**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 [2, 3] 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 технологии является чистый и интуитивно понятный код. А так же большое количество сторонних расширений, которые возможно использовать для решения типовых задач. Для динамического создания *DOM*-дерева используется *JSX* (*JavaScript Syntax eXtension*), расширения стандартного *JavaScript*, что позволяет использовать синтаксис приближенный к *HTML*.

Так же важной составляющей является поддержка так называемой виртуальной *DOM*-модели, что позволяет оптимизировать затраты на динамическое создание элементов веб-страницы. Естественно поддержание виртуальной *DOM*-модели обходится дополнительными затратами ресурсов. Однако правильное использование жизненного цикла (Рисунок 2) компонента позволяет существенно оптимизировать данный процесс [4].

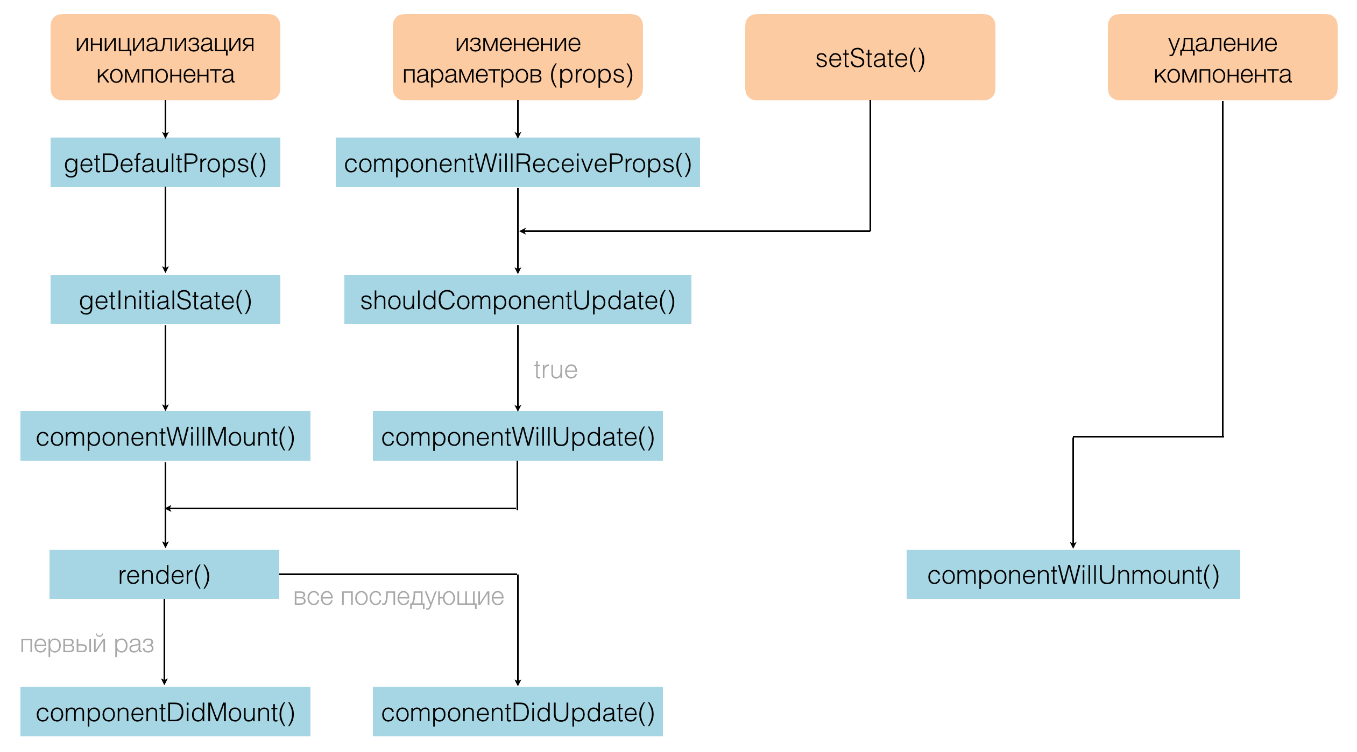


Рисунок 2 – жизненный цикл компонента *react.js*

При разработке компоненты можно разделить на несколько типов.

1. Простые компоненты. Данные компоненты служат для вывода ин-формации а так же стилизации типовых элементов, таких как заголовки (*title*) параграфы(*p*) и др. В общем виде работу таких компонентов можно представить, как черный ящик получающий данные на входе, генерирующий *dom*-дерево внутри, и возвращающий корневой   
элемент (Рисунок 3).

**Компонент**

**Результат**

**Данные**

Рисунок 3 – Схема работы простых компонентов

2. Компоненты обертки. Данные компоненты представляют собой сво-его рода обёртки над содержимым, содержащие общие компоненты для не-скольких вариантов содержимого (Рисунок 4).

**Компонент**

**Данные**

**Результат**

**Внутренние компоненты**

**Контент**

Рисунок 4 – Схема компонентов обёрток

3. Умные компоненты. Данные компоненты работают производят предварительную обработку данных изменения, способны динамически изменять свое состояние и осуществлять обмен данными с сервером.

Подытоживая вышесказанное можно сделать вывод что компонентный подход, при правильном использовании позволяет ускорить не только процесс разработки, но и дальнейшей модернизации кода.

**Библиографический список:**

1. Кузменков А.С., Причины возникновения и развития JS-фреймворков, преимущества, достоинства и недостатки. Научный обозреватель. 2017. №6. С. 49 – 52.
2. Banks, A., Е. Porcello. Learning React. O’Reilly Media, 2017. – 350 p.
3. Babel. Try it out URL: https://babeljs.io/repl/ (Дата доступа: 24.03.2018).
4. Metanit. Жизненный цикл компонента URL: https://metanit.com/web/react/2.6.php (Дата доступа: 24.03.2018).