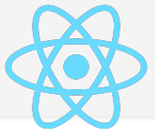


Tema 9

Librerías

Librerias

- En este tema veremos algunas de las librerías que hay interesantes para React. Las librerías son una manera muy cómoda de añadir funcionalidad a nuestra página, la mayoría son muy sencillas de implementar, de comprender y su diseño es bastante accesible e intuitivo.

- React tiene muchas librerías disponibles e información de como implementarlas en este enlace:

https://github.com/facebook/react/wiki/Complementary-Tools

React-notifications

- React-notifications es una libreria diseñada para proporcionar mensajes de notificación y una pila de mensajes acumulados que van desapareciendo después de un tiempo determinado.

- El estilo visual y la interacción determinados siguen las directrices del diseño material para snack-bar pero se pueden personalizar completamente.

React-notifications

- Para instalar esta librería tendremos que ejecutar en el directorio de la aplicación el siguiente comando:

npm install react-notification

- La aplicación se se basa en un elemento <**Notification/>** con unos atributos que definen como se comportara el mensaje.

- Dichos atributos se pueden modificar para cambiar el comportamiento del mensaje.

React-notifications

```
<Notification
  isActive={boolean}
  message={string}
  action={string}
  onClick={myClickHander}
/>
```

- i**sActive** define si el mensaje esta activo o no, es decir si es visible o no. Si le damos como valor **true** será visible y si le damos **false** no será visible.

- **message** es un string que define la cadena de texto que queremos que aparezca como mensaje.

React-notifications

- **action** es un string, el nombre de la acción que queramos que aparezca al lado derecho del mensaje.

- **onClick** es el evento al que asociaremos la función que queramos que ejecute cuando se haga click en el **action.**

- Un sencillo ejemplo de una notificación que se puede desactivar seria el siguiente:

React-notifications

```
import { Notification } from 'react-notification';

class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isActive: true,
    }
  }
```

- Aquí inicializamos las variables para que la notificación sea visible.

React-notifications

```
render() {
   const { isActive } = this.state;
   return (
     <div>
       <Notification
         isActive={this.state.isActive}
         message="Notification"
         action="Dismiss"
         title="Title!"
         onClick={() =>  this.setState({ isActive: false })}
       />
     </div>
    );
  }
}
export default App;
```

React-notifications

- Renderizamos el elemento **<Notification/>** cogiendo el estado de la variable **isActive** desde el constructor

- **title** es un string que define si queremos añadir titulo al mensaje.

- Al evento **onClick** le asociamos la función **setState** y establecemos la variable **isActive** a **false** de manera que una vez hagamos clic el mensaje no estará activo y no será visible.

React-notifications

- También podemos añadir un botón para que aparezcan las notificaciones cuando pulsemos.

```
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isActive: false,
    }
  }
  toggleNotification() {
    this.setState({
      isActive: !this.state.isActive
    })
  }
  render() {
    const { isActive } = this.state;
    return (
      <div>
        <button
          onClick={this.toggleNotification.bind(this)}
          children={ "Show notification" }
        />
```

React-notifications

- Por defecto definimos la variable isActive a false para que el mensaje no sea visible.
- Creamos la función toggleNotification( ) y en ella establecemos la variable isActive al valor diferente al que se encuentre en ese momento.
- En el render( ) creamos el botón y le asociamos al evento onClick la función toggleNotification( ).
- Todo esto permitirá que cuando se haga clic en el botón aparezca el mensaje y cuando se vuelva a clicar desaparezca.
- En los ejercicios veremos como crear una pila de mensajes de notificación que se acumulan.

React-slick

- React-slick es una librería diseñada para implementar carruseles o sliders de de forma sencilla a nuestra aplicación. Los carruseles o sliders son elementos muy intuitivos con respecto a la navegación y presentación de elementos en un pagina.

- Esta librería nos permite personalizar nuestro carrusel o slider de diferentes modos.

React-slick

- Para instalar esta librería tendremos que ejecutar en el directorio de la aplicación el siguiente comando:

npm install react-slick

- Ademas habrá que introducir en el html de la ruta **“tu aplicacion”/public/index.html** los siguientes cdn:

<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css" />
<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css" />

React-slick

- La aplicación se se basa en un elemento <**Slider/>** acompañado de un objeto que define el estilo del slíder o carrusel.

