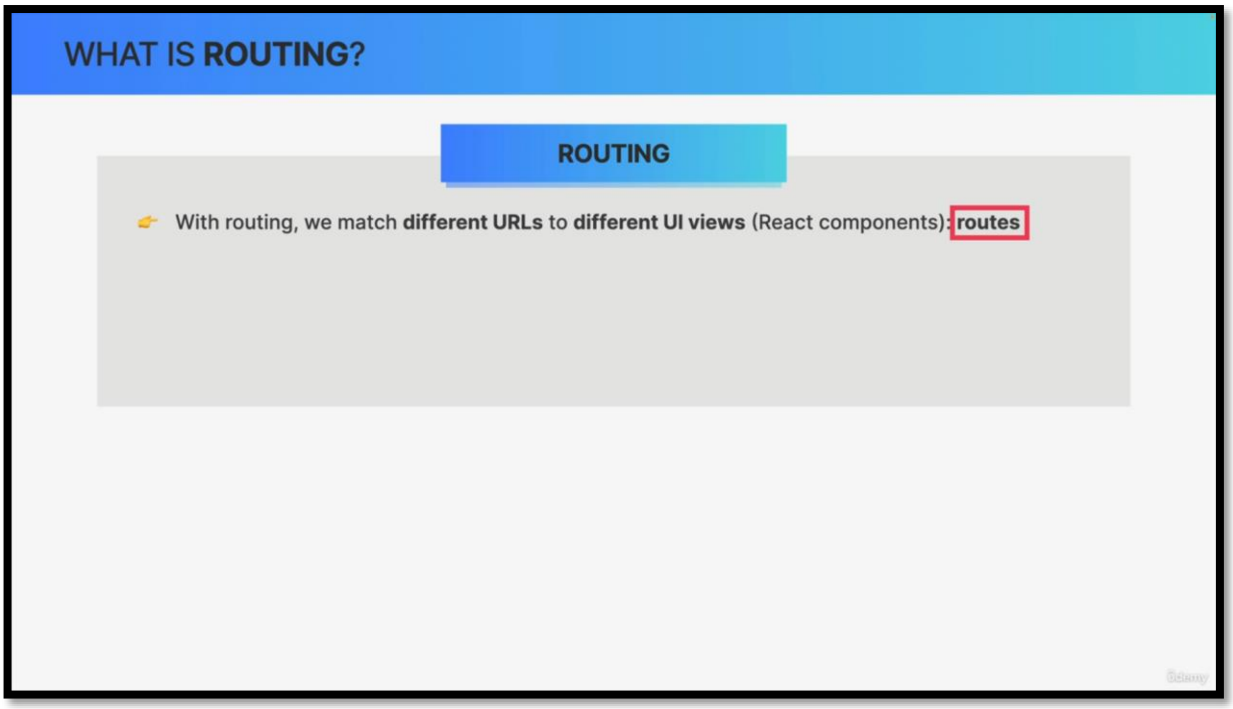
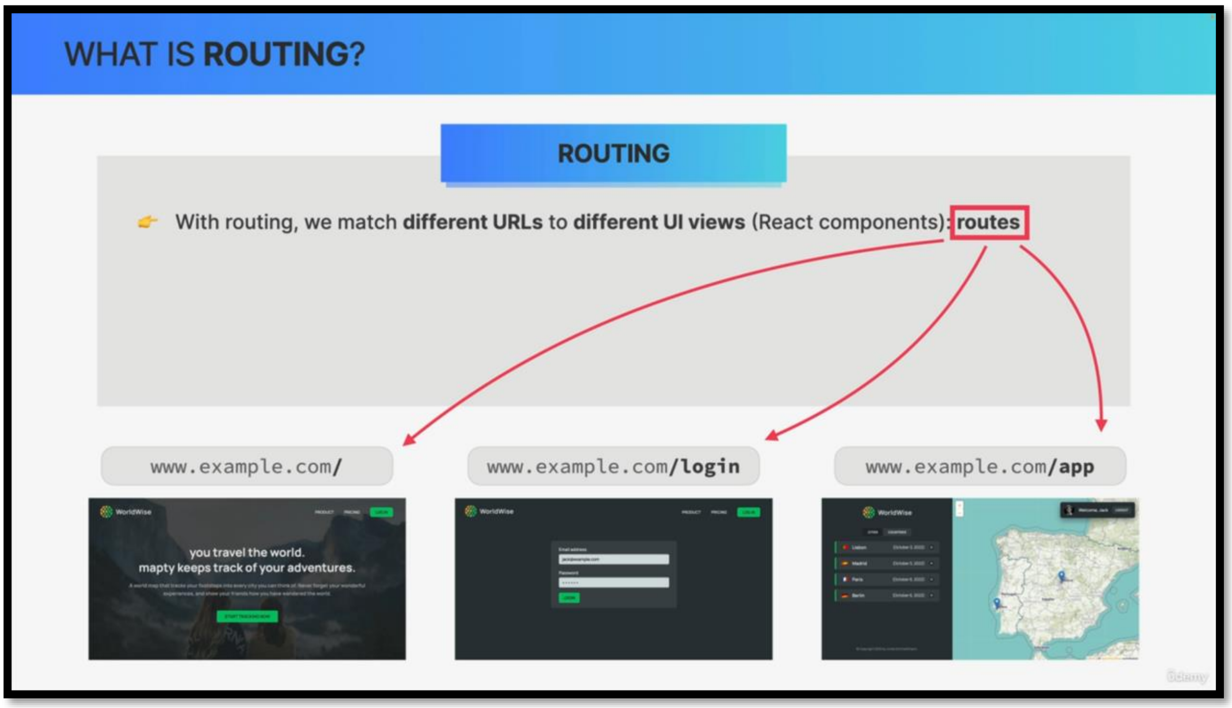

