

Different components might want to **display and update the same state**

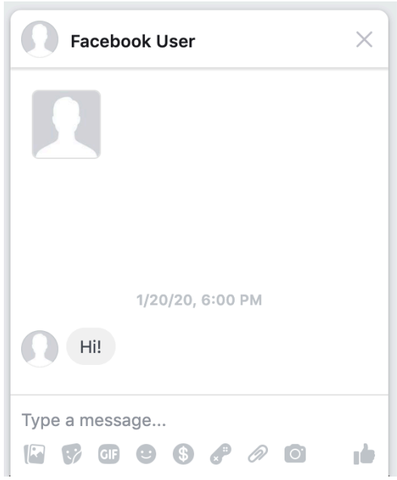
Lifting the state up might not be always feasible.



Shared global state enables:

Single source of truth

Single point of mutation



CORE CONCEPTS

State

The data of your app.

Global values that are shared by multiple components.

Think of it like a front-end data store.

Actions

Piece of code that indicates change requests the state.

Can contain a "payload"

Reducers/ Derivations

Handle how state changes as a response to actions.

TOOLS

Use read-only state that can only be updated through actions.

Reducers handle the updates and create a new state

Define observable states and views that update with respect to states

Views use derivations to update the state

Built-in solutions with React

Easy to use

MANAGING STATE WITH HOOKS

```
function useReducer(reducer, initialState) {
  const [state, setState] = useState(initialState);

  function dispatch(action) {
    const nextState = reducer(state, action);
    setState(nextState);
  }

  return [state, dispatch];
}
```

Local State → **useState**

Global State → **useReducer**

Derived Properties → **custom hooks**

MANAGING STATE WITH HOOKS

```
const initialState = {count: 0};

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return {count: state.count + 1};
    case 'decrement':
      return {count: state.count - 1};
    default:
      throw new Error();
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);
  return (
    <>
      Count: {state.count}
      <button onClick={() => dispatch({type: 'decrement'})}>-</button>
      <button onClick={() => dispatch({type: 'increment'})}>+</button>
    </>
  );
}
```

RESOURCES

History API - https://developer.mozilla.org/en-US/docs/Web/API/History_API

React Router Quick Start - https://reactrouter.com/web/guides/quick-start

React Hooks - https://reactjs.org/hooks

React Hooks Announcement - https://www.youtube.com/watch?v=dpw9EHDh2bM

Redux - https://redux.js.org/

MobX - https://mobx.js.org/

Recoil - https://recoiljs.org/

NEXT CLASS: DATABASES

https://uiuc-web-programming.gitlab.io/fa21/