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

¿Qué es React JS?

2. React

React es una biblioteca o librería de código abierto que está escrita en JavaScript. Fue desarrollada por Facebook en el 2013 con la finalidad de facilitar la creación de componentes reutilizables e interactivos para las interfaces de usuario.

Uno de sus puntos más destacados es que no solo se usa en el lado del cliente, sino que también se puede representar en el servidor y trabajar juntos. Son datos interesantes que React se utiliza en Facebook para la producción de componentes y que Instagram está escrito enteramente en React.

Adicionalmente, también se utiliza en otras plataformas como Netflix, PayPal, AirBnb, Uber, Reddit y Twitter.

React es una excelente alternativa para realizar todo tipo de aplicaciones web o para dispositivos móviles, así como para crear *single page applications* (SPA o aplicaciones de una sola página).

También se destaca por contar con un completo ecosistema de componentes, herramientas y módulos que posibilitan el desarrollo de funcionalidades complejas en poco tiempo.

¿Cómo funciona React JS?

2. React

Para comprender cómo funciona React es clave que nos situemos en un contexto, pues cuando se aprende desarrollo web se obtiene conocimiento de tres conceptos básicos:

•HTML: la semántica, estructura e información de la página web; es decir, su esqueleto.
•CSS: la apariencia de nuestra página web.
•JavaScript: básicamente es el cerebro de nuestra página. Determina qué hacer en función de lo que sucede en ella.

Sin embargo, antes de React estos conceptos funcionaban por separado, en diferentes archivos y carpetas, por lo que escalar y extraer diversas partes del código para reutilizar o migrar funcionalidades era más complicado.

Por esto, los ingenieros de Facebook decidieron incluir todo en un solo paquete, al que llamaron "componente". En los componentes, la estructura HTML y JS son inseparables, y combinables con CSS.

Esto hizo posible la implementación de una nueva notación que hace más eficiente el desarrollo de aplicaciones escalables: **JSX, que significa JavaScript XML**.

2. React

¿Qué es JSX?

Consiste en una extensión de la sintaxis de JavaScript que permite que compilemos un objeto JavaScript para mapear un elemento del DOM (Objeto del Documento). A través de JSX se crea un DOM virtual (que es una representación en memoria del DOM) que pesa muy poco y usa menos recursos.

Con esto, cuando React es informado de un cambio de estado, vuelve a ejecutar esas funciones y determina una nueva representación virtual de dicha página. Posterior a esto, traduce automáticamente ese resultado en los cambios del DOM necesarios, reflejando así la nueva presentación de la página.

A primera vista, esto suena como que fuera más lento que el enfoque JavaScript habitual de actualización de cada elemento según sea necesario. Sin embargo, detrás de escena, React JS tiene un algoritmo muy eficiente para determinar los aspectos desiguales entre la representación virtual de lo que es la página actual y la nueva.

2. React

¿Qué es JSX?

A partir de esas diferencias, crea el conjunto mínimo de cambios necesarios en el DOM.

Para esto, utiliza un concepto llamado DOM virtual, el cual selecciona subárboles de los nodos sobre la base de cambios de estado. De esta manera mantiene los componentes actualizados con la menor cantidad de manipulación DOM posible.

¿Cómo funciona el DOM virtual?

2. React

Imaginar que se tiene un objeto que es un modelo en torno a una persona. Se Tienen todas las propiedades relevantes de una persona que podría tener, y refleja el estado actual de la persona. Esto es básicamente lo que React hace con el DOM.

