UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Desarrollo en React JS

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo III

Eventos, Firebase

Unidad 3: Desarrollar nuestra app en react js

Presentación:

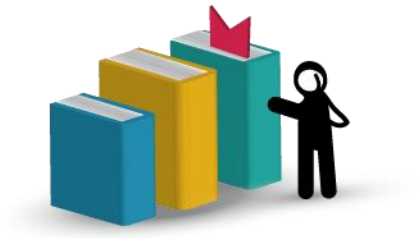
En esta unidad aprenderemos como utilizar condicionales al momento de renderizar, diferentes formas sintácticas que tenemos para realizar esto.

Por otro lado que es el helper keys y cómo implementarlo.

Objetivos:

Que los participantes*:

- Aprendan que es el renderizado condicional y las listas
- Apliquen renderizado condicional y utilicen keys



Bloques temáticos*:

- Renderizado condicional
- Listas
- Key

Consignas para el aprendizaje colaborativo

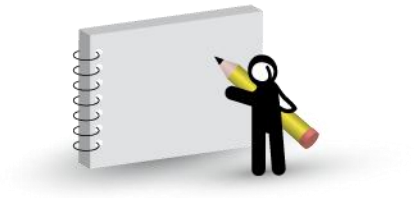
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

Renderizado condicional

En react podemos colocar condicionales al momento de realizar el renderizado.

Una forma de hacerlo es la siguiente

```
render() {
  if(true){
    return (
      <div>

      <h1>{this.state.nombre}</h1>
      </div>
    );
  }else{
    return (
      <div>

      <h1>{this.state.nombre} false</h1>
      </div>
    );
  }

}
```

Como podemos observar, dentro del método render colocamos el condicional en el cual colocamos un return dentro de cada posible camino a tomar.

Otra opción es manejar la condición a través de un estado, por ejemplo

```
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>
  );
}
```

En este caso vemos el estado isLoggedIn y la condición de acuerdo a ese valor.

Como vemos podemos asignar código JSX a una variable javascript y luego renderizar el contenido de esa variable.

React nos permite anidar la condición con el operador &&, veamos el siguiente ejemplo

```
return (
  <div>
    <h1>Hello!</h1>
    {unreadMessages.length > 0 &&
      <h2>
        You have {unreadMessages.length} unread messages.
      </h2>
    }
  </div>
);
```

Vemos que la primer parte (**unreadMessages.length > 0**) es la condición, si la misma es true se renderiza el contenido JSX que es lo que vemos a la derecha del **&&**

Por último también podemos utilizar el operador ternario:

```
render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      The user is <b>{isLoggedIn ? 'currently' : 'not'}</b> logged in.
    </div>
  );
}
```

Listas

Las listas o colecciones (arrays) las podemos iterar utilizando la función map (vista anteriormente)

Por ejemplo:

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

Como vemos la función map itera la lista **numbers** y mediante esta iteración renderiza el contenido de cada componente del array.

Keys

La key es un helper de react que nos permite saber que item ha cambiado, se ha eliminado o se ha agregado.

Siguiente el ejemplo anterior podemos ver la combinación de este helper con el método map

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

La mejor práctica es utilizar como **key** un atributo unívoco, por ejemplo un id:

```
const todoItems = todos.map((todo) =>
  <li key={todo.id}>
    {todo.text}
  </li>
);
```

En caso de no tener un id unívoco para conformar la key, podemos utilizar el index (aunque no es la mejor práctica o recomendada)

```
const todoItems = todos.map((todo, index) =>
  // Only do this if items have no stable IDs
  <li key={index}>
    {todo.text}
  </li>
);
```

En caso de recorrer 2 arrays (que tienen el mismo contenido pero se renderizan 2 veces) podemos utilizar la misma key, por ejemplo

```
function Blog(props) {
  const sidebar = (
    <ul>
      {props.posts.map((post) =>
        <li key={post.id}>
          {post.title}
        </li>
      )}
    </ul>
  );
  const content = props.posts.map((post) =>
    <div key={post.id}>
      <h3>{post.title}</h3>
      <p>{post.content}</p>
    </div>
  );
  return (
    <div>
      {sidebar}
      <hr />
      {content}
    </div>
  );
}
```

Como vemos el array tiene el mismo contenido, pero primero se renderiza el **sidebar** y luego el **content**

Bibliografía utilizada y sugerida

Fedosejev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt.

https://reactjs.org/docs/conditional-rendering.html

https://reactjs.org/docs/forms.html

https://reactjs.org/docs/lifting-state-up.html

https://reactjs.org/docs/lists-and-keys.html

Lo que vimos:

En esta unidad aprendimos como renderizar utilizando condicionales y las ventajas del helper key de react js

Lo que viene:

En la próxima unidad veremos el manejo de estados con react context