WHAT IS **ROUTING?**

ROUTING

With routing, we match **different URLs** to **different UI views** (React components): **routes**

When we use routing in a web application, we basically match different URLs to different views in the user interface. In the specific case of React, we match each URL to a specific React component, and we call each of these matches between a URL and a component, a route.

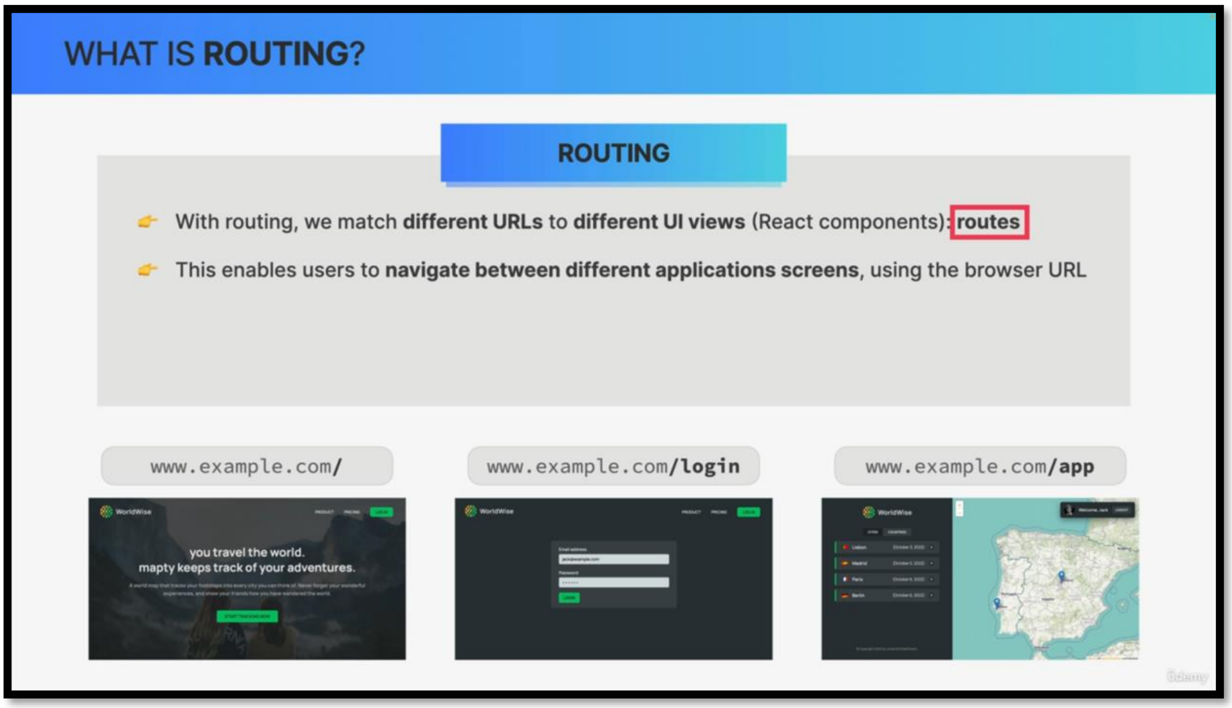

WHAT IS **ROUTING?**

ROUTING

With routing, we match **different URLs** to **different UI views** (React components): **routes**

When one of those specific URLs gets visited, the corresponding React component will be rendered for example, we could show this homepage, so this home component, whenever a user visits the route URL of example.com.

When they go to /login on the same URL, we then show the login component and if they log in successfully, we can then redirect them to the /app path, so to show them the app screen like this.



