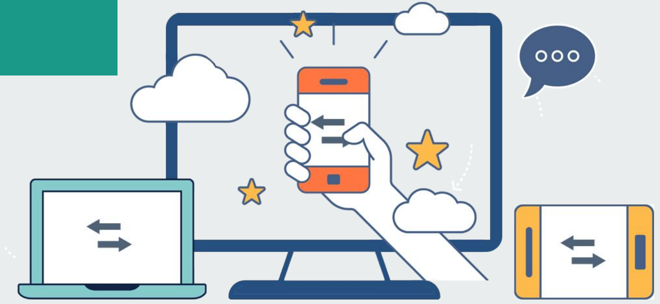
-Lecture 10-

Chapter 7– **React Router**

Adil **CHEKATI**, PhD

adil.chekati@univ-constantine2.dz



Prerequisites

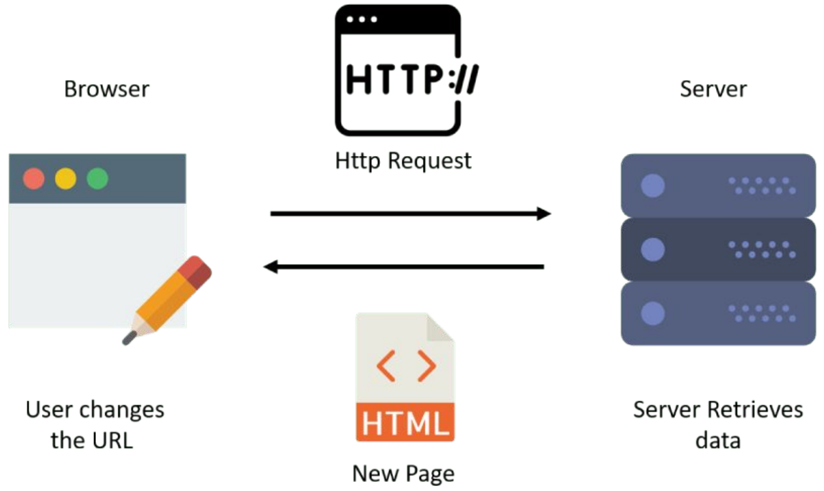
- ❏ Basic Understanding of React Fundamentals (Knowledge)
- ❏ Understanding of Node.js and npm (Application)

React Router

Objectives

- ➔ Describe client-side routing and its usefulness.
- ➔ Compare client-side routing to server-side routing.
- ➔ Implement basic client-side routing with React Router.

Introduction



Conventional Routing: When the user types an URL in the browser, an HTTP request is sent to the server, which then retrieves the HTML page.
For each new URL, the user is redirected to a new HTML page.

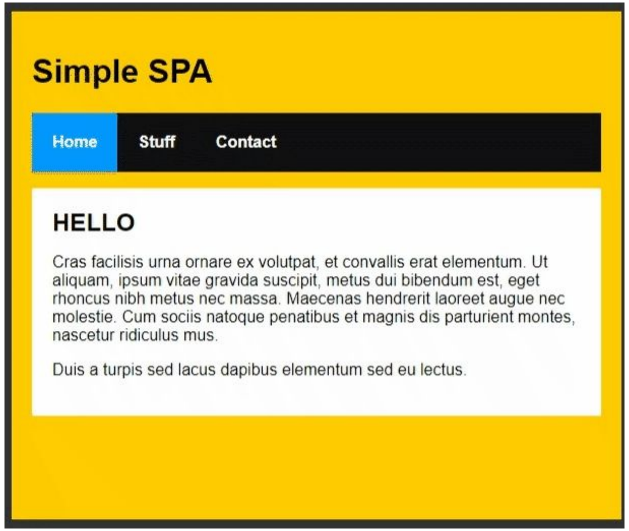
Introduction

➢ **Why Do We Need A React Router?**

Having a single page application limited to **single view** does not do justice to application which renders multiple views using React.

We need to move ahead and learn how to display **multiple views** in single page application.

