UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Diplomatura en programación web full stack con 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 5: React Inicial

Unidad 1: Ciclo de vida de los componentes + conexión con el servidor

Presentación:

En esta unidad vamos a aprender a conectarnos con el servidor. Al terminar esta unidad serán capaces de conectarse con su aplicación servidora creada en NodeJS!

Objetivos:

Que los participantes:

- Comprendan el ciclo de vida de los componentes en React.
- Sean capaces de redefinir los métodos básicos.
- Sean capaces de realizar la conexión con su servidor NodeJS.

Bloques temáticos:

1. Introducción al ciclo de vida de los componentes
2. Redefinición de métodos básicos
3. Introducción a AXIOS
4. Conexión con el servidor
5. Ejemplo
6. Trabajo Práctico

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.

**** El MEC es el modelo de E-learning colaborativo de nuestro Centro.***

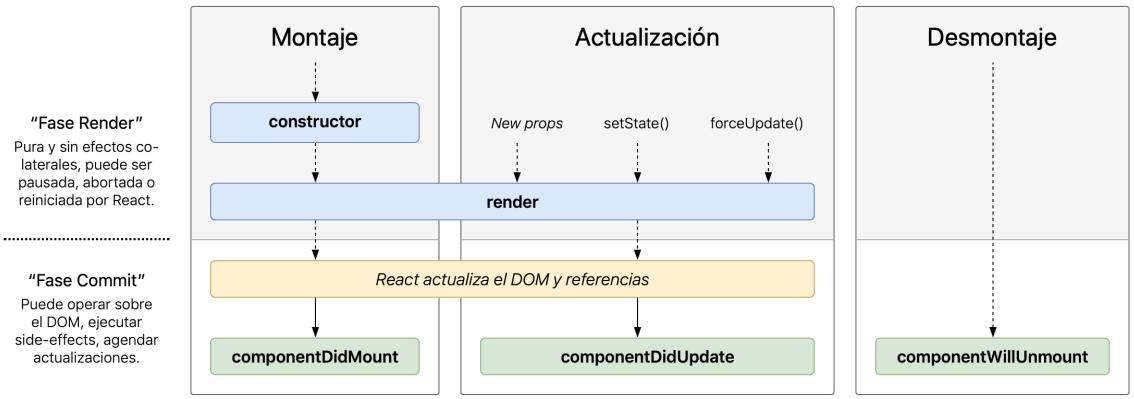
Tomen nota:

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.

1. Introducción al ciclo de vida de los componentes

Los componentes en React tienen un ciclo de vida, una creación, los procesos de render y por último una destrucción. A continuación abordaremos el ciclo de vida de cada componente.



Montaje

El montaje es el proceso por el cual el componente que creamos se incorpora en nuestra página web (DOM - Document Object Model). El proceso de montaje sigue los siguientes pasos.

1. Se llama a la función del componente
    - Se inicializa el estado (si es necesario)
    - Se ejecuta el cuerpo de la función (retornando el JSX necesario). En el cuerpo de la función no debemos actualizar el estado (porque cada llamada de cambio de estado, provoca que se vuelva a ejecutar el cuerpo de la función, produciendo un bucle infinito)
2. Se llama al useEffect (más adelante se detalla)
    - Se incluye el código que se debe ejecutar una única vez al "montar" el componente. Por ejemplo: llamada a un servidor, alguna modificación de estado que se realice al iniciarse, etc.

useEffect

Es una función especial de React la recibe como parámetro una función que será ejecutada cuando:

- El componente se cargue
- Se produzca un cambio en un state o propiedad recibida (props)

La función useEffect recibe 2 parámetros, el primero es la función que se deberá ejecutar en una determinada ocasión, y el segundo parámetro es la ocasión en la cual se debe ejecutar (Ej: cuando se carga el componente, cuando se cambia la props X, cuando se cambia el estado Y, etc)

Actualización

Existen determinados eventos que provocan que un componente se vuelva a dibujar (se ejecuta de nuevo el cuerpo de la función). Los mismos son:

- Cambio de estado de algunos de los useState utilizados en el componente
- Cambio de una propiedad que es recibida (props)
- Forzado de actualización por medio de forceUpdate()

Los pasos de la actualización de un componente son

1. Cambio del estado (llamando a un setXYZ, ej: setNombre), cambio de una o varias propiedades props, llamado a forceUpdate()
2. Se vuelve a ejecutar el cuerpo de la función
3. Se ejecutan los useEffect (en caso que existan) que estén relacionados con los cambios que se hayan realizado en las props/state. Si dentro del useEffect se vuelve a modificar un estado, se vuelve al paso 1 (cambio de estado)

Desmontaje

Cuando el componente deja de ser visualizado, el mismo es desmontado (se elimina del DOM).

2. Redefinición de comportamientos básicos

useEffect(<función>, [])

Esta función es utilizada para obtener registros de un servidor remoto o necesitemos inicializar alguna propiedad usando como referencia el espacio que ocupa el componente o algún elemento visual.

En caso de realizar un cambio en el estado, debemos llamar a la función específica (setXYZ). Esto provocará que se ejecute el cuerpo de la función/componente nuevamente.

```
React.useEffect(() => {
  // Este código se ejecuta solamente cuando se monta el
componente.
}, [])
```

useEffect(<función>, [<elemento 1>, <elemento 2>, ...])

