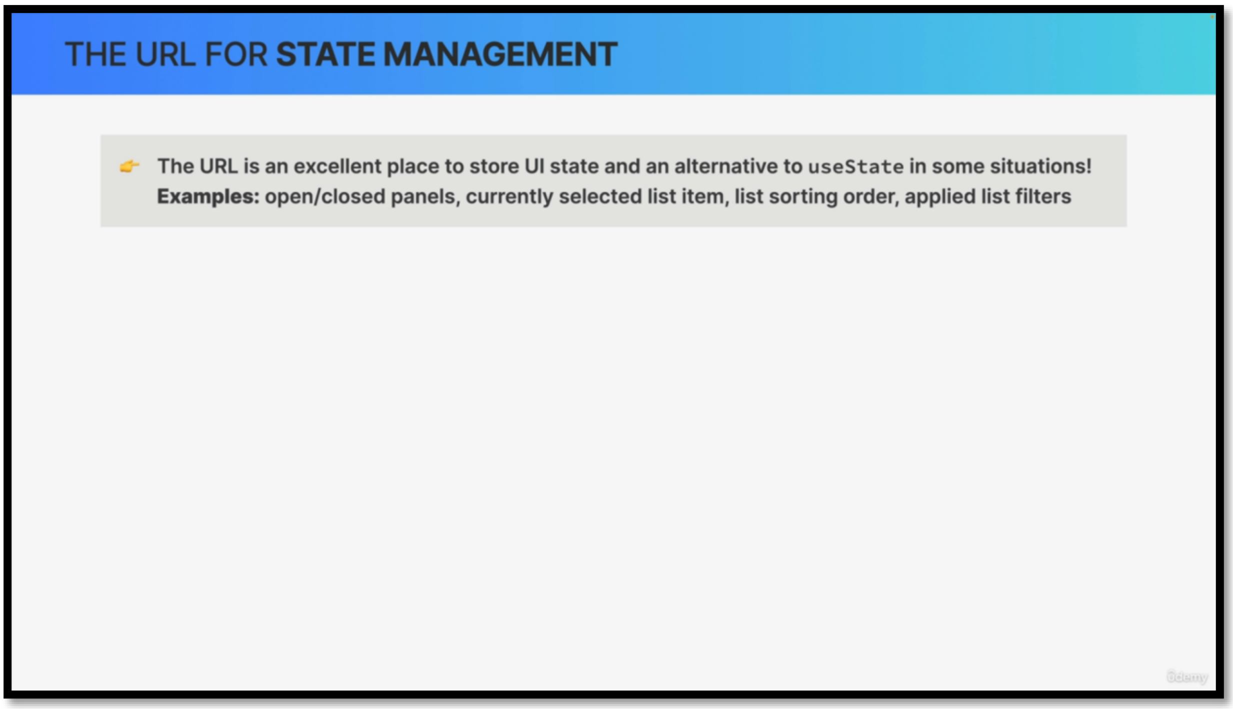


THE URL FOR **STATE MANAGEMENT**

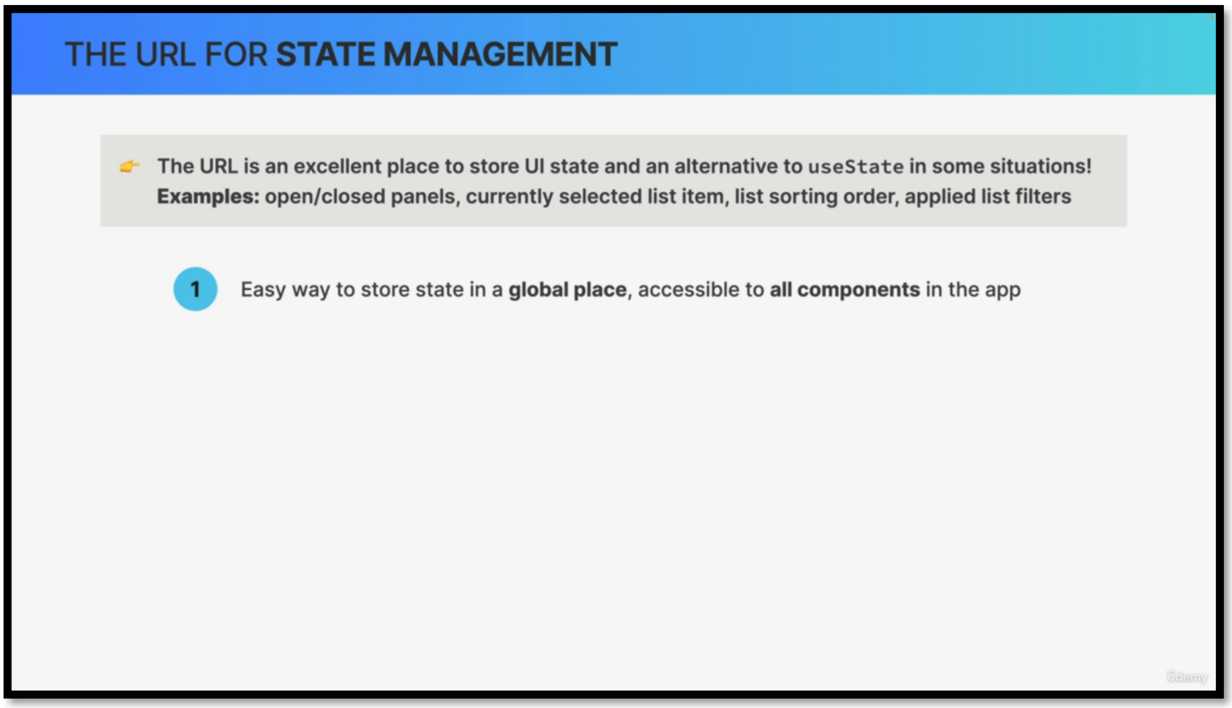
👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

"Storing state in the URL, don't we actually use the useState hook to manage state?"

Well, that's true most of the time. But the URL is actually also an excellent place to store a state and especially UI state.

