6 - Componentes

Las componentes son una característica fundamental de React. Nos permiten dividir la aplicación en trozos de nuestra interfaz visual con objetivos bien definidos: 'menú de opciones', 'formulario de búsqueda', 'ventanas de mensajes', 'tablas de datos' etc.

La división de nuestra aplicación en componentes nos ayudan a reutilizar dicho código en proyectos futuros.

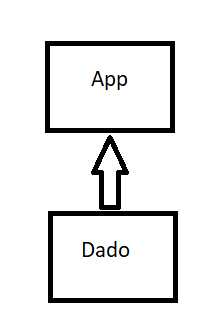
Hasta ahora cada proyecto que hemos desarrollado ha implementado una sola componente llamada 'App'

Problema

Crear un nuevo proyecto con la herramienta create-react-app llamado **proyecto006**

Confeccionar una aplicación que muestre tres dados.

Plantear una clase llamada 'Dado' que tenga una propiedad llamada 'valor' a la cual le llega el dato a mostrar.  
Nuestra clase principal seguirá siendo 'App' y dentro de la misma definiremos 3 instancias de 'Dado':



Veamos todos los pasos que debemos dar para poder implementar este proyecto empleando 2 componentes:

* Primero creamos en la carpeta src dos archivos llamados 'Dado.js' y 'Dado.css'



* Codificamos la componente 'Dado' en el archivo 'Dado.js':
* import React, { Component } from 'react';
* import './Dado.css';
* class Dado extends Component {
* render() {
* return (
* <div className="Dado-recuadro">{this.props.valor}</div>
* );
* }
* }
* export default Dado;

Toda componente debe heredar de la clase Component e implementar el método render(), en este método mostramos el dato almacenado en la propiedad 'valor' que llegará desde la componente 'App' donde definamos los dados:

<div className="Dado-recuadro">{this.props.valor}</div>

Utilizamos estilos para que el dado aparezca mejor en pantalla. Los estilos los guardamos en el archivo 'Dado.css' y lo debemos importar para utilizarlo en el div:

import './Dado.css';

Como importaremos la clase 'Dado' luego en la clase 'App' debemos exportarla:

export default Dado;

También codificamos el archivo 'Dado.css':

.Dado-recuadro {

display: flex;

width: 2rem;

height: 2rem;

background-color: black;

color: white;

justify-content: center;

align-items: center;

margin: 1rem;

border-radius: 4px;

}

* Ahora codificamos la clase 'App':
* import React, { Component } from 'react';
* import './App.css';
* import Dado from './Dado'
* class App extends Component {
* render() {
* const valor1 = Math.trunc(Math.random()\*6)+1;
* const valor2 = Math.trunc(Math.random()\*6)+1;
* const valor3 = Math.trunc(Math.random()\*6)+1;
* return (
* <div>
* <Dado valor={valor1} />
* <Dado valor={valor2} />
* <Dado valor={valor3} />
* </div>
* );
* }
* }
* export default App;

Lo primero que debemos hacer para poder utilizar la componente Dado es importarla:

import Dado from './Dado'

Lo nuevo es que en el método render() definimos tres etiquetas Dado y pasamos la propiedad valor con distintos valores aleatorios previamente generados:

render() {

const valor1 = Math.trunc(Math.random()\*6)+1;

const valor2 = Math.trunc(Math.random()\*6)+1;

const valor3 = Math.trunc(Math.random()\*6)+1;

return (

<div>

<Dado valor={valor1} />

<Dado valor={valor2} />

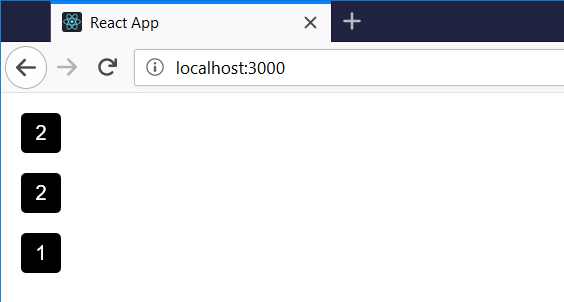
<Dado valor={valor3} />

</div>

);

}

Podemos ver la potencia que puede tener la declaración de componentes y luego crear tantas como necesitemos.

* Si ejecutamos ahora la aplicación desde la línea de comandos de Node.js 'npm start' tenemos como resultado: 

Problema

En el problema anterior si necesitamos sortear otros tres dados estamos obligados a recargar la página. Modificaremos la aplicación y agregaremos un botón para que cada vez que se presione nos actualice únicamente los valores de los tres dados sin tener que recargar en forma completa la página.

Solo debemos efectuar cambios en la clase App:

import React, { Component } from 'react';

import './App.css';

import Dado from './Dado'

class App extends Component {

constructor(props) {

super(props);

this.state = {

valor1: this.generarValor(),

valor2: this.generarValor(),

valor3: this.generarValor()

}

this.tirar=this.tirar.bind(this);

}

render() {

return (

<div>

<Dado valor={this.state.valor1} />

<Dado valor={this.state.valor2} />

<Dado valor={this.state.valor3} />

<button onClick={this.tirar}>Tirar</button>

</div>

);

}

tirar() {

this.setState({

valor1: this.generarValor(),

valor2: this.generarValor(),

valor3: this.generarValor()

});

}

generarValor() {

return Math.trunc(Math.random()\*6)+1;

}

}

export default App;

En el constructor almacenamos en la propiedad el estado de tres valores que representan el valor de cada dado:

constructor(props) {

super(props);

this.state = {

valor1: this.generarValor(),

valor2: this.generarValor(),

valor3: this.generarValor()

}

También enlazamos el método a ejecutar cuando se presione el botón:

this.tirar=this.tirar.bind(this);

En el método render() recuperamos del 'estado' los tres valores a pasar a cada componente 'Dado':

render() {

return (

<div>

<Dado valor={this.state.valor1} />

<Dado valor={this.state.valor2} />

<Dado valor={this.state.valor3} />

<button onClick={this.tirar}>Tirar</button>

</div>

);

}

El método tirar() modifica el 'estado' llamando al método setState, esto hace que se actualicen en pantalla los nuevos valores de cada dado sin tener que recargar toda la página:

tirar() {

this.setState({

valor1: this.generarValor(),

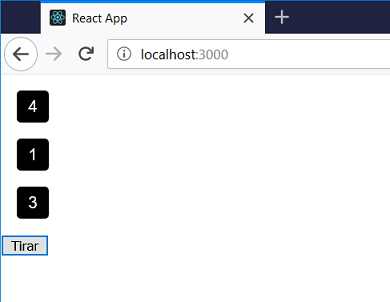
valor2: this.generarValor(),

valor3: this.generarValor()

});

}

En el navegador tenemos como resultado:



Acotaciones

Si queremos cambiar parte del 'estado' de la componente podemos llamar a setState y cambiar solo lo que necesitemos:

this.setState({

valor2: this.generarValor(),

valor3: this.generarValor()

});

Con esto solo cambiamos los valores de los atributos 'valor2' y 'valor3'. 'valor1' no se modifica.

[Retornar](https://www.tutorialesprogramacionya.com/reactya/index.php?inicio=0)