CICLO 3

[FORMACIÓN POR CICLOS]

Desarrollo
Web

React js

Ingeni@ Soluciones TIC

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Agenda

1. JSX
2. Renderizando elementos
3. Componentes
4. Estado y ciclo de vida
5. Eventos
6. Renderizado condicional
7. Listas y keys
8. Formularios

1. JSX

Es una extensión de la sintaxis de Javascript. Se reutiliza en react para describir cómo luce la interfaz de usuario. JSX produce elementos de React.
En React la lógica de renderizado está unida a la lógica de interfaz de usuario

```
const name = 'Josh Perez';
const element = <h1>Hola, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

2. Renderización de elementos

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

setInterval(tick, 1000);
```

3. Componentes

Los componentes permiten separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada.

```
import React, {Component} from 'react';
import ListadoUsuarios from './ListadoUsuarios';
import Usuario from './Usuario';

export default class MainUsuarios extends Component {
  constructor(props) {
    super(props);
    this.state = { opcion: 'Listado' };
    this.setOpcion = this.setOpcion.bind(this);
  }

  setOpcion(nombreOpcion) {
    this.setState({ opcion: nombreOpcion });
  }

  render() {
    let componente =
      this.state.opcion === 'Listado' ? <ListadoUsuarios setOpcion={this.setOpcion}/> : <Usuario setOpcion={this.setOpcion}/>;
    return componente;
  }
}
```

3. Componentes

Componentes funcionales y de clase

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

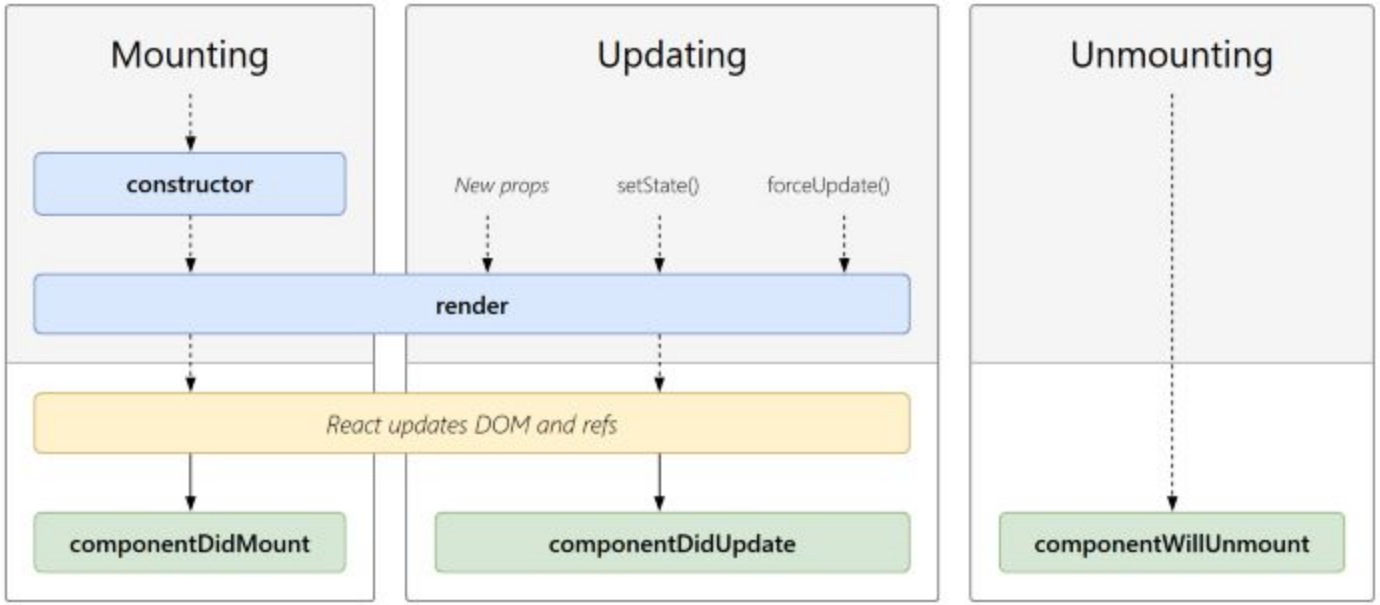
```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

4. Estado y ciclo de vida

El estado corresponde a los datos manejados de un componente. Este estado está encapsulado en el componente y no puede ser accedido normalmente por los componentes padre o hijos. Vamos a ver manejo de estado en componentes de tipo clase.

```
this. state = {...};
...
setState({...});
```

4. Estado y ciclo de vida



https://medium.com/codex/the-lifecycle-of-a-react-component-8e01332a068d

5. Eventos

- Los eventos en react se nombran usando camelCase.
- Con JSX se pasa una función como manejador del evento en vez de un string.

En HTML:

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

En React:

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

6. Renderizado condicional

En React podemos encapsular el comportamiento que requerimos renderizando los componentes basados en condiciones. Podemos usar un operador condicional mostrando los elementos o componentes basados en el estado actual.

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}
```

7. Listas y keys

Es muy normal tener un listado de elementos y necesitar generar una interfaz que nos renderice ese listado. Hay que tener en cuenta que cada elemento de lista debe tener un atributo “key” con valor único para que React identifique qué ítems han cambiado.

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
);
```

8. Formularios

Los elementos del formulario conservan un estado interno y es importante saber cómo obtenemos ese estado. (Aunque ya aprendimos cómo).

```
<select value={this.state.value} onChange={this.handleChange}>
  <option value="grapefruit">Grapefruit</option>
  <option value="lime">Lime</option>
  <option value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

Referencias

https://es.reactjs.org/