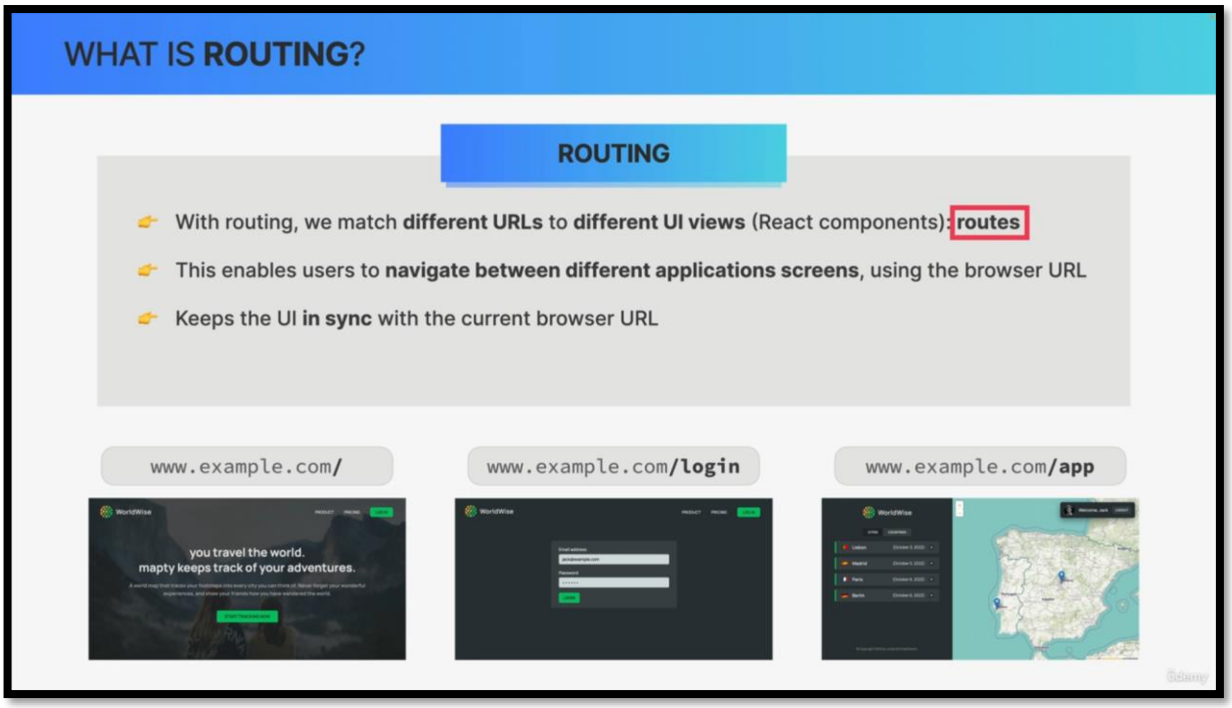
WHAT IS **ROUTING?**

ROUTING

- With routing, we match **different URLs** to **different UI views** (React components): **routes**
- This enables users to **navigate between different applications screens**, using the browser URL

In this example, we have three routes that the user can visit, the root URL, /login and /app which will all render different React components.

This enables users to navigate between different screens of the application by simply using links and the URL in the browser.

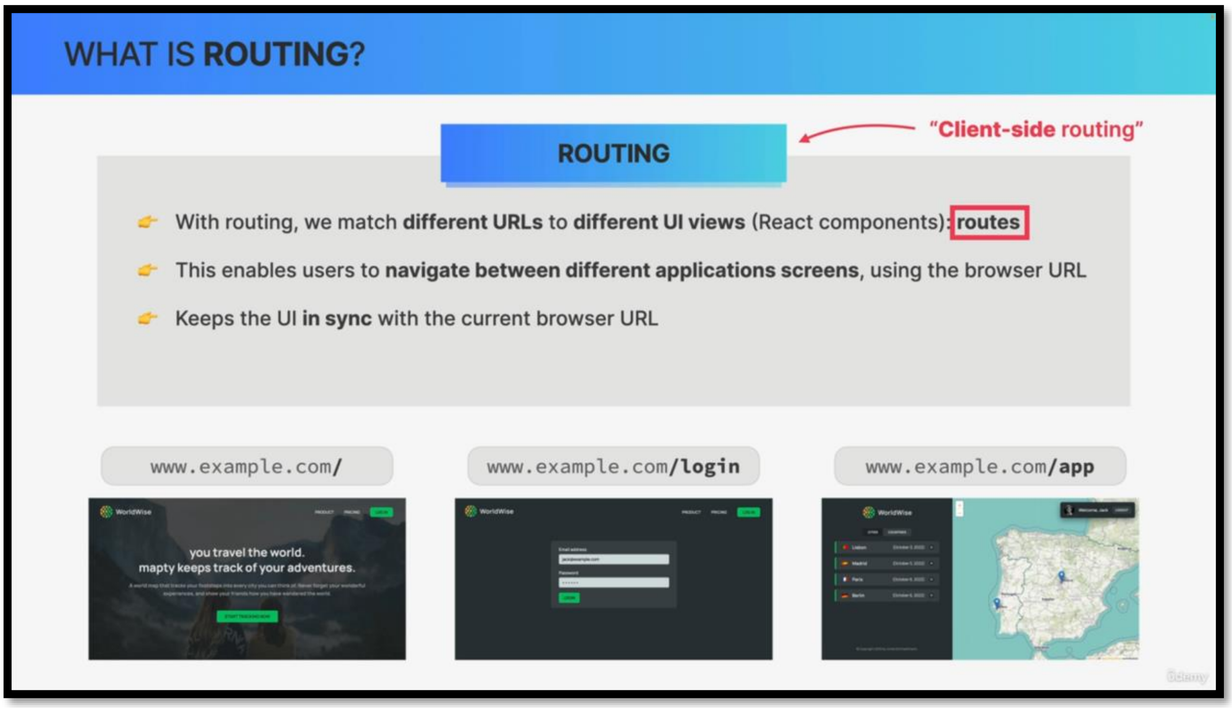


WHAT IS **ROUTING?**

ROUTING

- With routing, we match **different URLs** to **different UI views** (React components): **routes**
- This enables users to **navigate between different applications screens**, using the browser URL
- Keeps the UI **in sync** with the current browser URL

