

# REACT ROUTER

*Declarative routing for React*

# REVIEW

# NOT SINGLE PAGE APPLICATION



# Server



index.html

GET /



User clicks on link . . .



# Server



settings.html

GET /settings



# NOT SINGLE PAGE APPLICATIONS

- **Views stored on the server, served up as HTML pages.**
- **When user goes to a new page, the browser navigates in totality, navigating, refreshing and retrieving a brand new HTML document.**
- **Each page, since it is a new page, retrieves stylesheets, script files, etc.**

# **SINGLE PAGE APPLICATION (SPA)**





# Server



index.html

GET /



User clicks on link . . .





# Server

```
{  
  "billingInfo": {},  
  "notifyMe": true  
}
```

GET /user/1/settings



## Display settings view

# SINGLE PAGE APPLICATIONS

- On page change, a new page is not loaded. The front-end JavaScript replaces elements on existing DOM to update view.
- **AJAX** plays a big part to fill in the data
- **Two options for dealing with the url bar:**
  - Use the **HTML5 Browser History API**, OR
  - Manipulate the **Document Fragment Identifier (#)**



# WHAT IS REACT-ROUTER?

- **React Router keeps your UI in sync with the URL.**
- **Ties into URL and history to allow for easy navigation to and between different parts of your application.**
- **Easily integrates nesting of components.**

# INGREDIENTS

# INGREDIENT LIST

- **A Router (either HashRouter or BrowserRouter)**
- **Routes**
- **Links**



# **“ROUTER” (HASH ROUTER OR BROWSER ROUTER)**

# ROUTER

- **Answers simple question: *Use the BrowserHistory API, or abuse the Document Fragment Identifier (#)***
- ***Needs to be the parent of almost everything***
  - *Why? Because whenever the url changes, it's the Router component that changes its state, which causes everything beneath it to re-render*

```
import {HashRouter as Router} from 'react-router-dom'
```

```
import {HashRouter as Router} from 'react-router-dom'

const Main = () => {
  return (
    <Router>
      {/* basically everything else */}
    </Router>
  )
}
```

**ROUTE = PATH + COMPONENT**

# ROUTE = PATH + COMPONENT

```
<Route path='/somePath' component={SomeComponent} />
```



**WHEN THE URL BAR MATCHES THE ONE  
SPECIFIED IN THE ROUTE, REACT-ROUTER  
CAUSES THAT COMPONENT'S RENDER  
FUNCTION TO EXECUTE**

**THIS MATCHING IS  
"FUZZY" (THINK "JUST LIKE  
APP.USE!")**

# LINK

## LIKE AN "A" TAG BUT BETTER

# LINK

```
<Link to='somePath'>Go to SomePath</Link>
```