Introduction



HELLO

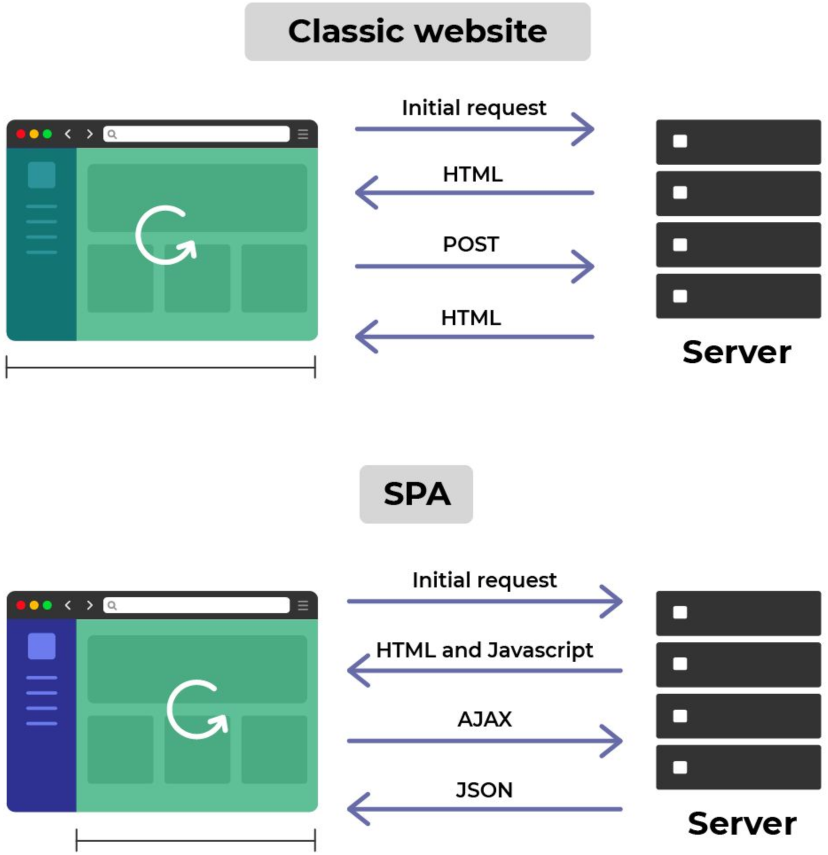
Cras facilisis urna ornare ex volutpat, et convallis erat elementum. Ut aliquam, ipsum vitae gravida suscipit, metus dui bibendum est, eget rhoncus nibh metus nec massa. Maecenas hendrerit laoreet augue nec molestie. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Duis a turpis sed lacus dapibus elementum sed eu lectus.

*Adding a **router library** into our application solves this requirement.*

Introduction

- ➔ In React, there is only a single 'HTML' file involved.
- ➔ Whenever a user types in a new URL request, instead of fetching data from the server, the Router swaps in a different Component for each new URL request.
- ➔ The user is tricked into switching among multiple pages but in reality, each separate Component re-renders achieving multiple views as per our needs.



1 What is React Router?

REACT ROUTER

- ❑ React Router is a powerful library that allows us to handle routing in React applications.
- ❑ It provides an easy and declarative way to define application navigation and render different components based on the URL.
- ❑ Routing is an essential part of a single-page application (SPA) as it helps users navigate between different views without refreshing the page.

React Router simplifies this process by providing a set of components and functionalities to manage routing seamlessly within a React application.

1.1 Why use React Router?

React Router offers several benefits that make it a popular choice for handling routing in React applications:

a) Declarative routing

Follows a declarative approach, where we define the routing configuration using components and their properties. This makes it easier to understand and maintain the navigation logic of our application.

b) Component-based routing

Associate specific components with different routes. When a user navigates to a particular URL, React Router will render the associated component, providing a smooth and seamless user experience.

c) Nested routing

Supports nested routing, allowing us to create complex application structures with multiple levels of routing. This feature is particularly useful when building larger applications with hierarchical views and sub-routes.

1.1 Why use React Router?

d) URL parameters

Enables us to define dynamic routes that include parameters in the URL. These parameters can be accessed and used within components, allowing for more flexible and dynamic routing behavior.