With UI state, we mean state that affects what the UI looks like. So, things like an open or closed panel, or a currently applied list sorting order or filter. So, these examples of state are great candidates to be stored in the URL and basically to be managed by the URL with React Router.

"Now, why would we want to do that?"



THE URL FOR STATE MANAGEMENT

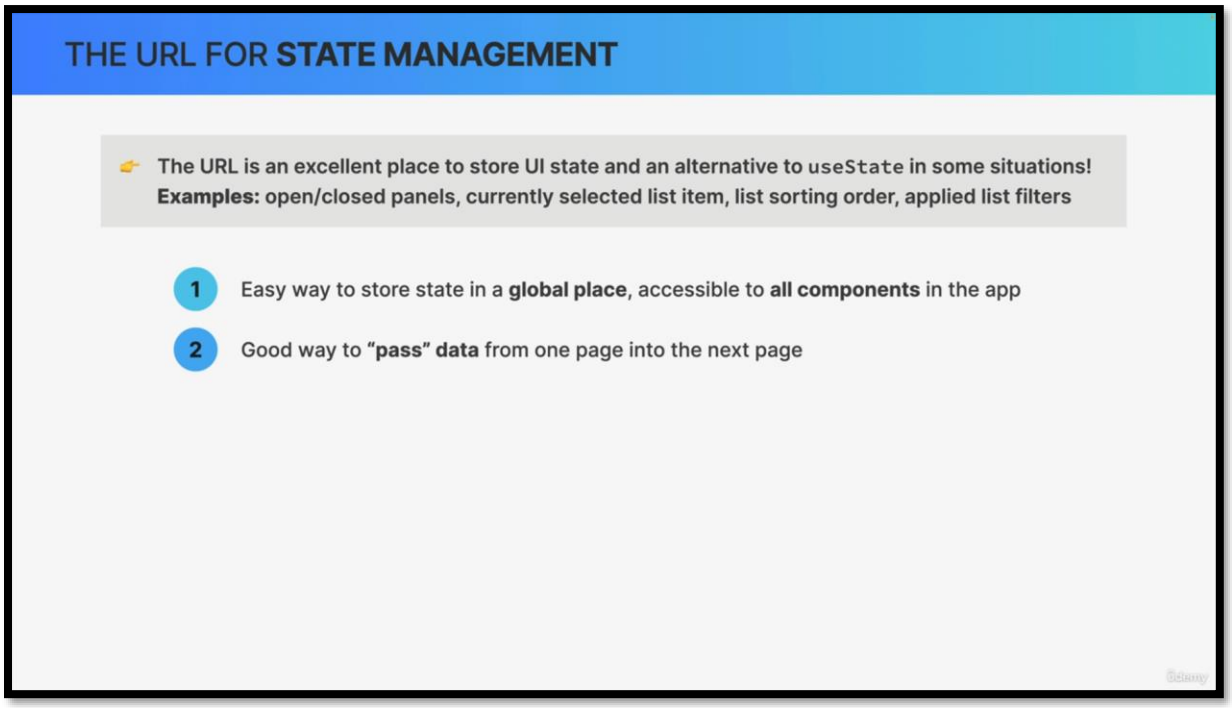
👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