Routing like this keeps the user interface nicely in sync with the current browser URL.



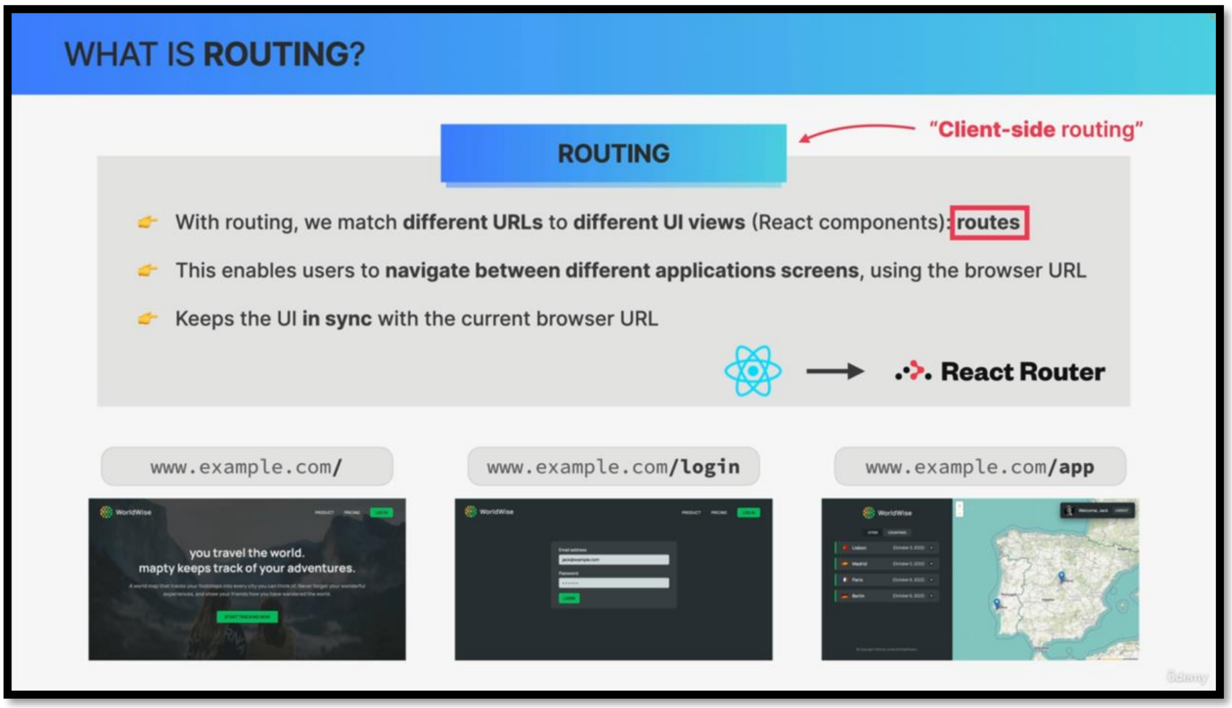
WHAT IS ROUTING?

ROUTING ← "Client-side routing"

- With routing, we match **different URLs** to **different UI views** (React components): **routes**
- This enables users to **navigate between different applications screens**, using the browser URL
- Keeps the UI **in sync** with the current browser URL

www.example.com/ — www.example.com/**login** — www.example.com/**app**

This type of routing only works this way on the client side i.e. in the user's browser.



WHAT IS ROUTING?

ROUTING ← "Client-side routing"

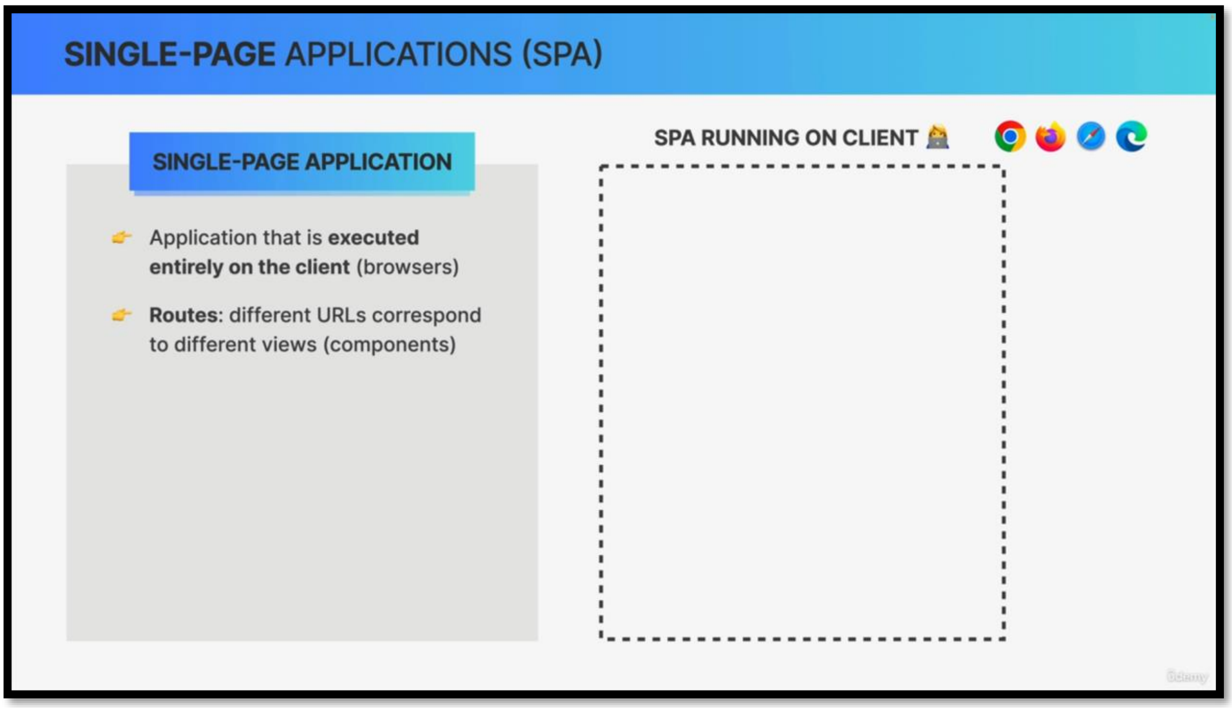
- With routing, we match **different URLs** to **different UI views** (React components): **routes**
- This enables users to **navigate between different applications screens**, using the browser URL
- Keeps the UI **in sync** with the current browser URL

→ **React Router**

www.example.com/ — www.example.com/**login** — www.example.com/**app**

Most front-end frameworks have these client side routing capabilities baked right into the framework but React is different, because it relies on third party packages for many different functionalities and routing is one of them.

So, in React, routing is usually handled by this third party package called React Router.

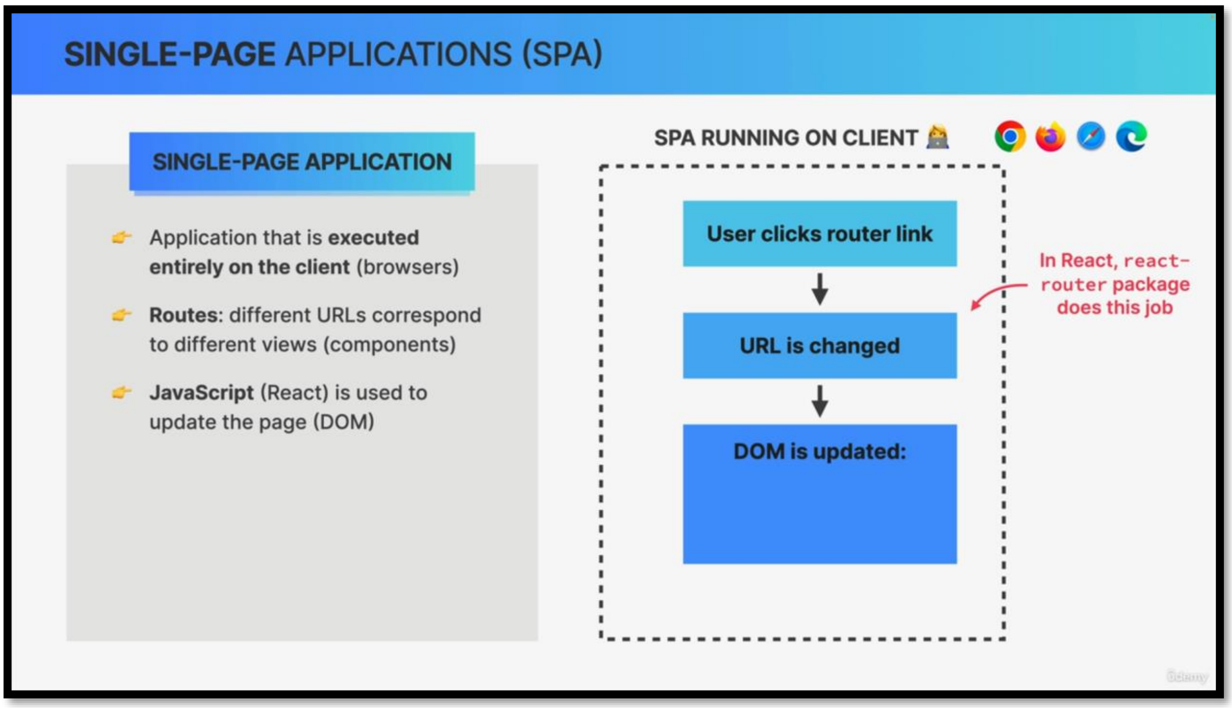


SINGLE-PAGE APPLICATIONS (SPA)

SINGLE-PAGE APPLICATION

- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)

Single-page applications rely heavily on the concept of routes where different URLs correspond to different views.



SINGLE-PAGE APPLICATIONS (SPA)

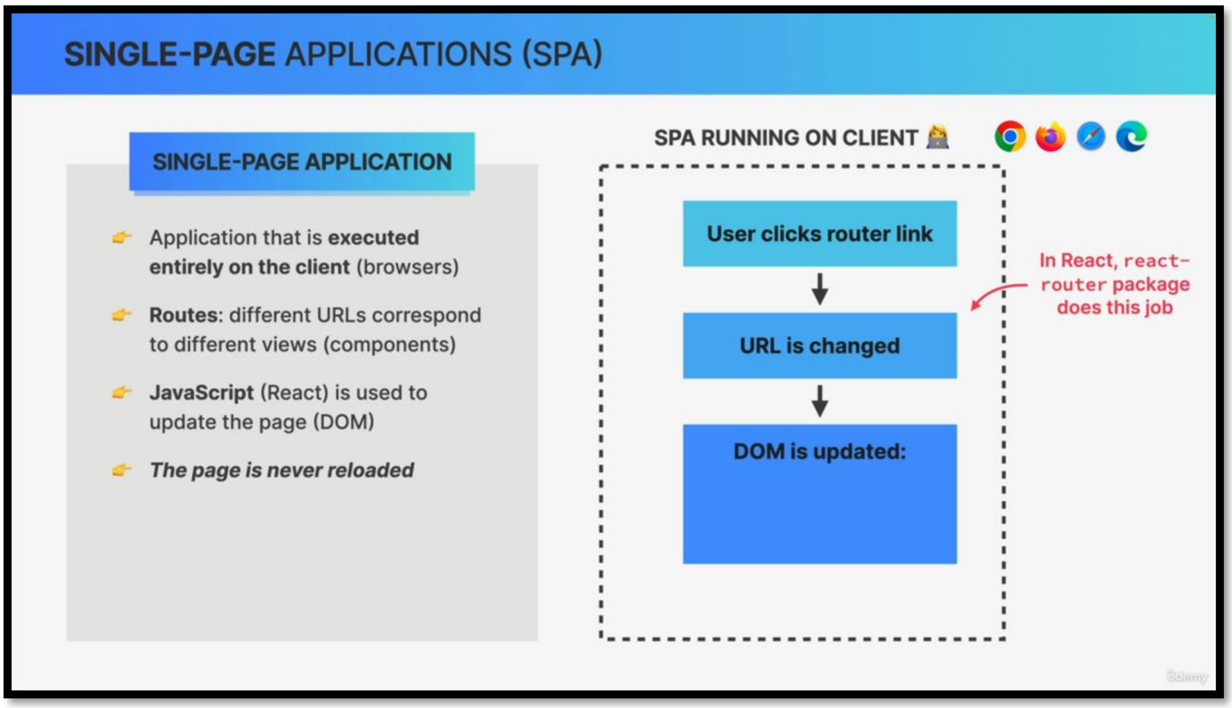
SINGLE-PAGE APPLICATION

- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)
- **JavaScript** (React) is used to update the page (DOM)

Here is how single-page applications work.

Whenever a user clicks on a special link provided by the Router, the URL in the browser simply changes. In the case of React, this job is usually done by React Router.

Now, changing the URL will then trigger the DOM to be updated as a result. In single-page applications, it's always JavaScript that will update the DOM and therefore the page.

