Proyecto BID-MINEC

PROYECTO UPSKILLING & RESKILLING

Orientado a la reconversión y mejora de habilidades digitales y tecnológicas

FORMÁS
CENTRO DE FORMACIÓN AVANZADO PARA PROFESIONALES

FEDISAL
Fundación para la Educación Integral Salvadoreña

BID

TECH SKILLS COUNCIL

GOBIERNO DE EL SALVADOR

MINISTERIO DE ECONOMÍA

FrontEnd básico-intermedio con JavaScript

React

Facilitador: Iván Alvarado

Contenido

1. ~~Recursos básicos (NodeJS)~~
2. React
    1. Estructura de directorios
    2. Prototypes
    3. Defaultprops
    4. Hooks
    5. useContext
    6. Redux
    7. Autenticación y autorización

2. React

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // Este enlace es necesario para hacer que `this` funcione en el callback
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

ReactDOM.render(
  <Toggle />,
  document.getElementById('root')
);
```

2. React

¿Qué es React Router?

- **React Router** es una colección de componentes de navegación la cual podemos usar como ya lo mencione tanto en web o en móvil con React Native. Con esta librearía vamos a obtener un enrutamiento dinámico gracias a los componentes, en otras palabras tenemos unas rutas que renderizan un componente.

Beneficios de React Router

- Establecer rutas en nuestra aplicación ej: Home, About, User.
- Realizar redirecciones
- Acceso al historial del navegador
- Manejo de rutas con parámetros
- Páginas para el manejo de errores como 404

2. React

Componentes

- **BrowserRouter**
- Este componente es el encargado de envolver nuestra aplicación dándonos acceso al API historial de HTML5 (pushState, replaceState y el evento popstate) para mantener su UI sincronizada con la URL.
- **Switch**
- Este componente es el encargado de que solo se renderice el primer hijo **Route** o **Redirect** que coincide con la ubicación. Si no usar este componente todos los componentes Route o Redirect se van a renderizar mientras cumplan con la condición establecida.
- **Route**
- Con **Route** podemos definir las rutas de nuestra aplicación, quizás sea el componente más importante de React Router para llegar a comprender todo el manejo de esta librería. Cuando definimos una ruta con Route le indicamos que componente debe renderizar.

2. React

Componentes

- **Path:** la ruta donde debemos renderizar nuestro componente podemos pasar un string o un array de string.
- **Exact:** Solo vamos a mostrar nuestro componente cuando la ruta sea exacta. Ej: /home === /home.
- **Strict:** Solo vamos a mostrar nuestro componente si al final de la ruta tiene un slash. Ej: /home/ === /home/
- **Sensitive:** Si le pasamos true vamos a tener en cuenta las mayúsculas y las minúsculas de nuestras rutas. Ej: /Home === /Home
- **Component:** Le pasamos un componente para renderizar solo cuando la ubicación coincide. En este caso el componente se monta y se desmonta no se actualiza.
- **Render:** Le pasamos una función para montar el componente en línea.

2. React

Creando nuestro proyecto

Para crear nuestro proyecto como lo mencione antes vamos a usar `create-react-app` para eso ejecutamos el siguiente comando en la terminal.

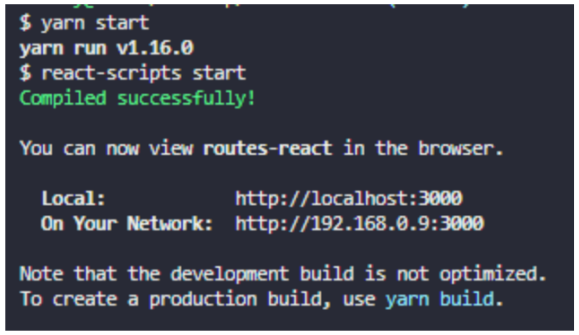
```
npx create-react-app routes-react
```

Esperamos que instale y cree nuestro proyecto y accedemos desde la terminal con:

```
cd routes-react
```

Abrimos nuestro editor de código si estas usando **VScode** puede usar la terminal parado en la raíz de nuestro proyecto usamos `code .` para abrirlo desde VScode.

Si queremos probar nuestro proyecto podemos correr **npm start** o **yarn start** desde la terminal. Hasta el momento tenemos un proyecto de react listo para modificar y agregar nuestros componentes e instalar **React Router**.



2. React

Instalación de React Router

Para instalar la librería solo tenemos que ir a la terminal estar ubicados en la raíz de nuestro proyecto y ejecutar el siguiente comando.

