UD4 – React – *básicos…*

DAW2 - DEWC

Historia

Desarrollado por Jordan Walke en 2011 para Meta.

En 2013 se hace código abierto.

En 2015 se lanza React Native.

Características

Declarativo

Basado en componentes

Agnóstico de la plataforma

Pensado para la seguridad

Librerías

React => Define la estructura de la aplicación y la lógica de los componentes de manera independiente a la plataforma.

Renderizadores

- Para web -> “react-dom”. Mas concretamente:
  - “react-dom/client”
  - “react-dom/server”
- Para móviles -> “react-native”.
- Para Windows -> “react-native-windows”.
- Para macOS -> “react-native-macos”.

Conceptos

Componente -> Unidad básica. Tiene propiedades, estados y ciclo de vida. Código JSX.

Estado de un componente -> Datos internos que afectan al renderizado del componente.

Estados globales -> Estados compartidos entre componentes.

Hooks -> Herramientas para gestionar el estado y los efectos en los componentes. Por ej: useState, useEffect, useContext…

Enrutado -> Permite gestionar la navegación. Es un complemento “react-router-dom”.

Uso de CSS -> Maquetamos con clases CSS (no etiquetas, no estilos en línea)

Crear y arrancar un proyecto

Crear la estructura de un proyecto => **npm create vite@latest <nombre-proyecto>**

- Elige “React” como framework.
- Elige “JavaScript” como variante.

La salida del comando anterior te indica los siguientes comandos:

1. Entrar en el directorio => **cd nombre-proyecto**
2. Instalar los módulos de nodejs => **npm install**
3. Arrancar el proyecto => **npm run dev**

Herramientas

Piensa que podemos editar el código y ver en tiempo real las modificaciones.

Te va a solicitar instalar "***React DevTools***" => añade al depurador las pestañas "**components**" y "**profiler**".

ESLint => Linter de Microsoft, posiblemente ya lo tengas.

Prettier – Code formatter => Formateador de estilos recomendado por React.

Estructura del proyecto

Fichero “eslint.config.js”, añade las siguientes reglas:

```
rules: {
  ...
  'no-unused-vars': 'off',
  'react/prop-types': 'off',
},
```

Fichero “package.json”

- Apartado “scripts” => comandos de “npm”
  - npm run dev
  - npm run build
- Apartado “dependencies” => módulos para producción
- Apartado “devDependencies” => módulos adicionales para desarrollo

Cosas que hay que saber - I

Sólo hay un punto de entrada en “index.html”

- Necesita un contenedor (root)
- Carga el script “main.jsx”

“main.jsx” define el nodo raíz de la aplicación.

- Único renderizador de la aplicación -> “react-dom/client”
- StrictMode -> componente desarrollo, formatea los errores.
  - CUIDADO: renderiza dos veces cada componente.
- Hoja de estilos global “index.css”

Cosas que hay que saber - II

El componente “App.jsx” es el hola mundo, es muy completo.

- Define un estado.
- Introduce código JS dentro del código JSX con {}
- Define manejadores de eventos con “***onClick***”
- Maqueta con “***className***”
- Envuelve el código con un componente “***Fragment***”, diminutivo **<></>**

Estructura de un componente

Zona de “imports”

Definición del componente mediante “function”

- Zona de “Hooks”, en este caso “useState” pero pueden ser más.
- Código JavaScript auxiliar. Manejadores de eventos, validaciones, negocio…
- “return” código JSX.

Zona de “exports”

IMPORTANTE: un componente únicamente puede devolver un nodo, solución, emplear “Fragment”

Cosas de JSX…

Los bloques de código JSX van envueltos entre paréntesis **()**. Hay que ponerlos en los return y en los condicionales.

Te he engañado un poco… si sólo tenemos una línea no hace falta ().

No puedes usar if-else => usar ternario **?**, o expresiones "**truthy**" **&&**

En JS no hay propiedad "class" => usamos "**className**".

La etiqueta HTML label cambia "for" por "**htmlFor**" en JSX.

Comentarios => {/* Un comentario */}

CUIDADIN: en JSX el valor 0 es "truthy".

Estados - useState

Son variables cuya modificación renderiza el componente.

Se parecen, sin serlo, a los “getter/setter”.

```
const [mostrar, setMostrar] = useState(true);
```

Se crean e inicializan con “useState”.

El estado se comporta como una constante.

El estado se actualiza con el método “set”

=> El método “set” espera un parámetro con el nuevo valor del estado.

Intercambio propiedades

Los componentes tienes dos parámetros:

“props” => JSON con las propiedades que puede recibir el componente.

“children” => Nodos hijos (palabra reservada).

IMPORTANTE: ¡¡¡SON INMUTABLES!!!

¿Cómo puede comunicarse un nodo hijo con el padre? => callBacks….

```
function Hijo({ mensaje, comunicarPadre }) {
  const handleClick = () => {
    comunicarPadre("¡Mejor mañana!");
  };
```

Conclusiones

Leer la documentación.

Configurar el entorno.

Revisar ficheros del proyecto.

Hacer pruebas.

Preguntas