**Сведения об авторе:**

**ФИО:** Кузменков Анатолий Сергеевич

**Академическое звание:** Магистр технических наук

**Наименование учебного заведения:** Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (Университет ИТМО)

**Адрес электронной почты:** njkz-09@yandex.by

**Контактный телефон:** +375298526443

**Рубрика:** Технические науки

**Количество журналов:** 0

**Компонентный подход разработки приложений во *front-end*.**

За несколько последних десятилетий человечество проделало огромный путь в области компьютерных технологий в целом, и веб-технологиий в частности. За эти годы многократно увеличилась производительная мощность техники, а так же пропускная способность каналов связи. Веб-технологии затронули большинство сфер деятельности современного общества. Так веб-технологии получили широкое распространение в сферах электронного бизнеса, средствах массовой информации, литература, музыка, кино, связь и общение [1].

Однако при все нарастающих темпах использования всемирной паутины, появлялись новые требования от разработчиков в плане оптимизации процесса разработки и требования к производительности. Не смотря на развитие новых подходов и технологий основополагающими во front-end по прежнему являются *HTML*, *CSS* и *JavaScript*.

В плане организации работы современные реалии заключается в необходимости разработки приложений в рамках команд разбросанных по всему миру. Такая ситуация приводит к необходимости создания приложений с максимально простым для понимания синтаксисом, а также требует совершенствования инструментов разработчика.

И так современные реалии приводят к необходимости организации всего приложения или больших частей в виде универсальных модулей и компонент с максимально понятным синтаксисом. И дальнейшее их использование в большом количестве проектов.

Справиться с нарастающими требованиями может помочь компонентно-ориентированное программирование. Данный подход позволяет строить приложение как набор взаимодействующих друг с другом компонент. При этом компонент представляет собой независимый модуль программы, предназначенный для повторного использования и развертывания, реализованный в виде множества языковых конструкций, объединённых по схожим признакам [45].

Основным преимуществом подхода является легкость модернизации приложения путем замены одного компонента другим. Такой эффект достигается благодаря использованию черного ящика (Рисунок 1), то есть элемент про-граммы проектируется и реализуется таким образом, что бы иметь определенный типовой интерфейс при этом совершенно не важна внутренняя реализация.

Выходные   
данные

Черный ящик

Входные   
данные

Рисунок 1 – Схема работы организации работы программного компонента  
 «Черный ящик»

Для веб технологий так же характерно разделение компонентов на логические и стилизующие, что в значительной степени облегчает дальнейшую работу с кодом программы, так как позволяет беспрепятственно подменять один компонент другим.

Наиболее перспективной реализацией компонентного подхода во front-end разработке приложений является разработанная Facebook библиотека react.js. Тем не менее на момент своего появления в 2013 году предложенная концепция была принята весьма скептически так как имела уникальный подход [52, 53] и отбрасывала некоторые общепринятые концепции. По существу разработчики откинули опыт сообщества и создали библиотеку, позволяющую получить максимум от использования модульной структуры приложения.

Основным преимуществом данной библиотеки являлось возможность со-здания веб-интерфейсов быстро, с использованием читаемого и понятного кода, что во многом обеспечило большой приток свежей крови. А так же серьёзного расширения возможностей стандартной библиотеки путём использования сто-ронних библиотек, построенных на react.js. Если осуществить поиск на github или с помощью npm, можно найти большое количество решений типовых за-дач.

Следующей ключевой особенностью языка является возможность исполь-зования языковых конструкций при динамическом создании html-узлов в при-ближенном к HTML виде с использованием так называемого JSX (JavaScript Syntax eXtension), расширения стандартного JS для динамического создания элементов. Что значительно сокращает и упрощает понимание синтаксиса ком-понента, увеличивая тем самым эффективность создания и поддержки кода.

Следующим преимуществом является высокая скорость перестроения DOM, благодаря внутренней оптимизации. Это достигается путем создания так называемой виртуальной DOM-модели, которая накапливает изменения и, в дальнейшем, вносит сразу все накопленные правки. Как известно работа с DOM является одной из наиболее затратных процедур, и оптимизация позво-ляет в значительной степени ускорить работу front-end [52, 53]. При этом пере-стройка происходит при изменение данных state (внутреннее состояние компо-нента) и props (внешнее состояние). Вовремя своей работы компоненты прохо-дит через ряд этапов жизненного цикла. На каждом этапе вызывается опреде-ленная функция, переопределив которую можно задать собственное поведение. При этом функционал можно разделить на три основных части: создание, из-менение и удаление.

**Библиографический список:**

1. Lindley, C. JavaScript Enlightenment. O’Reilly Media, 2012. 166 p.

2. JavaScript.ru. Полифилы. URL: https://learn.javascript.ru/dom-polyfill (Дата доступа: 30.05.2017).

3. DailyJS - JavaScript Blog. Building A JavaScript Framework. URL: http://dailyjs.com (Дата доступа: 30.05.2017)

4. Бенедетти, Р., Крэнли Р. Изучаем работу с jQuery. Спб.: Питер, 2012. 508 с.

5. Garcia, J. Ext JS in Action. Manning Publication, 2011. 495 p.