
ICARO
Nuestro compromiso es con el futuro.

React

Clase 7

React: Producción 1

Temas:

- Conceptos
- Estado Global
- Navegación

React

Producción 1: Conceptos

Producción 1: Conceptos

Conceptos en producción:

- SPA
- Escalabilidad
- Lógica UI
- Lógica Negocio
- Performance
- Accesibilidad

Producción 1: Conceptos

Todos estos conceptos son universales en cualquier proyecto de FrontEnd.

React no tiene opiniones sobre como trabajar cada uno de estos conceptos.

Producción 1: Conceptos

El conocimiento de las soluciones que aportan las librerías complementarias a React más utilizadas nos ahorrará tiempo de desarrollo y testeo.

Producción 1: Conceptos

SPA:

- *Sigla en inglés para “Single Page Application”, son páginas que tienen la lógica en el navegador.*
- *Cuando creamos un proyecto iniciado con Vite estamos desarrollando este tipo de páginas por defecto.*
- *Tienen problemas de SEO, se comportan como APPs y se trata de evitar recargas en el navegador.*

Producción 1: Conceptos

Escalabilidad:

- *El concepto es universal a cualquier proyecto, trata sobre el crecimiento del mismo.*
- *Si el proyecto es grande, es conveniente aplicar la mayor cantidad de prácticas que mejoren este aspecto.*
- *En React la componentización es clave para tratar este punto.*

Producción 1: Conceptos

Lógica UI:

- *Toda UI se comporta como una APP, la lógica está orientada a tratar con el usuario de nuestra APP.*
- *La experiencia de usuario puede mejorar mucho si trabajamos correctamente la lógica de nuestra UI.*
- *También podemos validar preventivamente datos ingresados por el usuario para prevenir errores.*

Producción 1: Conceptos

Lógica Negocio:

- *Es la lógica relacionada con las reglas y objetivos de un proyecto en particular.*
- *Este punto pertenece mayoritariamente al BackEnd, sin embargo puede trabajarse desde el front.*
- *La buena comunicación con el backend es fundamental para evitar tener este tipo de lógica en el front.*

Producción 1: Conceptos

Performance:

- *Hoy en día los procesadores son lo suficientemente veloces como para no estar obsesionados con esto.*
- *Sin embargo existen mejoras que permiten optimizar nuestra APP.*
- *En React los hooks “avanzados” son de utilidad a la hora de realizar este tipo de mejoras.*

Producción 1: Conceptos

Accesibilidad:

- *Se refiere a las capacidades de nuestros usuarios para interactuar con la UI.*
- *React no tiene manejo propio de la accesibilidad, tiene la misma accesibilidad que el HTML común.*
- *Existen librerías que están fuertemente probadas y cumplen con estándares altos de accesibilidad.*

React

Producción 1: Estado Global

Producción 1: Estado Global

¿Qué veremos?:

- Prop Drilling
- Problemas que resuelve
- Problemas que trae
- Zustand
- Aplicaciones

Producción 1: Estado Global

Prop Drilling:

- *En React la forma de comunicación entre componentes es a través de las “props”.*
- *La comunicación por medio de props en componentes anidados se conoce como “prop drilling”.*
- *Este tipo de comunicación se hace de padre a hijo, y es siempre de forma explícita e imperativa.*

Producción 1: Estado Global

Prop Drilling:

```
1.  const Nieto = ({ tema = "" }) => {
2.    return <button>{tema === "oscuro" ? "🌙" : "🌞"}</button>
3.  }
4.
5.  const Hijo = ({ tema = "" }) => <Nieto tema={tema} />
6.
7.  const Padre = ({ tema = "" }) => <Hijo tema={tema} />
8.
9.  function App() {
10.   return <Padre tema="oscuro" />
11. }
```

Producción 1: Estado Global

Prop Drilling:

```jsx
const Nieto = ({ tema = "" }) => {
  return <button>{tema === "oscuro" ? "🌙" : "🌞"}</button> // El nieto consume el valor.
}

const Hijo = ({ tema = "" }) => <Nieto tema={tema} /> // El hijo pasa el valor del padre al nieto.

const Padre = ({ tema = "" }) => <Hijo tema={tema} /> // El Padre recibe el valor del componente App.

function App() {
  return <Padre tema="oscuro" /> // En este lugar se determina el valor que va a llegar hasta el último nivel.
}
```

Producción 1: Estado Global

Problemas que resuelve:

- *El más notorio de los problemas que resuelve el estado global, es el del Prop Drilling.*
- *Esta práctica nos permite almacenar datos del usuario luego de un login.*
- *También nos permite guardar datos que el usuario ingresa en nuestra App y usarlo en toda la App.*