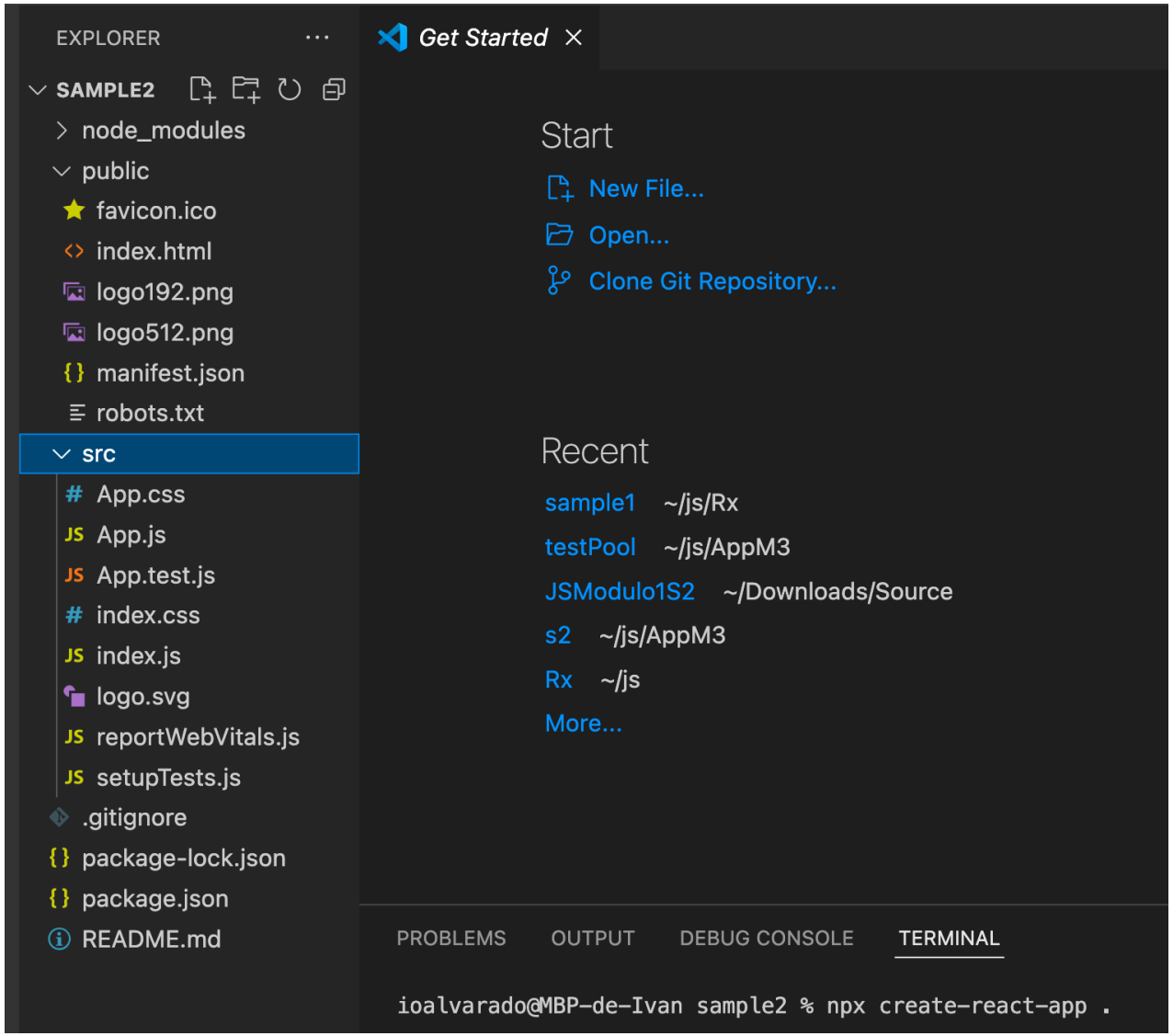
Ahora piensa que si se hace algun cambio, es decir, le añadimos bigote y tatuajes en los brazos, es claro que lo notarás. Y React también lo hace.

Entonces, cuando se aplican estos cambios, ocurren dos cosas a continuación:

- En primer lugar, React ejecuta un algoritmo de *diffing*, que identifica lo que ha cambiado; y
- El segundo paso es la reconciliación, donde se actualiza el DOM con los resultados de *diff*.

En pocas palabras, React JS está construido para tomar las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante.
Lo que hace React ante las variaciones —en lugar de tomar la imagen real y reconstruirla desde cero— es tomar los cambios de la cara y los brazos (o cualquiera que sea el cambio que se haya realizado). Esto significa que si tenías el texto en una entrada y se llevó a cabo una actualización con React, el texto no sería cambiado.

2. React


EXPLORER
SAMPLE2
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
Get Started
Start
New File...
Open...
Clone Git Repository...
Recent
sample1 ~/js/Rx
testPool ~/js/AppM3
JSModulo1S2 ~/Downloads/Source
s2 ~/js/AppM3
Rx ~/js
More...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
ioalvarado@MBP-de-Ivan sample2 % npx create-react-app .

2. React

Carpetas

1. public : se ubican los recursos gráficos y la plantilla index.html (Layout de la SingleApp)

```
<!DOCTYPE html>
<html lang="en">
  <head>…
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template
```

1. src : El código fuente de la app.

Carpetas src

2. React

App.js

src > App.js > App

```
1   import logo from './logo.svg';
2   import './App.css';
3
4   function App() {
5     return (
6       <div className="App">
7         <header className="App-header">
8           <img src={logo} className="App-logo" alt="logo" />
9           <p>
10            Edit <code>src/App.js</code> and save to reload.
11          </p>
12          <a
13            className="App-link"
14            href="https://reactjs.org"
15            target="_blank"
16            rel="noopener noreferrer"
17          >
18            Learn React
19          </a>
20        </header>
21      </div>
22    );
23  }
24
```

Carpetas src

2. React

JS index.js

src > JS index.js > ...

```
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import './index.css';
4   import App from './App';
5   import reportWebVitals from './reportWebVitals';
6
7   const root = ReactDOM.createRoot(document.getElementById('root'));
8   root.render(
9     <React.StrictMode>
10      <App />
11    </React.StrictMode>
12  );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18
```

Carpetas src

2. React

index.js

src > index.js > ...

```
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import './index.css';
4   import App from './App';
5   import reportWebVitals from './reportWebVitals';
6
7   const root = ReactDOM.createRoot(document.getElementById('root'));
8   root.render(
9     <React.StrictMode>
10      <App />
11    </React.StrictMode>
12  );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18
```

Carpetas src

2. React

Index.html
root

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(<div>Hola Mundo</div>,document.getElementById('root'));
```

2. React

Carpetas src

App.js

```
import React  from 'react';

export function App(){
        return(<div>Hola Mundo</div>);
}
```

Index.html

```
<div>root</div>
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
Import {App} from './App'

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

2. React

Hooks

- La sintaxis de desestructuración de un array permite dar diferentes nombres a las variables de estado que se declaran llamando a useState. Estos nombres no forman parte de la API useState. En su lugar, React asume que si llamas a useState muchas veces, se hace en el mismo orden durante cada renderizado.
- Los Hooks son funciones que permiten "enganchar" el estado de React y el ciclo de vida desde componentes de función. Los hooks no funcionan dentro de las clases. Permiten usar React sin clases.