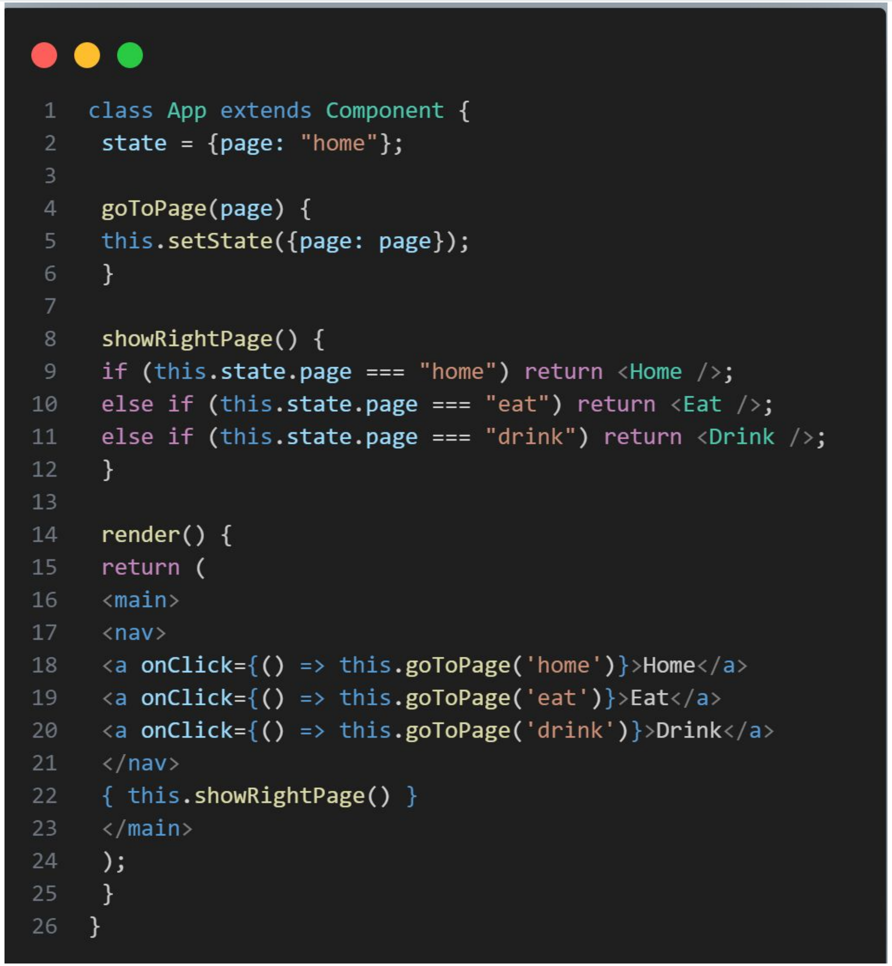
e) Programmatic navigation:

Provides a set of APIs to programmatically navigate between routes. This allows us to trigger route changes based on user interactions or certain application events, enhancing the overall user experience.

2. Client-Side Routing

- ➔ React router makes it possible to reproduce the operation of a classic web application as described above without requiring a server.

- ➔ To understand this concept, we'll use an *example of "artificial" client-side routing*



```
class App extends Component {
 state = {page: "home"};

 goToPage(page) {
 this.setState({page: page});
 }

 showRightPage() {
 if (this.state.page === "home") return <Home />;
 else if (this.state.page === "eat") return <Eat />;
 else if (this.state.page === "drink") return <Drink />;
 }

 render() {
 return (
 <main>
 <nav>
 <a onClick={() => this.goToPage('home')}>Home</a>
 <a onClick={() => this.goToPage('eat')}>Eat</a>
 <a onClick={() => this.goToPage('drink')}>Drink</a>
 </nav>
 { this.showRightPage() }
 </main>
 );
 }
}
```

2. Client-Side Routing

2.1. Artificial client-side routing

This type of artificial routing has a number of important features:

➔ The ability to display different "pages
- ◆ Remain at front-end level, without loading new pages from a server.

➔ But we **don't have** :
- ◆ Different URLs when navigating through the pages.
- ◆ The ability to use the browser's previous/next buttons.
- ◆ The ability to bookmark a page.
- ◆ More complex routing patterns..

2. Client-Side Routing

2.2. Real client-side routing

Client-side routing is used to manage the mapping between the URL contained in the browser's address bar and the content of the page displayed.

