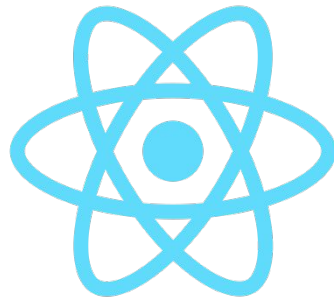




Universidad
Nacional
de Quilmes

React

Conceptos y características



CONSTRUCCIÓN DE INTERFACES DE USUARIO

2do Cuatrimestre de 2019

Repasando React...

- ⦿ Es una biblioteca para JavaScript creada por Facebook.
- ⦿ Es una herramienta para construir componentes de interfaces de usuario.
- ⦿ En vez de manipular directamente el DOM, crea un Virtual DOM en memoria, sobre el cual hace manipulaciones que luego replica en el DOM del browser.
- ⦿ React determina qué cambió en su Virtual DOM y aplica los cambios de la manera más óptima que puede en el mejor momento que puede.

Actualizando DOM

```
function tick() {  
  const date = new Date()  
    .toLocaleTimeString()  
  const element = (  
    <div>  
      <h1>Hello, world!</h1>  
      <h2>It is {date}</h2>  
    </div>  
  );  
  ReactDOM.render(  
    element,  
    document.getElementById('root'));  
}  
setInterval(tick, 1000);
```

Hello, world!

It is 12:26:46 PM.

```
Console Sources Network Timeline  
▼ <div id="root">  
  ▼ <div data-reactroot>  
    <h1>Hello, world!</h1>  
    ▼ <h2>  
      <!-- react-text: 4 -->  
      "It is "  
      <!-- /react-text -->  
      <!-- react-text: 5 -->  
      "12:26:46 PM"  
      <!-- /react-text -->  
      <!-- react-text: 6 -->  
      "."  
      <!-- /react-text -->  
    </h2>  
  </div>  
</div>
```

Components & Props

- © Los componentes permiten dividir la UI en partes independientes y reutilizables.
- © Posibilitan pensar en cada parte de forma aislada.
- © Conceptualmente, los componentes son como las funciones JavaScript.
- © Aceptan parámetros arbitrarios (*props*) y retornan **elementos** React que describen lo que debe aparecer en la pantalla.

Components & Props

```
// props = { name: "Jon" }  
const Welcome = (props) =>  
  <h1>Hello, {props.name}</h1>;  
  
const element = <Welcome name="Jon" />;  
ReactDOM.render(  
  element,  
  document.getElementById('root')  
)  
  
// Los componentes (por convención) empiezan  
// con la primer letra mayúscula.
```

Composición de Componentes

```
const Welcome = (props) =>  
  <h1>Hello, {props.name}</h1>;
```