Producción 1: Estado Global

Problemas que trae:

- *Usando esta práctica heredamos todos los problemas que la misma tiene en la programación en general.*
- *Muchos estados globales dificultan el mantenimiento y la escalabilidad del proyecto.*
- *Si no usamos “useContext” requeriremos instalar librerías, cada una tiene diferentes configuraciones.*

Producción 1: Estado Global

Zustand:

- *La librería que utlizaremos como ejemplo en este curso será Zustand.*
- *Requiere mínima configuración y es más sencilla de utilizar que otras alternativas.*
- *Algunas alternativas son “useContext”, o librerías como “Redux” o “MobX”.*

Producción 1: Estado Global

Zustand:

npm:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  npm install zustand
```

yarn:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  yarn add zustand
```

documentación: https://github.com/pmndrs/zustand

Producción 1: Estado Global

./stores/useUserStore.js:

```
1.  import { create } from "zustand“
2.
3.  const useUserStore = create(set ⇒ ({
4.    tema: "oscuro",
5.    toggleTema: () ⇒
6.      set(state ⇒ ({ tema: state.tema === "oscuro" ? "claro" : "oscuro" })),
7.  }))
8.
9.  export { useUserStore }
```

Producción 1: Estado Global

./stores/useUserStore.js:

```
1.  import { create } from "zustand" // Importamos el método `create` de la librería zustand.
2.
3.  const useUserStore = create(set => ({ // Creamos el modelo de nuestro store.
4.    tema: "oscuro", // Configuramos el valor inicial del campo `tema`.
5.    toggleTema: () => // Configuramos la lógica del setter para el campo `tema`.
6.      set(state => ({ tema: state.tema === "oscuro" ? "claro" : "oscuro" })),
7.  }))
8.
9.  export { useUserStore }
```

Producción 1: Estado Global

Prop Drilling:

```
1.  import { useUserStore } from "./stores/useUserStore"
2.
3.  const Nieto = () => {
4.    const tema = useUserStore(state => state.tema)
5.    const toggleTema = useUserStore(state => state.toggleTema)
6.    return <button onClick={toggleTema}>{tema === "oscuro" ? "🌙" : "🌞"}</button>
7.  }
8.
9.  const Hijo = () => <Nieto />
10.
11. const Padre = () => <Hijo />
12.
13. function App() {
14.   const tema = useUserStore(state => state.tema)
15.   console.log("estoy dentro del componente App:", tema)
16.   return <Padre />
17. }
```

Producción 1: Estado Global

Prop Drilling:

```
1.  import { useUserStore } from "./stores/useUserStore" // Importamos nuestro store.
2.
3.  const Nieto = () => {
4.    const tema = useUserStore(state => state.tema) // Declaramos la variable que contiene el campo `tema`.
5.    const toggleTema = useUserStore(state => state.toggleTema) // Declaramos el setter del estado global.
6.    return <button onClick={toggleTema}>{tema === "oscuro" ? "🌙" : "🌞"}</button> // Llamamos al setter en cada click.
7.  }
8.
9.  const Hijo = () => <Nieto />
10.
11. const Padre = () => <Hijo />
12.
13. function App() {
14.   const tema = useUserStore(state => state.tema)
15.   console.log("estoy dentro del componente App:", tema) // A modo de ejemplo observamos que el dato es global.
16.   return <Padre />
17. }
```

React

Producción 1: Navegación

Producción 1: Navegación

¿Qué conceptos veremos?

- Renderizado Condicional
- Evitar recarga del navegador
- React Router
- URL / Hooks

Producción 1: Navegación

Renderizado Condicional:

- *Es ideal para proyectos pequeños ya que no requiere configuración extra ni librerías.*
- *Si usamos esta técnica, toda implementación tiene que se trabajada por nosotros explícitamente.*
- *Las librerías como React Router, Reach Router o TanStack Router, implementan esta técnica por debajo.*

Producción 1: Navegación

Renderizado condicional:

```
1. import { create } from "zustand“
2.
3. const usePagesStore = create(set ⇒ ({
4.   page: "home",
5.   navigateTo: pageString ⇒ set(() ⇒ ({ page: pageString })),
6. }))
7.
8. export { usePagesStore }
```

Producción 1: Navegación

Renderizado condicional:

```
1. import { create } from "zustand“ // Creamos un nuevo store para las navegación.
2.
3. const usePagesStore = create(set ⇒ ({
4.   page: "home", // Declaramos el campo `page`.
5.   navigateTo: pageString ⇒ set(() ⇒ ({ page: pageString })), // Seteamos page con el valor enviado en el componente.
6. }))
7.
8. export { usePagesStore }
```