➔ *In this way, pages are displayed **from the client** and not from the server.*

- ❑ Websites that use client-side routing exclusively are known as **Single-Page Applications (SPAs).**
- ❑ This is made possible by using JavaScript to manipulate the browser's address bar, using the History API.

3. React Router

- ❑ To start using React Router in your React application, you need to install the `react-router-dom` package, which includes the necessary components and functionalities for web applications.

- ❑ You can install `react-router-dom` using npm or yarn by running the following command in your project directory:

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

Or

```
yarn add react-router-dom
```

- ❑ Once installed, you can import the necessary components from `react-router-dom` in your application files.

3. React Router

3.1. Basic Routing with React Router

React Router provides several components to set up basic routing in your application:

a) BrowserRouter

This component wraps your entire application and enables routing functionality.

b) Route

The `Route` component acts as a translation service between routes and components.
This component defines a route and associates it with a specific component.

c) Link

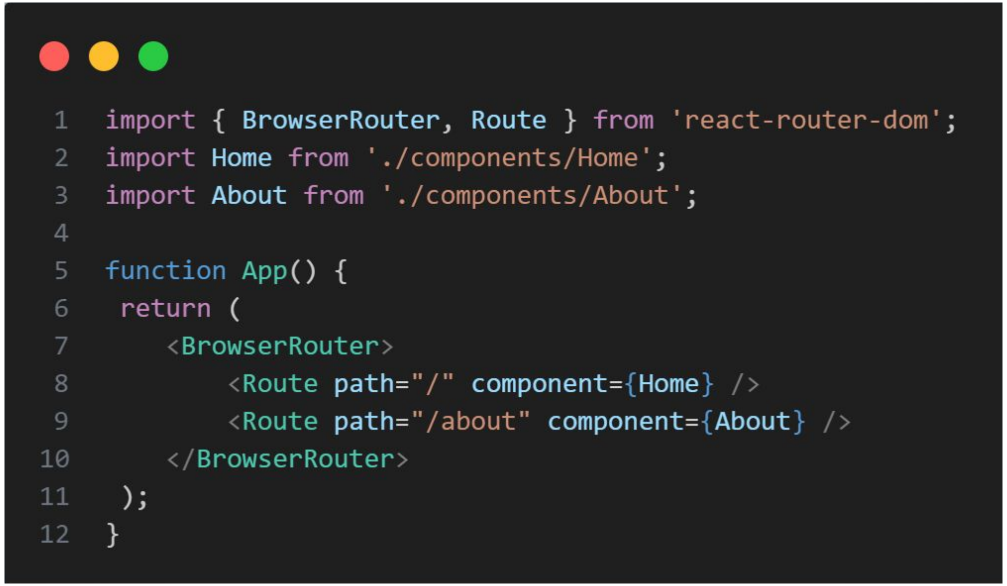
This component creates navigation links that trigger route changes when clicked.
It acts like an `<a>` element in HTML. Instead of the `href` attribute , the Link component uses a prop : `to`.

3. React Router

3.1. Basic Routing with React Router

To define routes and associated components, you can use the `Route` component within the `BrowserRouter` .

Example:



```
import { BrowserRouter, Route } from 'react-router-dom';
import Home from './components/Home';
import About from './components/About';

function App() {
 return (
    <BrowserRouter>
        <Route path="/" component={Home} />
        <Route path="/about" component={About} />
    </BrowserRouter>
 );
}
```

