UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Centro de e-Learning
Secretaría de Cultura y Extensión Universitaria

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Diplomatura en programación web full stack con React JS

UTN.BA | Centro de e-Learning
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Secretaría de Cultura y Extensión Universitaria

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 4:

Introducción Reactjs y Nodejs

Unidad 1:

React (parte 1)

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Presentación

En esta unidad comenzamos a conocer ReactJs, una librería enfocada en el desarrollo de interfaces.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Objetivos

Que los participantes logren...

- Saber què es y para què sirve ReactJs.
- Entender y utilizar el lenguaje JSX.
- Crear y utilizar nuestros propios componentes..

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Bloques temáticos

1. Introducción.
2. JSX.
3. create-react-app.
4. Componentes.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

1. Introducción

ReactJS es una librería JavaScript de código abierto enfocada a la visualización que nos permite el desarrollo de interfaces de usuario de forma sencilla mediante componentes interactivos y reutilizables.

React está basado en un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar. Estas piezas se crean usando una sintaxis llamada JSX permitiendo escribir HTML (y opcionalmente CSS) dentro de objetos JavaScript.

Estos componentes son reutilizables y se combinan para crear componentes mayores hasta configurar una web completa.

Esta es la forma de tener HTML con toda la funcionalidad de JavaScript y el estilo gráfico de CSS centralizado y listo para ser abstraído y usado en cualquier otro proyecto.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

2. JSX

JSX es una extensión de sintaxis de Javascript, que como podremos observar se parece a HTML.

Uno de sus beneficios es mejorar la legibilidad del código, mejorar la experiencia del desarrollador, y reducir la cantidad de errores de sintaxis, ya que no es necesario repetir tantas veces el mismo código. Otro beneficio es el de poder integrar a otros miembros de tu equipo que no sean programadores en el desarrollo. Es "fácil" leer el código si ellos están acostumbrados a leer HTML.

React no requiere usar JSX, pero la mayoría de la gente lo encuentra útil como ayuda visual cuando trabajan con interfaz de usuario dentro del código Javascript. Esto también permite que React muestre mensajes de error o advertencia más útiles.

Insertando expresiones en JSX

```
1 const name = 'Josh Perez';
2 const element = <h1>Hello, {name}</h1>;
3
4 ReactDOM.render(
5   element,
6   document.getElementById('root')
7 );
```

En el ejemplo declaramos una variable llamada name y luego la usamos dentro del JSX envolviendola dentro de llaves

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

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

En el ejemplo insertamos el resultado de llamar la función de JavaScript, formatName(user), dentro de un elemento <h1>.

Especificando atributos con JSX

Podemos utilizar comillas para especificar strings literales como atributos:

```
const element = <div tabIndex="0"></div>;
```

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

También puedes usar llaves para insertar una expresión JavaScript en un atributo:

```
1 const element = <img src={user.avatarUrl}></img>;
```

Podemos usar comillas rodeando llaves cuando insertes una expresión JavaScript en un atributo. Debemos utilizar comillas (para los valores de los strings) o llaves (para las expresiones), pero no ambas en el mismo atributo.

Advertencia:

Dado que JSX es más cercano a JavaScript que a HTML, React DOM usa la convención de nomenclatura camelCase en vez de nombres de atributos HTML.

Por ejemplo, **class** se vuelve **className** en JSX, y **tabindex** se vuelve **tabIndex.**

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

3. create-react-app

create-react-app es un ambiente cómodo para aprender React, y es la mejor manera de comenzar a construir una nueva aplicación de página única usando React.

create-react-app configura tu ambiente de desarrollo de forma que puedas usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando tu aplicación para producción. Necesitarás tener Node mayor o igual a la versión 10.16 y npm mayor o igual a la versión 5.6 instalados en tu máquina.

Para crear un proyecto ejecuta:

```
create-react-app nombredemiproyecto

cd nombredemiproyecto

npm start
```

create-react-app no se encarga de la lógica de backend o de bases de datos; tan solo crea un flujo de construcción para **frontend**, de manera que lo puedes usar con cualquier backend

4. Componentes

Componentes de clase vs. componentes de función

Los componentes de React pueden definirse de dos formas distintas: como funciones o como clases.

```
1 class Saludo extends React.Component {
2   render() {
3     return <h1>Hola, {this.props.name}</h1>;
4   }
5 }
```

Ejemplo de componente de clase

```
1 function Saludo(props) {
2   return <h1>Hola, {props.name}</h1>;
3 }
```

Ejemplo de componente de función

Ambos componentes son equivalentes, sin embargo la forma más sencilla, y la más utilizada en las últimas actualizaciones es de función.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

El ejemplo de función es un componente de React válido porque acepta un solo argumento de objeto "props" (que proviene de propiedades) con datos y devuelve un elemento de React. Llamamos a dichos componentes "funcionales" porque literalmente son funciones JavaScript.

Creación de componentes

Como vimos anteriormente podemos definir un componente usando una simple función, y su valor de retorno será el contenido HTML que se muestre. Así mismo podemos incluir todos los componentes que necesitemos dentro del valor de retorno de otros componentes. Este concepto se llama composición.

```
1 function Saludo(props) {
2   return <h1>Hola, {props.name}</h1>;
3 }
4
5 function App() {
6   return (
7     <div>
8       <Saludo name="Sara" />
9       <Saludo name="Cahal" />
10      <Saludo name="Edite" />
11    </div>
12  );
13 }
```

Por lo general, las aplicaciones de React nuevas tienen un único componente App en lo más alto. Sin embargo, si se integra React en una aplicación existente, se podría empezar de abajo hacia arriba con un pequeño componente como Button y poco a poco trabajar el camino a la cima de la jerarquía de la vista.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Es importante que en cada archivo en el que vayamos a definir un componente (ya sea de clase o funcional recordemos importar React al principio de nuestro archivo js y exportarlo al final.

```
1 // components/MiComponente.js
2 import React from 'react'
3
4 const MiComponente = (props) => {
5     return <p>Soy un componente</p>;
6 };
7
8 export default MiComponente;
```

Asumiendo que este componente se encuentre en la carpeta componentes, dentro del archivo MiniComponente.js podríamos importarlo dentro de otro componente en la misma carpeta de la siguiente forma:

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

```
// components/OtroComponente.js
import React from 'react'
import MiComponente from './MiComponente'

const OtroComponente = (props) => {
    return (
        <>
            <h1>Soy Otro Componente</h1>
            <MiComponente/>
        </>
    )
}

export default OtroComponente;
```

En caso de ser archivos locales usamos siempre la ruta relativa desde el archivo en el que estamos escribiendo. No es necesario agregar la extensión a los archivos js al importarlos.

Centro de e-Learning SCEU UTN - BA. Medrano 951 2do piso
(1179) // Tel. +54 11 7078- 8073 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Bibliografía utilizada y sugerida

Artículos de revista en formato electrónico:

MDN Web Docs. Disponible desde la URL: https://developer.mozilla.org/

React.Disponible desde la URL: https://es.reactjs.org/

Libros y otros manuscritos:

Alex Banks & Eve Porcello. Learning React: desarrollo web funcional con React y Redux. 2017.