Contexto

Tiempo estimado: 5min

Hasta el momento, hemos presentado dos objetos donde almacenar y a través de los cuales pasar datos a los componentes: props y state. Existe un tercero: context. Vamos a verlo.

La lección comienza presentando el concepto de contexto y el acoplamiento que lleva consigo. A continuación, se muestra cómo definir propiedades en el contexto. Y finalmente, cómo usarlas.

Al finalizar la lección, el estudiante sabrá:

- Qué es el contexto.
- Para qué sirve el contexto.
- Cómo definir propiedades en el contexto.
- Cómo usar propiedades del contexto.

Introducción

El contexto (*context*) es un mecanismo a través del cual pasar datos de arriba abajo, es decir, de un componente a sus descendientes. De esta manera, se puede intercambiar datos entre un componente y sus descendientes sin necesidad de pasarlos a través de las propiedades o el estado.

El uso del contexto para intercambiar datos hace que los componentes se encuentren más acoplados. Recordemos que el acoplamiento (*coupling*) indica el nivel de interdependencia que presentan dos o más elementos del sistema. Atendiendo a este nivel de relación entre los componentes, se distingue entre acoplamiento débil y acoplamiento fuerte.

Se produce acoplamiento débil (*loose coupling*), cuando los componentes se encuentran poco relacionados o no necesitan saber mucho de las APIs de los otros. En cambio, se produce acoplamiento fuerte (*high coupling*) cuando los componentes se encuentran muy relacionados entre sí.

Cuanto menos acoplados se encuentren los componentes, mejor. Más fácil será su mantenimiento y hacer cambios. En cambio, cuanto más acoplados se encuentren, peor. Más difícil será su mantenimiento y la realización de cambios, inevitables en todo sistema de software.

Por esta razón, se recomienda utilizar el contexto con mucha cautela. Esto no quiere decir que no se use. Simplemente que hay que hacerlo con cuidado y teniendo en cuenta posibles problemas de mantenimiento que se podrían producir si acabamos haciendo que la aplicación React se encuentre fuertemente acoplada.

Es importante tener claro que el contexto se utiliza de arriba abajo. De un componente a sus subcomponentes directos o indirectos. Los datos se definen en un componente superior y se pueden acceder en cualquiera de sus descendientes. Pero ojo, son de sólo lectura, al igual que las propiedades.

La ventaja que tiene el uso del contexto es que no hace falta definir ninguna propiedad explícita en el elemento instanciador de los subcomponentes. Se define en el contexto y, entonces, los subcomponentes acceden a él a través del atributo context del componente.

Definición de contexto

Todo componente puede añadir propiedades al contexto heredado de su padre o, si es la raíz del contexto, definirlo. Ambas cosas se hacen igual. La cuestión es que estas propiedades serán accesibles por todos sus subcomponentes. Para ello, no tiene más que definir el método getChildContext(), el cual debe devolver un objeto con las propiedades que desea añadir al contexto:

```
getChildContext() : object
```

Por otra parte, hay que definir la propiedad estática childContextTypes, que no tiene otro objetivo que indicar los tipos de las propiedades del contexto, de manera similar a propTypes.

Veamos un ejemplo ilustrativo:

```
class MiForm extends React.Component {
  …

  /**
   * Devuelve las propiedades que el componente pone a disposición de sus
   * subcomponentes mediante el contexto.
   *
   * @return object
   */
  getChildContext() {
    return {
      form: this
    };
  }

  /**
   * Tipos de las propiedades del contexto.
   *
   * @type object
   */
  static get childContextTypes() {
    return {
      form: React.PropTypes.element
    };
  }

  …
}
```

Para definir un contexto, es muy importante definir ambos miembros, tanto el método getChildContext() como la propiedad estática childContextTypes.

Utilización de las propiedades del contexto

Si un componente necesita utilizar propiedades del contexto, no puede hacerlo sin más. Por un lado, es necesario definir la propiedad estática contextTypes, mediante la cual indica qué propiedades del contexto desea acceder. Se define de manera similar a childContextTypes y propTypes. Consiste en un objeto que tiene una propiedad para cada una de las propiedades del contexto a la que desea acceder, indicando su tipo.

Veamos un ejemplo ilustrativo:

```
static get contextTypes() {
  return {
    form: React.PropTypes.element
  };
}
```

Una vez configurada la propiedad contextTypes, el componente puede acceder a las propiedades del contexto mediante su atributo context. Ejemplo:

```
this.context.form
```

Generador de Justo

Cuando se crea un componente mediante el generador de Justo, recordemos, mediante su comando component, nos preguntará, por una parte, si deseamos definir contexto y, por otra parte, si usarlo.

Según el caso, no hay más que indicar *sí* o *no*.