Esta función es llamada cuando se produjo una actualización del estado (no es llamado la primera vez, en el montaje)

Puede ser utilizado para hacer peticiones de red adicionales (Ej: si se seleccionó un nuevo cliente, las direcciones del nuevo cliente)

Se utiliza para ejecutar un “efecto secundario”, alguna operación que depende de un cambio de alguno de los valores pasados al vector (elemento 1, elemento 2, etc). Es útil para los casos en los cuales debemos realizar una operación que dependa del cambio de un valor. Un caso común puede ser cuando tenemos desplegables de País y Provincia. Cuando el usuario selecciona un País, debemos hacer una petición al servidor para

consultar las provincias de ese país (en este caso, haríamos que dependa el useEffect del cambio del valor del estado país).

```
React.useEffect(() => {
  // Este código se ejecuta solamente cuando se modifica el valor del estado “pais”.
}, [pais])
```

3. Introducción a AXIOS

Cuando deseamos conectarnos con servidores remotos por medio del protocolo HTTP, es recomendable utilizar algún paquete NPM que haga más fácil la comunicación con dichos equipos. En este curso utilizaremos el paquete axios.

Para instalar axios debemos acceder por línea de comando a la carpeta del proyecto React y ejecutar

```
npm install --save axios
```

4. Conexión con el servidor

Para utilizar axios en un componente, generalmente, incorporamos el código en un useEffect(<función>, []), quedando de la siguiente forma.

```
import axios from 'axios';
...

React.useEffect(async () => {
   const respuesta = await axios.get('<url del servidor>');
}, [])
```

Primero indicamos que deseamos utilizar axios en nuestro componente.

Dentro de un useEffect realizamos la petición al servidor remoto por medio de axios.get(). La respuesta es guardada en la constante respuesta.

Axios dispone de diferentes métodos para ser llamado:

- `axios.get('url')` para hacer un HTTP GET a la URL especificada
- `axios.post('url', objJSON)` para hacer un HTTP POST a la URL especificada enviando el objJSON indicado
- `axios.put('url', objJSON)` para hacer un HTTP PUT a la URL especificada enviando el objJSON indicado (por put)
- `axios.delete('url')` para hacer un HTTP delete a la URL especificada

La respuesta de axios dispone de las siguientes propiedades principales:

- status (respuesta.status) nos permite acceder al código numérico HTTP de respuesta (Ej: 200)
- data (respuesta.data) nos permite acceder al contenido enviado por el servidor (generalmente un JSON)

5. Ejemplo

Manejo de campos de formulario

Los formularios son uno de los componentes más utilizados en las páginas web y aplicaciones, son un elemento fundamental para que el usuario pueda cargar información en un sistema. Disponemos de formularios para búsqueda, alta de información, modificación de información, etc.

En esta sección veremos cómo podemos asociar un input a un estado, lo que nos permite que el estado refleje el valor del input en todo momento. Y en caso que modifiquemos el estado, el input asociado se actualice automáticamente. Este tipo de enlace (muchas veces llamado two way data binding) es muy habitual cuando deseamos acceder a la información que ingresa el usuario desde React.

Antes de comenzar con un ejemplo, veamos las consideraciones que debe cumplir este two way binding.

- Debemos tener un estado
- Si modificamos la propiedad en el estado, se debe actualizar el input del formulario
- Si modificamos el input del formulario, se debe actualizar la propiedad en el estado

Ejemplo

Veamos un ejemplo del método render con esta etiqueta.

```
const [nombre, setNombre] = React.useState('');

const handleChange = (e) => {
   setNombre(e.target.value);
}

return (
    <div className="row">
        <input type="text" value={nombre} onChange={handleChange} />
    </div>
)
```

1. Definimos un estado para mantener el nombre
2. Asociamos al input el valor del estado `value={nombre}`
3. Al modificar el input, actualizamos el estado `onChange={handleChange}`
4. El evento handleChange() recibe como parámetro el evento que lo dispara (e), y si deseamos acceder al valor ingresado por el usuario, debemos utilizar e.target.value (el valor que ingresó el usuario en este input).

Uso de axios

```
import axios from 'axios';

...

export default function MiComponente() {
    React.useEffect(async () => {
        const respuesta = await axios.get('https://<server endpoint>');
    }, [])
}
```

Verificación del código de respuesta del servidor

```
const respuesta = await axios.get('');
...
respuesta.status
```

Acceder a los datos

```
const respuesta = await axios.get('');
...
respuesta.data
```

Manejo de la respuesta

```
    const [listado, setListado] = React.useState([])
    useEffect(async () => {
       const respuesta = await
axios.get('https://jsonplaceholder.typicode.com/comments')
       if (respuesta.status===200) {
           console.log('Respuesta OK')
           setListado(respuesta.data);
       }
    }, [])
```

6. Trabajo Práctico

Crear uno o varios componentes que se conecten a tu servidor express de Libros y que muestre el listado de todos los libros cargados en el servidor.

Bibliografía utilizada y sugerida

Node JS (n.d.) Recuperado de https://nodejs.org/es/

React JS (n.d.) Recuperado de https://es.reactjs.org/

Wikipedia (n.d.) Recuperado de https://es.wikipedia.org/wiki/React

Lo que vimos:

Ciclo de vida de los componentes y conexión con el servidor.

Lo que viene:

ReactJS Intermedio.