```
npm install react-router-dom
```

```
yarn add react-router-dom
```

2. React

Trabajando con React Router

Teniendo todo listo ahora si vamos a nuestro editor de código y abrimos el archivo **App.js** que está ubicado en `src/App.js` acá vamos a limpiar muchas cosas hasta que al final tengamos algo como el siguiente código.

```
import React from 'react';
import './App.css';
import {
  BrowserRouter as Router,
  Route
} from "react-router-dom";
import Home from './pages/Home'

function App() {
  return (
    <Router>
      <div className="App">
          <Route exact path="/" component={Home} />
      </div>
    </Router>
  );
}

export default App;
```

2. React

Importamos nuestro componente **BrowserRouter** le damos un nombre **Router** también importamos **Route** de `react-router-dom`. Envolvemos nuestra aplicación con Router y definimos nuestra primera ruta en este caso nuestro home le indicamos que debe ser exacta la ruta y que haga render de nuestro componente Home pero donde esta nuestro componente Home bueno vamos a crearlo.

```
// Home.js
import React from 'react'


const Home = () => (
  <section className="Home">
    <h3>Hello Home</h3>
  </section>
)


export default Home
```

2. React

Uso de los componentes Link y NavLink para navegar a una ruta

- React-Router proporciona los componentes <Link> y <NavLink>, que le permiten navegar a diferentes rutas definidas en la aplicación. Estos componentes de navegación se pueden considerar como enlaces de anclaje en la página que le permiten navegar a otras páginas del sitio. En un sitio web tradicional, cuando navega a través de la aplicación utilizando enlaces de anclaje, se actualiza la página y todos los componentes de la página se vuelven a representar. Los enlaces de navegación creados con <Link> y <NavLink> no generan una actualización de la página; solo se actualizan ciertas secciones de la página que se definen usando <Route> y que coinciden con la ruta URL.

2. React

El componente <Link>

- Un componente <Link> se usa para navegar a una <indexentry content=" componente:about"> ruta existente que se define usando el componente <Route>. Para navegar a una ruta, especifique el nombre de la ruta utilizada en la ruta como un valor para la prop:

2. React

En una aplicación típica de React, los datos se pasan de arriba hacia abajo (de padre a hijo) a través de *props*, pero esta forma puede resultar incómoda para ciertos tipos de *props* (por ejemplo, localización, el tema de la interfaz) que son necesarias para muchos componentes dentro de una aplicación. Context proporciona una forma de compartir valores como estos entre componentes sin tener que pasar explícitamente una *prop* a través de cada nivel del árbol.

- Cuándo usar Context
- Antes de usar Context
- API
    - React.createContext
    - Context.Provider
    - Class.contextType
    - Context.Consumer
    - Context.displayName
- Ejemplos
    - Context dinámico
    - Actualizando Context desde un componente anidado
    - Consumiendo múltiples Contexts
- Advertencias
- API antigua

2. React

Cuándo usar Context

Context está diseñado para compartir datos que pueden considerarse “globales” para un árbol de componentes en React, como el usuario autenticado actual, el tema o el idioma preferido. Por ejemplo, en el código a continuación, pasamos manualmente una *prop* de “tema” para darle estilo al componente *Button*:

```
class App extends React.Component {
  render() {
    return <Toolbar theme="dark" />;
  }
}

function Toolbar(props) {
  // El componente Toolbar debe tener un prop adicional "theme"
  // y pasarlo a ThemedButton. Esto puede llegar a ser trabajoso
  // si cada botón en la aplicación necesita saber el tema,
  // porque tendría que pasar a través de todos los componentes.
  return (
    <div>
      <ThemedButton theme={props.theme} />
    </div>
  );
}

class ThemedButton extends React.Component {
  render() {
    return <Button theme={this.props.theme} />;
  }
}
```