1. Easy way to store state in a **global place**, accessible to **all components** in the app

The first reason is that placing state in the URL is an easy way to store state in a global place that is easily accessible to all components in the app.

So before, if we wanted state to be accessible everywhere, we would have to store it in a parent component and then pass it all the way down to all child components using props but if we place state in the URL, we can easily just read the value from there, wherever the component is in the component tree.

So, basically we can move some state management from React to the URL.

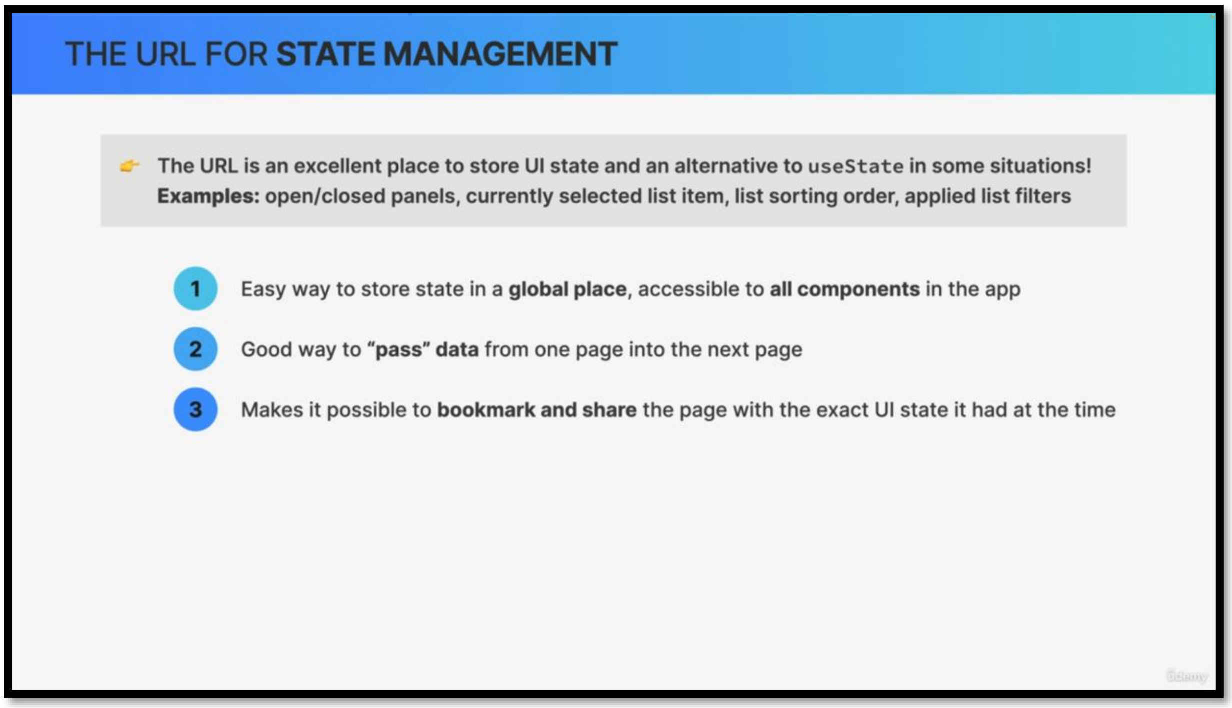


THE URL FOR **STATE MANAGEMENT**

👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

1. Easy way to store state in a **global place**, accessible to **all components** in the app
2. Good way to **"pass" data** from one page into the next page

Placing state in the URL is, in many situations, a good way to pass data from one page into the next page without having to store that data in some temporary place inside the app.