- Dicho objeto se le añaden atributos que permiten cambiar la manera en la que se comporta o la manera en la que se visualiza el slider.

```
<Slider {...settings}>
     <div><h3>1</h3></div>
     <div><h3>2</h3></div>
     <div><h3>3</h3></div>
     <div><h3>4</h3></div>
     <div><h3>5</h3></div>
     <div><h3>6</h3></div>
</Slider>
```

React-slick

- En este caso el objeto es el llamado settings. Es definido dentro del render( ).

```
render(){
  var settings = {
     dots: true,
     infinite: true,
     speed: 500,
     slidesToShow: 1,
     slidesToScroll: 1
};
```

- **Dots** es un valor booleano que define si van a aparecer unos puntos debajo de los elementos con los que puedes navegar entre estos.

React-slick

- **Infinite** es un valor **booleano** que si le pasamos **true** como valor cuando arrastramos el último elemento hacia adelante volveremos al primer elemento, en cambio si le pasamos **false** como valor no podremos pasar al siguiente elemento. Por defecto este atributo si no se incluye es como si estuviera y con el valor **true**.

- **Speed** es un **int** que define la velocidad en mili segundos con la se cambia entre elementos.

- **SlideToShow** es un **int** que define cuantos elementos se van mostrar en cada slide.

- **SlideToScroll** es un **int** que define a cuantos elementos se les va a hacer scroll cuando cambiemos de slide.

React-slick

- Existen otros atributos que se pueden añadir para cambiar el comportamiento y la visualización del slider:

- **Autoplay** es un valor booleano que define que si el valor pasado es true cambiara automáticamente de slide pasados unos segundos, estos segundos se pueden determinar con el atributo Autospeed.

- Para poder centrar los elementos se indica con el atributo **centerMode** pasándole true que el elemento esté centrado y con el atributo **centerPadding** pasándole un valor numérico en porcentaje o en pixeles para indicar el espacio que hay entre el centro y el lado izquierdo.

React-slick

- **fade** es un valor booleano que cambia la transición actual entre elementos a otra con un efecto de desvanecimiento.

- **vertical** es un valor booleano que permite cambiar la disposición de los slides a modo vertical.

- **verticalSwiping** es un valor booleano que permite arrastrar de abajo a arriba cuando están los slides en modo vertical.

Video-react

Video-react es una librería muy sencilla mediante la cual podemos añadir vídeos a nuestra web con estilos que siguen la mayoría de reproductores e iconos de Google Material.

Toda la información sobre esta librería se puede consultar en el siguiente repositorio de gitHub:

https://github.com/video-react/video-react

O accediendo a su página web:

https://video-react.js.org/components/player/

Video-react

Lo primero que tenemos que hacer es instalar las dependencias necesarias, las cuales se pueden instalar accediendo al proyecto desde la consola e introduciendo el siguiente comando:

npm install --save video-react react react-dom redux

Dentro de la clase donde queramos usar el componente podemos importar los estilos y el propio componente de la siguiente forma:

```
import { Player } from 'video-react';
import '../node_modules/video-react/dist/video-react.css'
```

Video-react

Algunos de los atributos y métodos más importantes o usados son los siguientes:

- **src:** la url del vídeo que queremos que se muestre, también puede usarse el tag <source>.
- **muted:** si el valor es true el sonido no se escucha.
- **autoPlay:** si es true el vídeo comienza a reproducirse automáticamente.
- **.volume:** modifica u obtiene el valor del volumen.
- **.muted:** modifica u obtiene si el vídeo tiene sonido o no.
- **.video:** devuelve el objeto vídeo.

Video-react

Estos métodos se utilizan para modificar y controlar el video del tag Player:

- **.play( ):** pone en marcha el vídeo.
- **.pause( ):** pausa el vídeo.
- **.load( ):** cambia la fuente del vídeo y carga este nuevo vídeo.
- **.replay(seconds):** vuelve hacia atrás un número especificado de **segundos**.