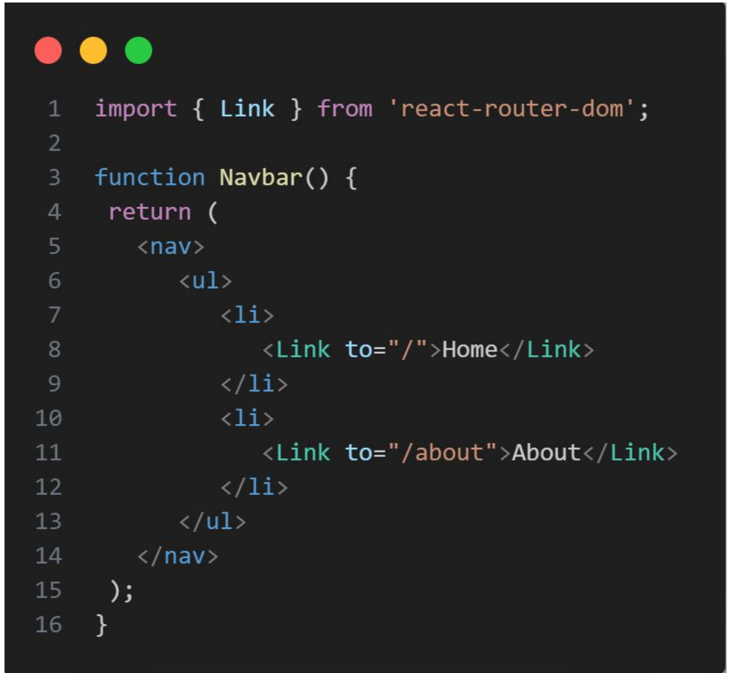
In this example, we define two routes using the `Route` component. The first route matches the root URL (/) and renders the Home component. The second route matches the /about URL and renders the About component.

3. React Router

3.1. Basic Routing with React Router

To create navigation links between routes, we can use the `Link` component.

Example:



```
import { Link } from 'react-router-dom';

function Navbar() {
 return (
   <nav>
      <ul>
         <li>
            <Link to="/">Home</Link>
         </li>
         <li>
            <Link to="/about">About</Link>
         </li>
      </ul>
   </nav>
 );
}
```

In this example, we create navigation links using the ***`Link`*** *component. When a link is clicked, React Router will handle the routing and render the associated component.*

3. React Router

3.1. Basic Routing with React Router

These are just the basics of React Router, and there is much more to explore and learn.

With React Router, you can easily implement complex routing logic and create dynamic and intuitive user experiences in your React applications.

4. Main rules for routing

➔ *React-Router provides client-side routing with the following properties:*

- ◆ *Browsing the application does not require loading from a server*
- ◆ *The address bar, bookmarks and page-back/next-page buttons work in the same way as in traditional web browsing.*

➔ *To do this, you need to*

- ◆ *Include the main application component* **`App`** *in a* **`BrowserRouter`** *component*
- ◆ *Use a different* **`Route`** *component for each route (representing a particular particular page of the application)*
- ◆ *For navigation through routes, use the* **`Link`** *component.*

5. Client-side vs. server-side routing

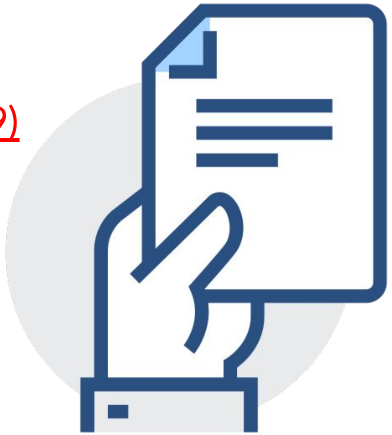
Client-side	Server-side
● Improved application interface (UI/UX) ● More modern architecture (leaner server) ● Problems of accessibility and search engine ranking, which may be poor or non-existent.	● Reload the page each time the URL is changed ● More traditional architecture (thin client, fat server) ● Better accessibility and ranking by search engines, better SEO (Search Engine Optimisation)

The decision whether to use client-side or server-side routing will depend on the application itself and the requirements it must meet (better UI or better SEO).

Mini Project Submission Guidelines

➔ **Deadline:**
At the end of each Lab session (no later than Saturday, January 6 at 23:59)
To: Classroom

➔ **Link to be submitted:**
Portfolio Link.



Textbook

➔ All academic materials will be available on:

Google Drive.
E-learning platform of Constantine 2 University.
Google Classroom.

aoa5lne

Google Classroom

جامعة قسنطينة 2 عبد الحميد مهري
خلية التعليم عن بعد

SCAN ME!

References

➔ **Book:**
Stoyan Stefanov -*React Up and Running: Building Web Applications - 2nd edition (2021).*

➔ **Online Resource:**
React Router Documentation
(https://reactrouter.com/en/main)

React Router v6: A Beginner's Guide
(https://www.sitepoint.com/react-router-complete-guide/)

Happy holidays

Adil **CHEKATI**, PhD

adil.chekati@univ-constantine2.dz

Questions,
& comments…

📧 adil.chekati@univ-constatine2.dz