THE URL FOR **STATE MANAGEMENT**

👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

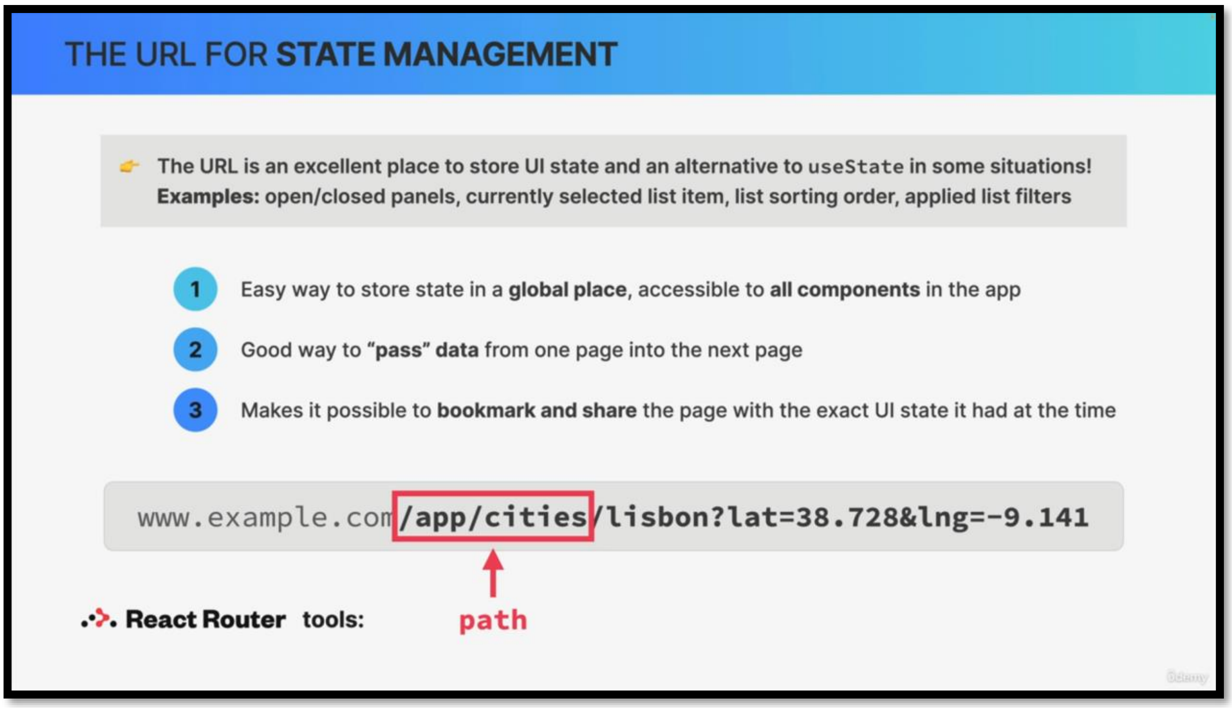
1. Easy way to store state in a **global place**, accessible to **all components** in the app
2. Good way to **"pass" data** from one page into the next page

3. Makes it possible to **bookmark and share** the page with the exact UI state it had at the time

Another amazing reason why we should place UI state right in the URL is that doing so makes it possible to bookmark or to share the page with the exact UI state that the page had at the time that we are sharing or bookmarking it.

For example, in an online shop, we might be filtering products by color and by price and if that filter is saved in the URL, we can then share the page with someone and they will see the exact same filters applied to the list of products.

So that's really helpful and enables a great user experience.



