EXERCÍCIOS

AULA 4

1. Implementar um componente, no arquivo “src/aula4/Pages/Refs/CustomTextInput.jsx”, que exiba um input de texto e um botão. Ao clicar no botão, mudar o foco do cursor para input text. Utilizar refs.

2. Implementar um componente, no arquivo “src/aula4/Pages/Refs/AutoFocusTextInput.jsx”, que utilize o componente criado no exercício anterior, o exibindo na tela, mas que receba o foco do cursor depois da construção do componente. Utilizar refs a função componentDidMount do ciclo de vida.

3. Implementar um componente Glossario, no arquivo “src/aula4/Pages/Acessibilidade/Glossario.jsx”. O componente deve receber, via props, um array “items”, contendo vários objetos com {id, nome, descricao}. O componente deve renderizar um <dl>, exibindo, para cada um dos itens, <dt>{item.nome}</dt> e <dd>{item.descricao}. O componente deve utilizar técnicas de acessibilidade.

4. Implementar um componente KeybordFocus, no arquivo “src/aula4/Pages/Acessibilidade/KeybordFocus.jsx”, que renderize um HTML link <a>, contendo, em seu href, o conteúdo “https://portal.dataprev.gov.br” e com o label “Acesse o portal da Dataprev”. O componente deve estar acessível aos leitores de tela.

5. Implementar um provedor de Contexto, com o nome ThemeContext, no arquivo “src/aula4/Pages/Contexto/ThemeContext.jsx, contendo uma instância de contexto React, que irá armazenar um tema selecionado, a partir de uma variável “themes”, que deve conter as definições de dois temas: “light” e “dark”. O provedor de tema deve utilizar React.createContext(), passando como tema inicial themes.dark.

6. Implementar um componente “App”, no arquivo “src/aula4/Pages/Contexto/index.jsx”, que deve renderizar uma árvore de componentes que façam uso do ThemeProvider criado no exercício anterior. Um desses componentes deve ser um botão, que, ao ser clicado, alterne entre o tema “dark” e “light”.

EXERCÍCIOS

7. Implementar um componente ErrorBoundary, no arquivo “src/aula4/Pages/ErrorBoundary/ErrorBoundary.jsx”. Esse componente será responsável por capturar os erros da aplicação e exibir na tela uma mensagem amigável. Utilizar o método componentDidCatch do ciclo de vida.

8. Implementar um componente “App”, no arquivo “src/aula4/Pages/ErrorBoundary/index.jsx”, que faça uso do ErrorBoundary, criado no exercício anterior. O componente deve renderizar um botão, que simule um erro para que seja capturado pelo ErrorBoundary.

9. Implementar um componente de alta ordem (HOC), chamado withLoading, no arquivo “src/aula4/Pages/HOCs/withLoading.js”, que renderize uma instância com componente CircularProgress, do Material-UI, enquanto “props.isLoading” for “true”. Ao mudar para “false”, renderizar o componente que estiver sido envolvido pela HOC withLoading.

10. Implementar um componente “App”, em “src/aula4/Pages/HOCs/index.jsx”, que renderize dois componentes que usem a HOC criada no exercício anterior: uma tabela listando usuários e uma exibição de logs do sistema, respectivamente. Simular o carregamento dos dados de uma API, para os dois componentes, criando um “delay” (utilizar setTimeout() do JavaScript) de 3 e 5 segundos, para cada componente, respectivamente.

11. Implementar um componente Modal, no arquivo “src/aula4/Pages/Portals/Modal.jsx”. Esse componente deve utilizar “Portals”, e renderizar o modal na div “modal-root” do “document”.