Version: 6.x

ServerContainer

The `ServerContainer` component provides utilities to render your app on server with the correct navigation state.

Example:

```
// Ref which will be populated with the screen options
const ref = React.createRef();

// Location object containing the `pathname` and `search` fields of the current
URL
const location = { pathname: '/profile', search: '?user=jane' };

// Get rendered HTML
const html = ReactDOMServer.renderToString(
  <ServerContainer ref={ref} location={location}>
    <App />
  </ServerContainer>
);

// Then you can access the options for the current screen in the ref
const options = ref.current.getCurrentOptions(); // { title: 'My Profile' }
```

The `ServerContainer` component should wrap your entire app during server rendering. Note that you still need a `NavigationContainer` in your app, `ServerContainer` doesn't replace it.'

See the `server rendering guide` for a detailed guide and examples.

Ref

If you attach a `ref` to the container, you can get the options for the current screen after rendering the app. The `ref` will contain a method called `getCurrentOptions` which will return an object with options for the focused screen in the navigation tree:

```
const options = ref.current.getCurrentOptions();
```

Then you can access the options for the screen from this object and put it in the HTML:

```
<title>{options.title}</title>
<meta name="description" content={options.description} />
```

Note that the `options` object can be undefined if you are not rendering a navigator on the initial render.

Props

`location`

Location object containing the location to use for server rendered output. You can pass the `pathname` and `search` properties matching the `location` object in the browsers:

```
<ServerContainer location={{ pathname: '/profile', search: '' }}>
  <App />
</ServerContainer>
```

Normally, you'd construct this object based on the incoming request.

Basic example with Koa (don't use as is in production):

```
app.use(async (ctx) => {
  const html = ReactDOMServer.renderToString(
    <ServerContainer location={{ pathname: ctx.path, search: ctx.search }}>
      <App />
    </ServerContainer>
  );

  ctx.body = html;
});
```

Edit this page