THE URL FOR **STATE MANAGEMENT**

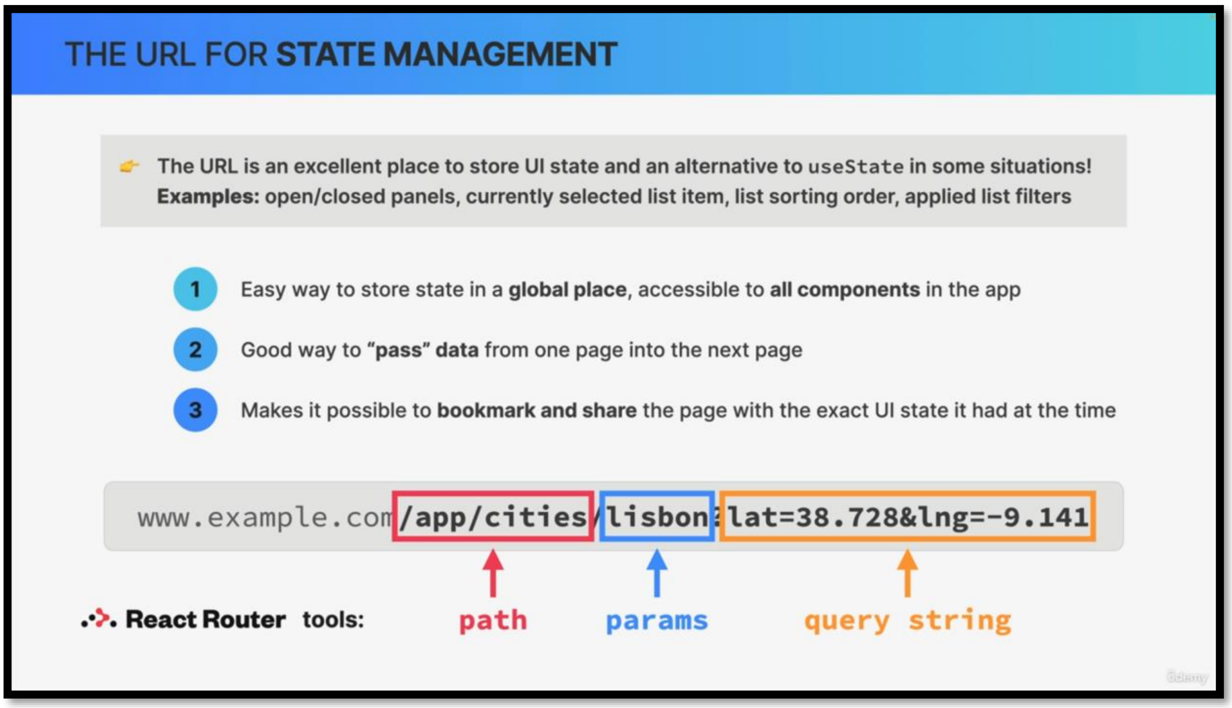
👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

1. Easy way to store state in a **global place**, accessible to **all components** in the app
2. Good way to **"pass" data** from one page into the next page
3. Makes it possible to **bookmark and share** the page with the exact UI state it had at the time

```
www.example.com/app/cities/lisbon?lat=38.728&lng=-9.141
```

In a URL like this, we already know that we have a path, for example, app/cities, and we can consider this part state because it corresponds to the component that is currently being displayed.

But this path is not really useful for state management in the way that they have been describing.



THE URL FOR STATE MANAGEMENT

👉 The URL is an excellent place to store UI state and an alternative to `useState` in some situations!
Examples: open/closed panels, currently selected list item, list sorting order, applied list filters

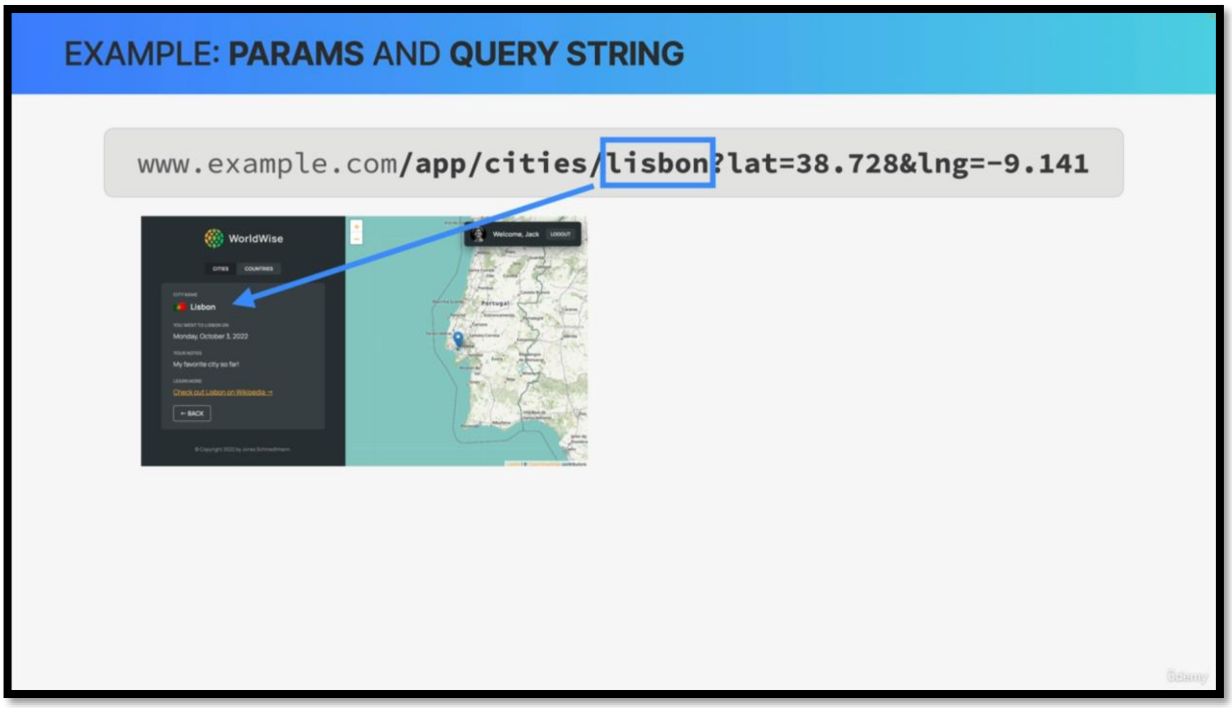
1. Easy way to store state in a **global place**, accessible to **all components** in the app
2. Good way to **"pass" data** from one page into the next page
3. Makes it possible to **bookmark and share** the page with the exact UI state it had at the time

```
www.example.com/app/cities/lisbon?lat=38.728&lng=-9.141
```