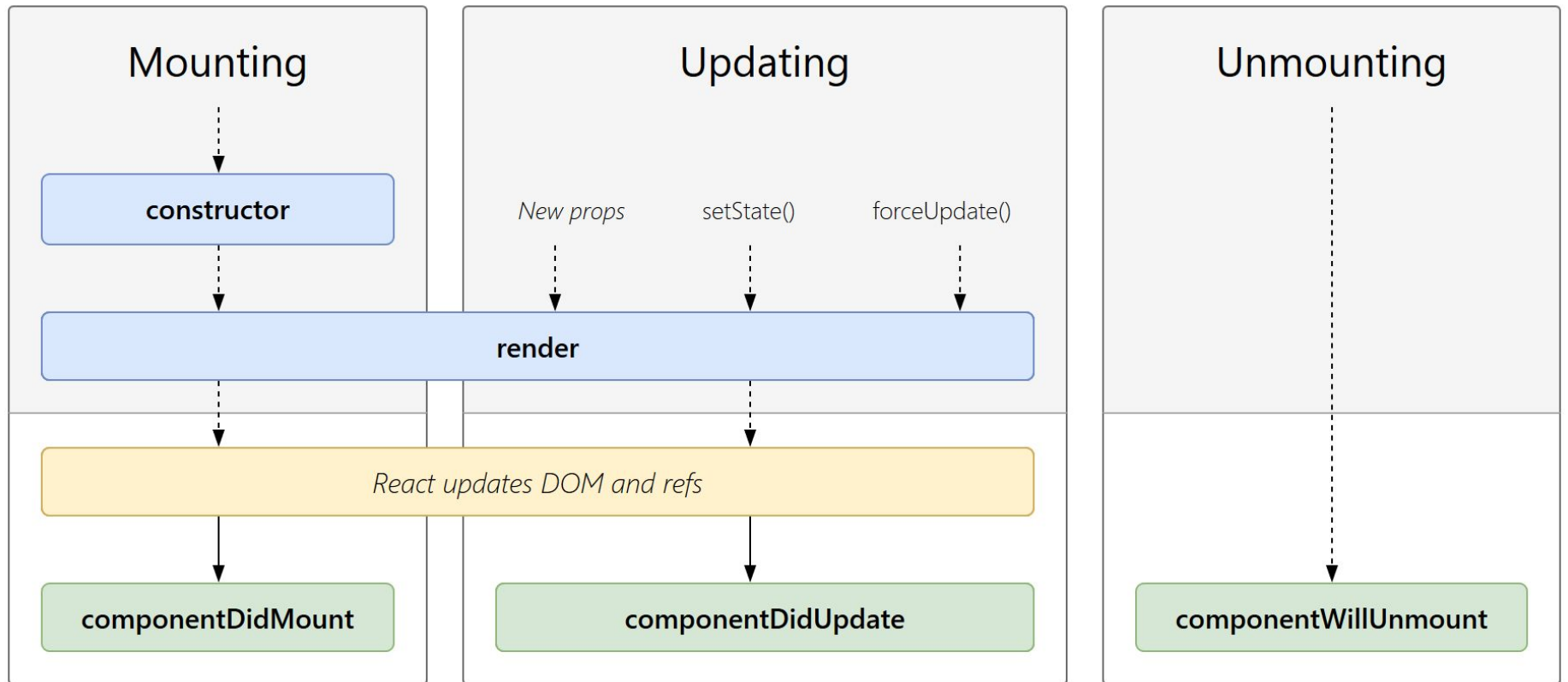
```
function App() {  
  return (  
    <div>  
      <Welcome name="Jon" />  
      <Welcome name="Dany" />  
      <Welcome name="Tyrion" />  
    </div>  
  );  
}  
ReactDOM.render(  
  <App />, document.getElementById('root')  
);
```

State

- ◎ React permite que sus componentes mantengan un estado propio para que puedan funcionar de forma interactiva.
- ◎ Cuando un componente modifica su estado, el framework recibe el aviso y vuelve a renderizar el componente
- ◎ Pero de forma eficiente, solo los elementos que cambiaron.

Lifecycle

- ◎ React mantiene un flujo ejecución el cual permite definir funciones extras que se ejecutan una vez renderizado el componente.
- ◎ Cada fase tiene una función asociada que podemos implementar. Existen tres fases:
 - **Mounting** » `componentDidMount()`
 - **Updating** » `componentDidUpdate()`
 - **Unmounting** » `componentWillUnmount()`



State & Lifecycle (I)

```
class Clock extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { date: new Date() };  
  }  
  render() {  
    return (  
      <div>  
        <h1>Hello, world!</h1>  
        <h2>It is  
          {this.state.date.toLocaleTimeString()}.  
        </h2>  
      </div>  
    );  
  }  
}  
ReactDOM.render(<Clock />, document.getElementById('root'));
```

State & Lifecycle (II)

```
class Clock extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { date: new Date() };  
  }  
  componentDidMount() {  
    this.timerID = setInterval(() => this.tick(), 1000);  
  }  
  componentWillUnmount() { clearInterval(this.timerID); }  
  
  tick() { this.setState({ date: new Date() }); }  
  
  render() { /* ... */ }  
}  
  
ReactDOM.render(<Clock />, document.getElementById('root'));
```