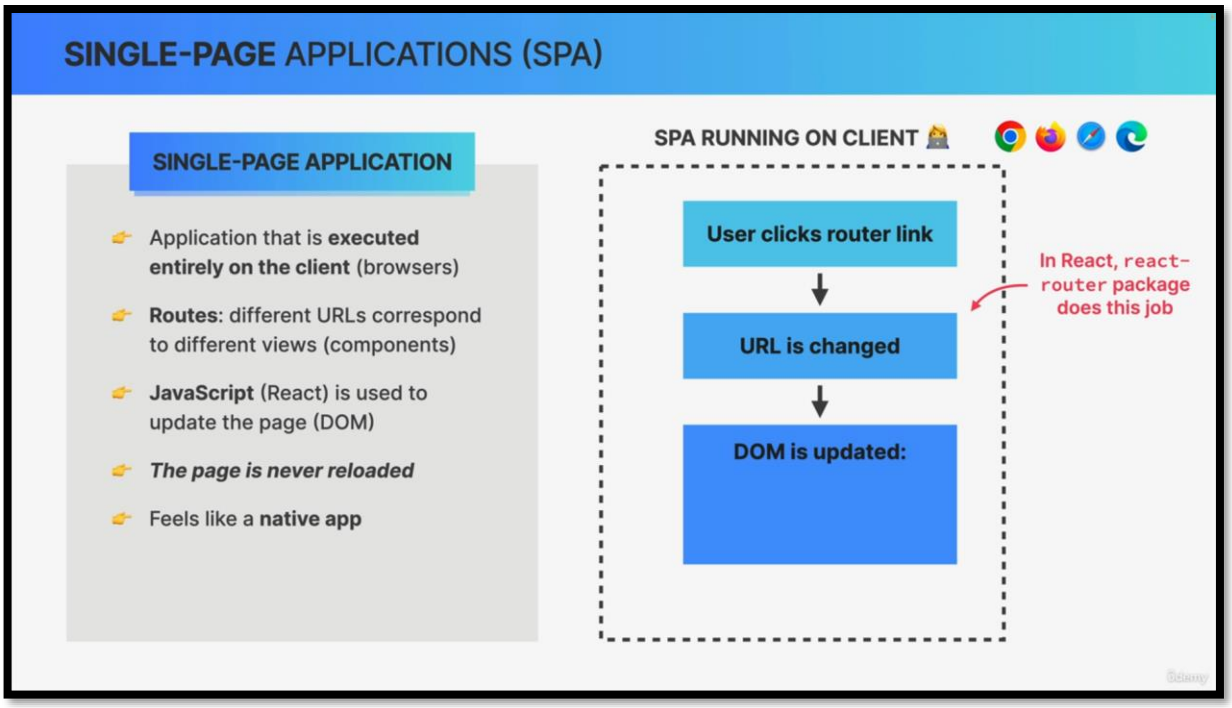


SINGLE-PAGE APPLICATIONS (SPA)

SINGLE-PAGE APPLICATION

- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)
- **JavaScript** (React) is used to update the page (DOM)
- ***The page is never reloaded***

So usually on a normal webpage, when we click on a link, the browser will load a completely new page and then show us that new page but single-page applications are completely different. The page is simply updated by JavaScript, which means that there will never be a complete page reload.

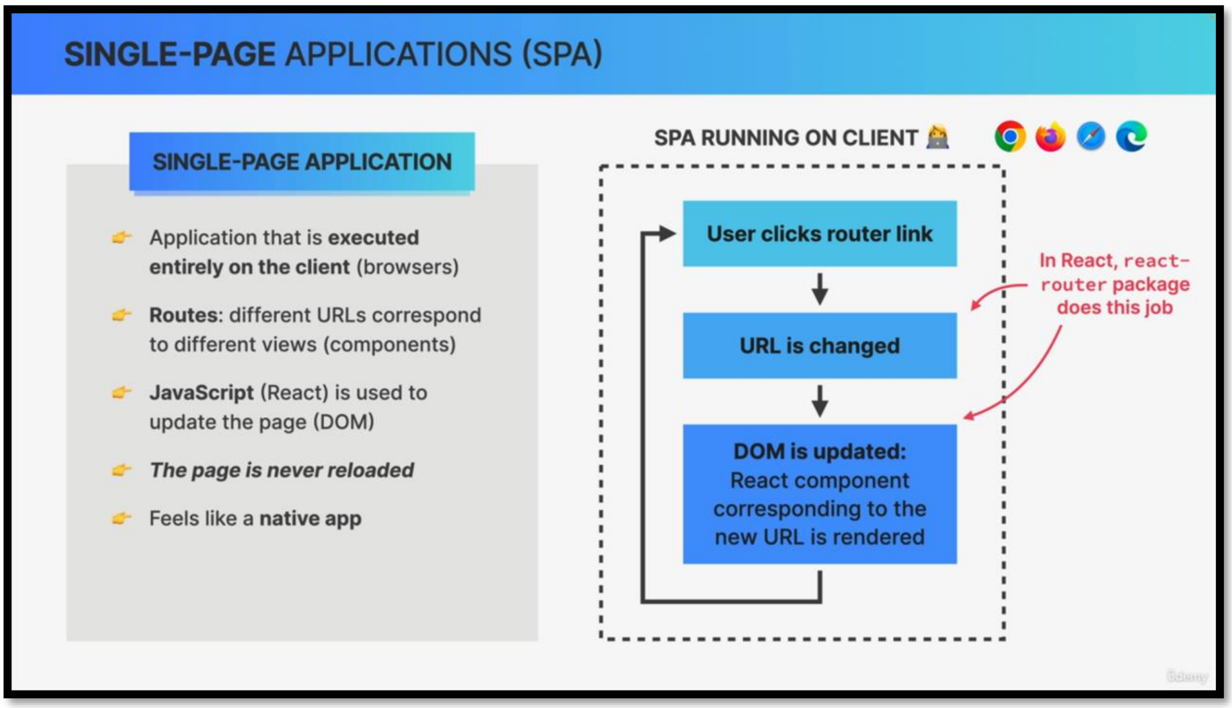


SINGLE-PAGE APPLICATIONS (SPA)

SINGLE-PAGE APPLICATION

- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)
- **JavaScript** (React) is used to update the page (DOM)
- ***The page is never reloaded***
- Feels like a **native app**

That's the whole point of the single-page application. It's the entire app in just one page. So, without any hard reloads. This makes the web application feel just like a native desktop or a mobile application, which is really a fantastic user experience.