React Router tools: path params query string

So, for actually storing state in the URL, we use Params or the Query string.

Params stands for parameters are very useful to pass data to the next page while the Query string is useful to store some global state that should be accessible everywhere.

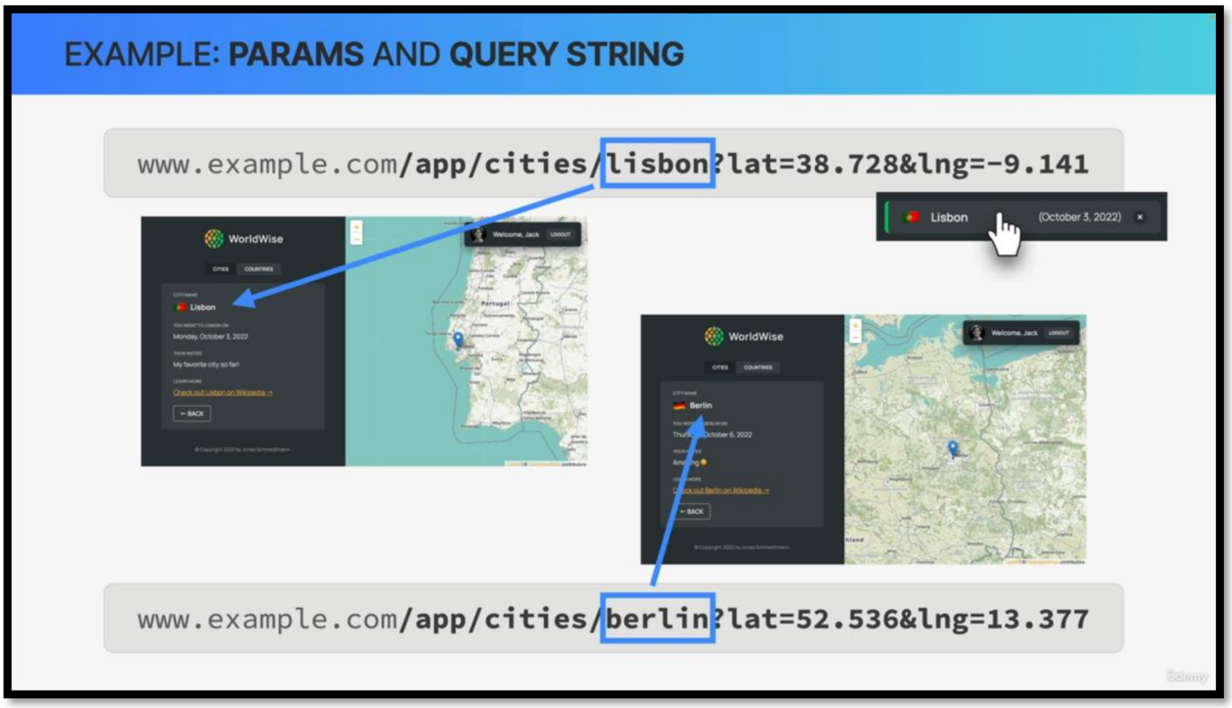


EXAMPLE: PARAMS AND QUERY STRING

```
www.example.com/app/cities/lisbon?lat=38.728&lng=-9.141
```

So, this URL that we just looked at corresponds to this view, and in the URL you see that the Param is Lisbon.