Video-react

Una vez tenemos importado tanto el componente como el css para que salga todo maquetado y siguiendo Material Design podemos usarlo muy fácilmente, basta con añadir un tag de tipo <Player> asignando la dirección de un vídeo.

```
<Player>
   <source src="http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4" />
</Player>
```

También podemos crear una lista de urls y usar la que más nos interese en cada momento, para esto usamos una lista, un state y en source añadimos el elemento del state.

```
const sources = {
  sintelTrailer: 'http://media.w3.org/2010/05/sintel/trailer.mp4',
  bunnyTrailer: 'http://media.w3.org/2010/05/bunny/trailer.mp4',
  bunnyMovie: 'http://media.w3.org/2010/05/bunny/movie.mp4',
  test: 'http://media.w3.org/2010/05/video/movie_300.webm',
};
```

Video-react

Creamos el constructor con el state y le asignamos el source deseado:

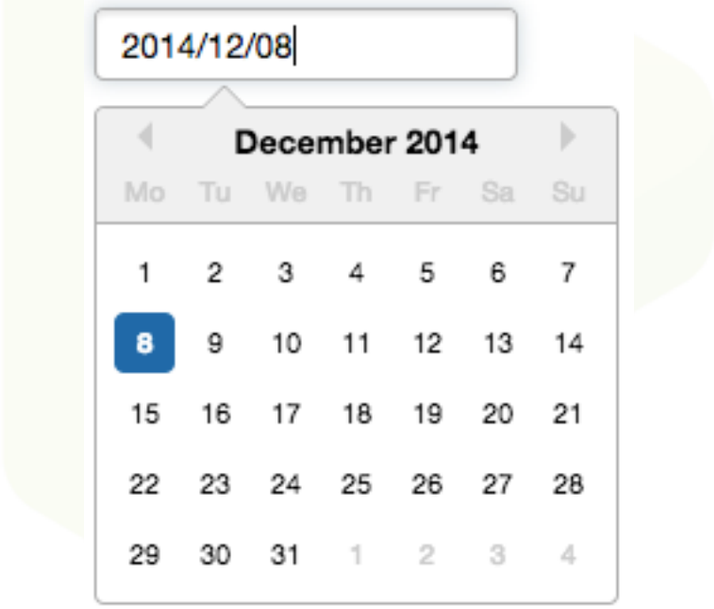
```
constructor() {
    super();
    this.state = {
      source: sources['bunnyMovie'],
    };
  }
```

En el método render solo tenemos que devolver el componente Player asignando como source el state.

```
<Player>
      <source src={this.state.source} />
</Player>
```

React Date Picker

React Date Picker es un componente simple y reusable para React que consiste en un selector de fechas.



Se puede instalar con la orden **$ npm install react-datepicker —save**.

React Date Picker

Se deberá importar **Moment.js** de manera separada a React ya que estas dependencias no están incluidas en el paquete. También será necesario requerir el archivo CSS de este paquete, aunque también se puede crear uno propio.

El siguiente ejemplo muestra cómo incluir el CSS de este paquete y cómo usar DatePicker en una vista React.

React Date Picker

```
var DatePicker = require('react-datepicker');
var moment = require('moment');

require('react-datepicker/dist/react-datepicker.css');

var Example = React.createClass({
  displayName: 'Example',

  getInitialState: function() {
    return {
      startDate: moment()
    };
  },

  handleChange: function(date) {
    this.setState({
      startDate: date
    });
  },

  render: function() {
    return <DatePicker
        selected={this.state.startDate}
        onChange={this.handleChange} />;
  }
});
```

React Date Picker

El uso más básico de DatePicker se puede describir con:

```
<DatePicker selected={this.state.date}
onChange={this.handleChange} />
```

Adicionalmente, hay soporte para el teclado:

- Izquierda: mover al día anterior.
- Derecha: mover al día siguiente.
- Arriba: mover a la semana anterior.
- Abajo: mover a la semana siguiente.
- PgUp: mover al mes anterior.
- PgDn: mover al mes siguiente.

TextArea redimensionable

Existe un módulo que nos permite que el tamaño del campo de texto de un componente **TextArea** se ajuste automáticamente cuando el texto cambie de tamaño. Se instala con **$ npm install react-textarea-autosize**.

Ejemplo de uso:

```
import Textarea from 'react-textarea-autosize';

React.renderComponent(
  <div>
    <Textarea></Textarea>
  </div>,
  document.getElementById('element'));
```