SINGLE-PAGE APPLICATIONS (SPA)

SINGLE-PAGE APPLICATION

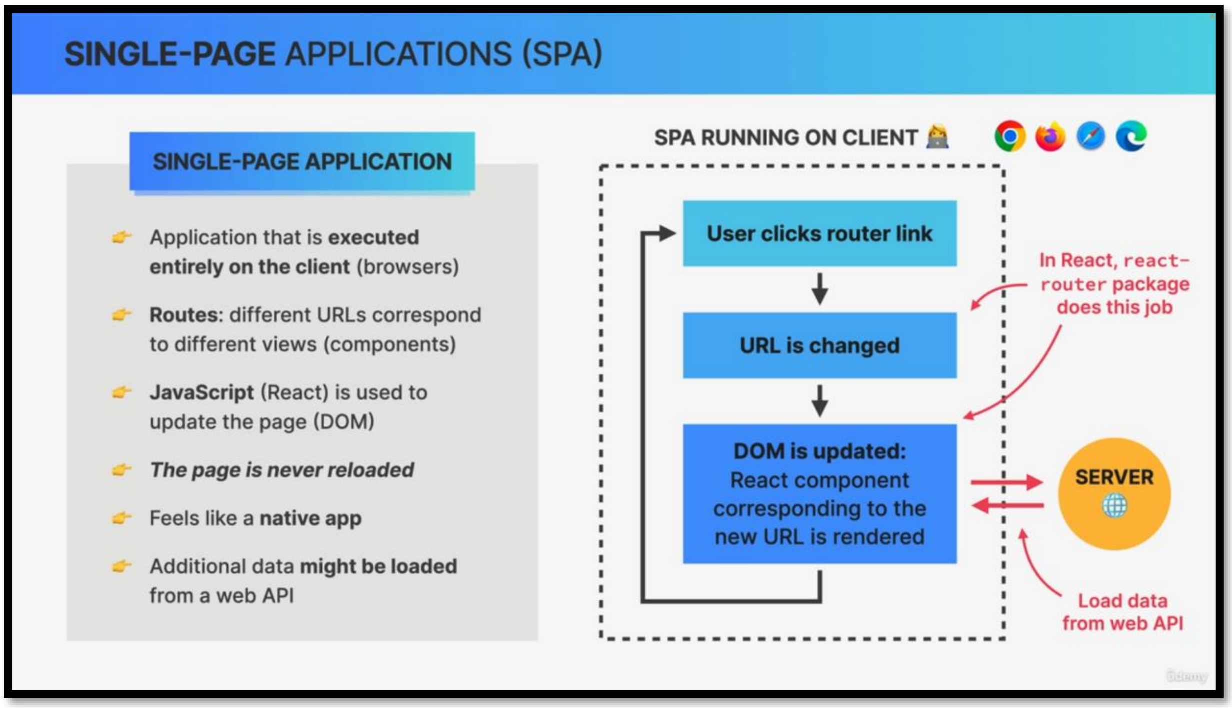
- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)
- **JavaScript** (React) is used to update the page (DOM)
- ***The page is never reloaded***
- Feels like a **native app**

SPA RUNNING ON CLIENT
User clicks router link
URL is changed
DOM is updated: React component corresponding to the new URL is rendered
In React, react-router package does this job

Whenever the URL is changed, React Router and React itself will update the DOM by simply rendering the component that corresponds to the new URL.

The whole cycle can be repeated as many times as necessary.

So, each time the user keeps clicking on a Router link, that will change the URL and the component that's being displayed on the screen, all without reloading the page.



SINGLE-PAGE APPLICATIONS (SPA)

SINGLE-PAGE APPLICATION

- Application that is **executed entirely on the client** (browsers)
- **Routes:** different URLs correspond to different views (components)
- **JavaScript** (React) is used to update the page (DOM)
- ***The page is never reloaded***
- Feels like a **native app**
- Additional data **might be loaded** from a web API

SPA RUNNING ON CLIENT
User clicks router link
URL is changed
DOM is updated: React component corresponding to the new URL is rendered
In React, react-router package does this job
SERVER
Load data from web API

It's quite common that some pages need to display some external data, but that's not a problem at all. Whenever that happens, a component can just load some additional data from a server, usually from some kind of web API.

So, while the single-page app itself runs entirely on the client, it can always communicate with a server in order to fetch some data that it needs, just like we have been doing before in other applications.

What we cannot do is to load a completely new page, because then it would no longer be a single-page app.

We could actually say that all React apps are in fact single-page applications, because all of them are never reloaded.