Because of that, the page that was loaded is exactly about the city of Lisbon. So, by creating a link that points to a URL with this Param, we are able to pass the city name to the next page whenever the user clicks on that link.

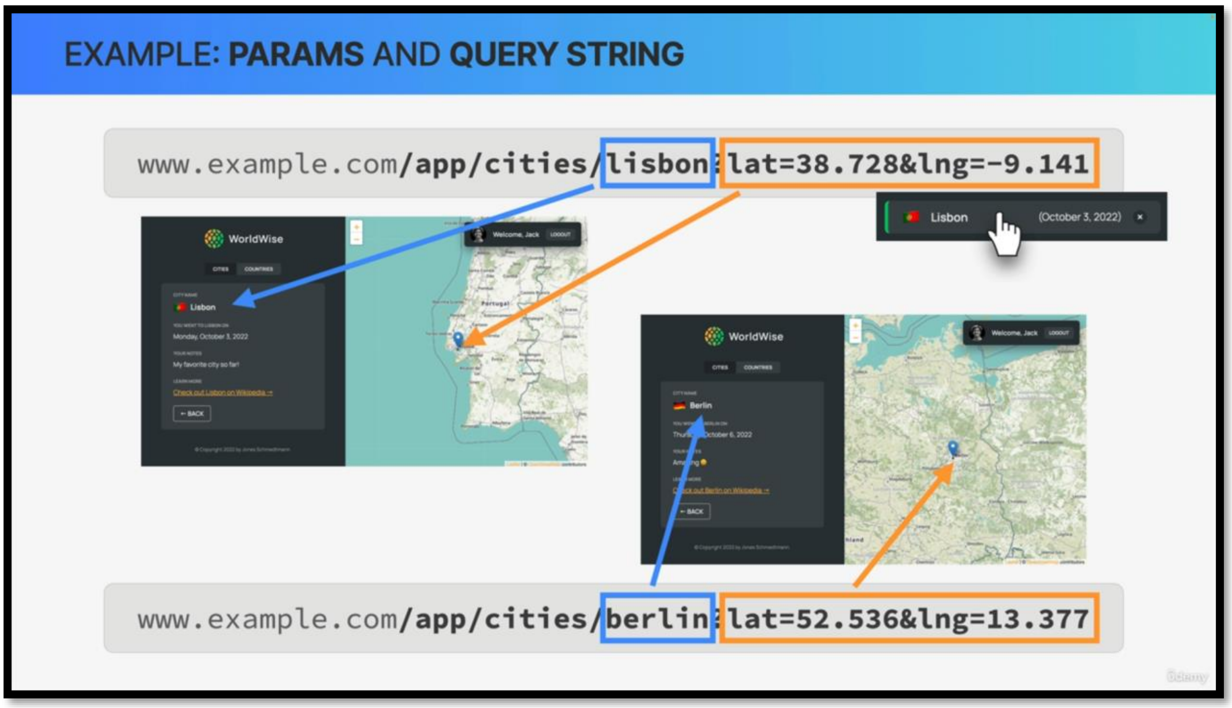


EXAMPLE: **PARAMS AND QUERY STRING**

```
www.example.com/app/cities/lisbon?lat=38.728&lng=-9.141
```

```
www.example.com/app/cities/berlin?lat=52.536&lng=13.377
```

If the URL had another city name as the Param, let's say Berlin then the loaded page would be about Berlin.



EXAMPLE: **PARAMS AND QUERY STRING**

```
www.example.com/app/cities/lisbon?lat=38.728&lng=-9.141
```

```
www.example.com/app/cities/berlin?lat=52.536&lng=13.377
```

So, in this example, we store the LAT and LNG pieces of state in a Query string which corresponds to disposition on the map and the same is true, of course, for the other URL.

So, each of these cities has a unique location, and that location is reflected right in the URL.

So in this example, we leverage the power of the URL to manage state in an effective way by reading the city name and the GPS location from the URL instead of using application state inside React.