2. React

Hooks

- React proporciona algunos Hooks incorporados como useState.
- También se pueden crear Hooks para reutilizar el comportamiento con estado entre diferentes componentes.

2. React

A medida que tu aplicación crece, puedes capturar una gran cantidad de errores con verificación de tipos. Para algunas aplicaciones, puedes usar extensiones de Javascript como Flow o TypeScript para verificar los tipos en tu aplicación. Pero incluso si no usas alguno de ellos, React tiene algunas habilidades de verificación de tipos incorporadas. Para usar verificación de tipos en las props de un componente, puedes asignar la propiedad especial `PropTypes`:

```
import PropTypes from 'prop-types';

class Greeting extends React.Component {
  render() {
    return (
      <h1>Hello, {this.props.name}</h1>
    );
  }
}

Greeting.propTypes = {
  name: PropTypes.string
};
```

2. React

Solicitar un sólo hijo

Usando `PropTypes.element` puedes especificar que únicamente un hijo se pase al componente.

```
import PropTypes from 'prop-types';

class MyComponent extends React.Component {
  render() {
    // Debe ser exactamente un elemento o generará una advertencia
    const children = this.props.children;
    return (
      <div>
        {children}
      </div>
    );
  }
}

MyComponent.propTypes = {
  children: PropTypes.element.isRequired
};
```

2. React

Valores por defecto de props

Puedes definir los valores por defecto de tus props al asignar la propiedad especial `defaultProps`:

```
class Greeting extends React.Component {
  render() {
    return (
      <h1>Hello, {this.props.name}</h1>
    );
  }
}

// Especifica los valores por defecto de props:
Greeting.defaultProps = {
  name: 'Stranger'
};

// Renderiza "Hello, Stranger":
ReactDOM.render(
  <Greeting />,
  document.getElementById('example')
);
```

2. React

Si estás usando un transform de Babel como plugin-proposal-class-properties (anteriormente *plugin-transform-class-properties*), puedes declarar `defaultProps` como una propiedad estática al interior de un componente clase de React. Esta sintaxis no se ha terminado y require un paso de compilación para funcionar en el navegador. Para mas información, puedes ver class field proposal.

```
class Greeting extends React.Component {
  static defaultProps = {
    name: 'stranger'
  }

  render() {
    return (
      <div>Hello, {this.props.name}</div>
    )
  }
}
```

2. React

Manejando eventos

- Manejar eventos en elementos de React es muy similar a manejar eventos con elementos del DOM. Hay algunas diferencias de sintaxis:
- Los eventos de React se nombran usando camelCase, en vez de minúsculas.
- Con JSX pasas una función como el manejador del evento, en vez de un string.
- Por ejemplo, el HTML:

```
<button onclick="activateLasers()">
    Activate Lasers
</button>
```

2. React

Manejando eventos

Por ejemplo, el React:

```
<button onclick={activateLasers}>
    Activate Lasers
</button>
```

Otra diferencia es que en React no puedes retornar false para prevenir el comportamiento por defecto. Debes, explícitamente, llamar preventDefault. Por ejemplo, en un HTML plano, para prevenir el comportamiento por defecto de enviar un formulario, puedes escribir:

```
<form onsubmit="console.log('You clicked submit.'); return false">
    <button type="submit">Submit</button>
</form>
```

2. React

Manejando eventos

Por ejemplo, el React:

```
function Form() {
    function handleSubmit(e) {
        e.preventDefault();
        console.log('You clicked submit.');
    }
    return (
        <form onSubmit={handleSubmit}>
        <button type="submit">Submit</button>
        </form>
    );
}
```

2. React

Manejando eventos

- Aquí, e es un evento sintético. React define estos eventos sintéticos acorde a las especificaciones W3C, para que no tengas que preocuparte por la compatibilidad entre distintos navegadores. Los eventos de React no funcionan exactamente igual que los eventos nativos. Mira la guía de referencia SyntheticEvent para aprender más.
- Cuando estás utilizando React, generalmente no necesitas llamar addEventListener para agregar escuchadores de eventos a un elemento del DOM después de que este es creado. En cambio, solo debes proveer un manejador de eventos cuando el elemento se renderiza inicialmente.
- Cuando defines un componente usando una clase de ES6, un patrón muy común es que los manejadores de eventos sean un método de la clase. Por ejemplo, este componente Toggle renderiza un botón que permite al usuario cambiar el estado entre "ENCENDIDO" y "APAGADO":