Events

- ◎ React permite manejar los eventos asociados a los elementos del DOM (onclick, onmouseover, ...)
- ◎ JS: `<button onclick="fire()">Fire!</button>`
- ◎ React: `<button onClick={fire}>Fire!</button>`
- ◎ ¡Bind this! Como las funciones en JS asocian this a quien ejecuta la función, perdemos contexto en los eventos. Para solucionar eso hay dos posibilidades:
 - `this.fire = this.fire.bind(this)` si
 - ◎ `fire() { ... } // normal function`
 - Usar directamente arrow functions
 - ◎ `fire = () => { ... }`

Events



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

    // El binding es necesario para que `this` tenga el contexto
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() { this.setState(state => ({
    isToggleOn: !state.isToggleOn }));
  }
  render() {
    return (<button onClick={this.handleClick}>
      {this.state.isToggleOn ? 'ON' : 'OFF'}</button>
    );
  }
}
```

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

Conditional

```
class LoginControl extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isLoggedIn: false};
  }
  handleLoginClick = () => this.setState({isLoggedIn: true});
  handleLogoutClick = () => this.setState({isLoggedIn: false});
  render() {
    const isLoggedIn = this.state.isLoggedIn;
    let button;
    if (isLoggedIn) {
      button = <LogoutButton onClick={this.handleLogoutClick} />;
    } else {
      button = <LoginButton onClick={this.handleLoginClick} />;
    }
    return (<div><Greeting isLoggedIn={isLoggedIn} />{button}</div>);
  }
}
```

Keys

```
const ListItem = props => <li>{props.value}</li>;
```

```
const NumberList = ({numbers}) => {  
  const listItems = numbers.map((number) =>  
    <ListItem key={number.toString()} value={number} />);  
  
  return (  
    <ul>  
      {listItems}  
    </ul>  
  );  
}
```

```
const numbers = [1, 2, 3, 4, 5];  
ReactDOM.render(<NumberList numbers={numbers} />,  
  document.getElementById('root'));
```

Routing

- ⦿ Dado que React renderiza los componentes mediante Virtual DOM, para navegar entre las diferentes páginas no podemos (bah, no debemos) usar los clásicos `Link`
- ⦿ Ese tipo de links “salen” de React y vuelven a “entrar” (renderizando la app por completo)
- ⦿ Hay que definir entonces un sistema de **Ruteo**
- ⦿ Y utilizar un componente propio para los links, llamado `<Link to="/ruta" />`
- ⦿ Es necesario instalar la dependencia **react-route-dom**

Routing

```
export default class App extends React.Component {  
  render() { return (  
    <BrowserRouter>  
      <Root>  
        <Switch>  
          <Route path="/page/redirect" component={Redirect} />  
          <Route path="/page" component={Page} />  
          <Route path="/search/:text"><Search /></Route>  
          <Route exact path="/" children={<Home />} />  
          <Route path="*" children={<NotFound />} />  
        </Switch>  
      </Root>  
      <nav>  
        <Link to="/">Home</Link>  
        <Link to="/page">Page</Link>  
        <Link to="/page/redirect">Redirect</Link>  
      </nav>  
    </BrowserRouter>);  
  }  
}
```



Routing

```
const Redirect = () => {  
  let history = useHistory();  
  history.push("/");  
}
```

```
const Page = () => {  
  return (  
    <div>  
      Go to <Link to="/">Home</Link>  
    </div>  
  );  
};
```

```
const Home = () => <div>Home</div>
```

```
const NotFound = () => <div>404 Not Found</div>
```

Form

```
class Form extends React.Component {
  constructor(props) { super(props);
    this.state = {inputValue: ''}; }
  handleInputChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    event.preventDefault();
    /* hacer cosas como guardar data, llamar a API, etc... */
    let history = useHistory();
    history.push("/");
  }
  render() { return (
    <form onSubmit={this.handleSubmit}>
      <label>Name:
      <input type="text" value={this.state.value}
        onChange={this.handleInputChange} /></label>
      <button type="submit" value="Submit" />
    </form>);
  }
}
```

Links

- © <https://es.reactjs.org/docs/getting-started.html>
- © <https://www.w3schools.com/react/default.asp>
- © <https://reacttraining.com/react-router/web/guides/quick-start>