Гриценко Екатерина Михайловна
канд. техн. наук, доцент, заведующая кафедрой
ФГБОУ ВО «Сибирский государственный
технологический университет»
г. Красноярск, Красноярский край
Бракк Даниил Олегович
магистрант
ФГБОУ ВО «Сибирский государственный
аэрокосмический университет им. академика М.Ф. Решетнева»
г. Красноярск, Красноярский край

ПРИМЕНЕНИЕ КОНЕЧНОГО АВТОМАТА ПРИ ПРОЕКТИРОВАНИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА ПРИМЕРЕ ПРОЕКТА «SCONTO»

***Аннотация**: в статье освещен метод проектирования пользовательского интерфейса проекта «SCONTO», основанный на конечном автомате, а также приводится один из технологических инструментов, позволяющий манипулировать состояниями пользовательского интерфейса.*

***Ключевые слова**: пользовательский интерфейс, проектирование.*

Введение

На данный момент такая область профессиональной деятельности, как проектирование пользовательских интерфейсов, пополнилась огромным сообществом и невероятным количеством незаурядных методов и инструментов. Прошло порядка 7 лет с момента первых полноценных публикаций на тему проектирования интерфейсов в 2009 году от Google и Apple. Это значительный промежуток, чтобы пересмотреть и подвергнуть исследованию привычные и проверенные методы, начать использовать то, что более эффективно будет решать повседневные задачи как для бизнеса, так и для разработчиков.

Конечный автомат и состояния

Ключевым понятием, которым оперирует конечный автомат является состояние. Под состоянием можно понимать любые вещи, которые могут быть зафиксированы в определенный момент времени. В теории конечных автоматов состояние может быть изменено на другое с помощью, так называемых, операций.

В сфере проектирования интерфейсов факторами смены состояний являются события. События – это реакции на те или иные действия в системе: пользовательский ввод (мышь, клавиатура, сенсор, голограмма), сигналы операционной системы и т. д.

В момент запуска системы она находится в каком-либо начальном состоянии. При возникновении любого события, система меняет свое состояние на то, которое предусмотрено разработчиком. В свою очередь, на данное изменение реагирует либо весь пользовательский интерфейс, либо отдельные его части. При этом, для системы не имеет значение, каким событием вызвано изменение состояния. Например, пользователь мог вызвать клик по кнопке, ввести определенную строку в текстовое поле или в системе появились новые данные с сервера. Для системы не важно, каков источник события. Если произошло событие, система должна среагировать. При этом разработчик сам реализует для системы такое понятие, как событие. Как правило, ими могут выступать обычные текстовые или числовые флаги.

Техническая реализация

Проект «SCONTO» включает в себя несколько компонентов взаимодействия с человеком, в том числе и через браузер. В браузере, как известно, единственным языком программирования логики является JavaScript. Любой проект, который включает в себя интерфейсную часть пользователя, в момент старта имеет список предопределенных технологий и методов, с которыми будут работать разработчики в команде.

В «SCONTO» ключевым методом проектирования стал компонентный подход. При этом, каждый компонент интерфейса имеет свои свойства, который могут быть изменены извне другим компонентом, и внутреннее состояние, измене-

ние которого может влиять на изменение конечного вида отображения. Компоненты могут вкладываться друг в друга, образуя иерархическое дерево. Компонент верхнего уровня может влиять на компонент более нижнего. Для этого новые данные передаются в качестве свойств в нижний компонент. Компонент нижнего уровня может отлавливать любые изменения его свойств и либо менять свое внутреннее состояние, либо нет. Любое изменение состояния компонента вызывает изменение его конечного отображения: цвета его блоков, вида кнопок, текста заголовков, количества вложенных других компонентов и т. д. В качестве библиотеки, позволяющей проектировать пользовательский интерфейс в браузере с помощью компонентов, выступает React.js.

Однако, количество компонентов интерфейса, которые реагируют на пользовательские действия, может превышать более 1000 элементов. Для интерфейсов, реализующих сложные интерактивные взаимодействия с пользователем, данный факт является в порядке вещей. Однако, даже имея внутренние состояния каждого компонента, контролировать каждое состояние локально является сложной задачей. Приемлемым решением является использовать, помимо локальных состояний для каждого компонента, общее глобальное состояние всей системы пользовательского интерфейса, а также определиться с методами (или событиями, или действиями), которые вызывают изменения.

Здесь, в качестве методов, как правило, выступают обычные функции, которые возвращают новое состояние системы. При этом, новое состояние может наследовать предыдущее.

Любое новое состояние может быть доступно в любом из компонентов интерфейса по усмотрению разработчика. Данные передаются через цепочку свойств от компонентов верхнего уровня к нижним. Удачным решением в выборе библиотеки для работы с глобальным состоянием и компонентами React.js может выступать Redux. С точки зрения разработчика, вся цепочка логики изменений состояний может быть реализована в виде чистых функций JavaScript и любых видов ветвлений. При этом то, что является событием в системе, разра-

ботчик решает сам: текстовые, числовые флаги или комплексные объекты. Библиотека лишь реагирует на изменения состояний. И если это так, то Redux «пробрасывает» новые данные по цепочке компонентов к тому, которое использует глобальное состояние напрямую, а не через свойства верхних по иерархии компонентов. В качестве состояний обычно выступают объекты JavaScript без каких-либо методов. Разработчик сам выбирает то, какая структура будет у состояний.

Заключение

В данной статье был освещен метод проектирования пользовательских интерфейсов с помощью конечного автомата. Также, была представлена техническая реализация описанного метода в браузере на примере проекта «SCONTO».

Список литературы

1. ХабраХабр, крупнейшие блог-посты в сфера IT в рунете // HabraHabr. – 2016 [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/

2. React, A JavaScript library for building user interfaces // Facebook Inc., 2014–2016 [Электронный ресурс]. – Режим доступа: http://facebook.github.io/react/

3. Redux, Redux is a predictable state container for JavaScript apps., 2015–2016 [Электронный ресурс]. – Режим доступа: http://redux.js.org/