Producción 1: Navegación

Renderizado condicional:

```jsx
import { usePagesStore } from "./stores/usePagesStore"
const HomePage = () => {
  const navigateTo = usePagesStore(state => state.navigateTo)
  return (
    <><h1>Home Page</h1>
      <button onClick={() => navigateTo("error")}>Ir al error.</button></>
  )
}
const ErrorPage = () => {
  const navigateTo = usePagesStore(state => state.navigateTo)
  return (
    <><h1>Error Page</h1>
      <button onClick={() => navigateTo("home")}>Ir Home.</button></>
  )
}
function App() {
  const page = usePagesStore(state => state.page)
  return page === "home" ? <HomePage /> : page === "error" ? <ErrorPage /> : <></>
}
```

Producción 1: Navegación

Renderizado condicional:

```jsx
import { usePagesStore } from "./stores/usePagesStore" // Importamos nuestro store de navegación.
const HomePage = () => {
  const navigateTo = usePagesStore(state => state.navigateTo) // Declaramos el método para navegar entre nuestras páginas.
  return (
    <><h1>Home Page</h1>
      <button onClick={() => navigateTo("error")}>Ir al error.</button></> // Enviamos el valor de la página a la que queremos ir.
  )
}
const ErrorPage = () => {
  const navigateTo = usePagesStore(state => state.navigateTo)
  return (
    <><h1>Error Page</h1>
      <button onClick={() => navigateTo("home")}>Ir Home.</button></>
  )
}
function App() {
  const page = usePagesStore(state => state.page) // Consumimos el estado global del valor de la página.
  return page === "home" ? <HomePage /> : page === "error" ? <ErrorPage /> : <></> // Creamos lógica de navegación.
}
```

Producción 1: Navegación

Recarga del navegador:

- *El potencial de React se explota al máximo si usamos nuestra página como una App.*
- *Si Recargamos la página desde el navegador, todos nuestros estados globales de React se “resetean”.*
- *Por ese motivo es importante evitar usar la etiqueta <a>, y usar métodos como .preventDefault(),*

Producción 1: Navegación

React Router:

- *Es la librería más utilizada para la navegación, tiene buena documentación. Usaremos la versión 6.*
- *Tiene implementaciones como, actualización de URL, control de historial, params, etc.*
- *Es indicada para la gran mayoría de proyectos que requieran navegación, sean pequeños o grandes.*

Producción 1: Estado Global

React Router:

npm:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  npm install react-router-dom
```

yarn:

```
alumno@DESKTOP-ABCDFGH9 IJKLM0 ~/Documents/mi-proyecto
$  yarn add react-router-dom
```

documentación: https://reactrouter.com

Producción 1: Navegación

Renderizado condicional:

```jsx
import { BrowserRouter, Routes, Route, Link } from "react-router-dom"
const HomePage = () => {
  return (<><h1>Home Page</h1>
      <Link to={"/error"}>Ir al error.</Link></>)
}
const ErrorPage = () => {
  return (<><h1>Error Page</h1>
      <Link to={"/"}>Ir al error.</Link></>)
}
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/error" element={<ErrorPage />} />
      </Routes>
    </BrowserRouter>
  )
}
```

Producción 1: Navegación

Renderizado condicional:

```
1.  import { BrowserRouter, Routes, Route, Link } from "react-router-dom" // Importamos los componentes de la librería.
2.  const HomePage = () => {
3.    return (<><h1>Home Page</h1>
4.        <Link to={"/error"}>Ir al error.</Link></>) // Usamos el componente <Link /> para navegar a la página de Error.
5.  }
6.  const ErrorPage = () => {
7.    return (<><h1>Error Page</h1>
8.        <Link to={"/"}>Ir al error.</Link></>) // Usamos el componente <Link /> para navegar a la página de Home.
9.  }
10. function App() {
11.   return (
12.     <BrowserRouter> // Este componente habilita la navegación de manera global, forma parte de la configuración de la librería.
13.       <Routes> // Este componente se encarga de manejar las rutas.
14.         <Route path="/" element={<HomePage />} /> // Este componente asocia cada ruta con cada página.
15.         <Route path="/error" element={<ErrorPage />} />
16.       </Routes>
17.     </BrowserRouter>
18.   )
19. }
```

Producción 1

Traten de Recrear cualquier página que hayan creado hasta ahora.

Estas herramientas trabajan sobre los fundamentos de React.

Es importante leer la documentación oficial de cada herramienta.

¡Vamos al código!

Clase 7: Producción 1

¡Muchas gracias!

ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar