*«N. B.: данную анкету я отправлял чуть менее года назад, поэтому, если где-то Вы встретите ответы, схожие с прошлыми – это значит, что я так считаю до сих пор, а не то, что я невнимательно отнёсся к её написанию.»*

***Q1. Расскажите, чем, на ваш взгляд, отличается хорошее клиентское приложение от плохого с точки зрения:***

1. Хорошее клиентское приложение:
   1. Пользователь. Интуитивно-понятное взаимодействие с приложением, удобочитаемость или доступность (в том числе для пользователей с ограниченными возможностями), быстродействие, привлекательность, информативность, при просмотре с карманных устройств – отсутствие перегруза батареи.
   2. Менеджер проекта. Соответствие требованиям заказчика (дизайн, ТЗ), чётко составленная документация по проекту, возможность в короткие сроки и с наименьшими затратами вносить изменения, исправлять ошибки.
   3. Дизайнер. Мне кажется, дизайнерам не особо важно, как будет реализовано приложение =) может быть, чтобы оно было выполнено как можно точнее по макету для более чёткого понимания каких-либо недочётов с возможностью в дальнейшем их безболезненно исправить.
   4. Верстальщик. Легкочитаемый, семантичный код, наличие грамотного описания приложения в части, касающейся вёрстки и её взаимодействия с этапами разработки. Чёткое разделение сфер ответственности на этапах разработки и поддержки приложения. Отсутствие на проекте перегруженных стилевых фреймворков, чьи компоненты сложно переопределять.
   5. Серверный программист. Сведение к минимуму необходимости вмешиваться в html/css/js – части приложения. Минимальная нагрузка приложения на сервер. Грамотно налаженный функционал на клиентской части, принимающий данные с сервера.
2. Плохое клиентское приложение:

Которое не отвечает вышеперечисленным пунктам =)

***Q2. Опишите основные особенности разработки крупных многостраничных сайтов, функциональность которых может меняться в процессе реализации и поддержки.***

По моему мнению, разработкой подобных сайтов должна заниматься полноценная квалифицированная команда, хотя, с другой стороны, это является и отличной возможностью получения опыта для начинающих специалистов. Важно уделять внимание каждому этапу разработки. В составлении ТЗ должны принимать участие, помимо менеджера проекта, специалисты по дизайну и разработке (возможно и от серверной части и от клиентской), это позволит более чётко определить временные и трудозатраты, более оперативно вносить изменения в ТЗ, составлять техническую документацию для этапов разработки, расставлять задачи и взаимодействовать в команде.

Также я считаю, что, при делении разработки на фронт и бэк, на каждой стороне должен быть сотрудник, который в достаточной степени владеет знаниями и опытом в соответствующей сфере. Это позволит выполнять задачи более согласованно и качественно, с учётом современных требований к разработке, что, в свою очередь, приведёт к минимизации проблем, которые могут возникнуть в процессе реализации и поддержки сайта в дальнейшем.

***Расскажите о своем опыте работы над подобными сайтами: какие подходы, инструменты и технологии вы применяли на практике, с какими проблемами сталкивались и как их решали.***

В своей работе я успел столкнуться со следующими фреймворками/ библиотеками: Vue (2%), React (95%) и Angular (3%). На проектах с React и Angular на клиентской стороне основным языком является Typescript. В качестве вспомогательных стилевых фреймворков используются Blueprint.js, Angular Material. На проекте с React применяется Mobx для управления состоянием приложения и Nodejs в качестве серверного языка. На проекте с Angular – C#. В клиентских частях приложений проектов применяются различные компонентные подходы для создания интерфейсов.

Основной проблемой при работе было взаимодействие с дизайнерами, которая заключалась в использовании различных приложений для редактирования, просмотра макетов (у дизайнеров – Sketch, у разработчиков - Photoshop). Проблема была решена включением Zeplin в инструментарий с обеих сторон.

***Q3. При разработке интерфейсов с использованием компонентной архитектуры часто используются термины Presentational Components и Container Components. Что означают данные термины? Зачем нужно такое разделение, какие у него есть плюсы и минусы?***

Presentational Component – компонент, предназначенный для отображения данных. Он содержит в себе по большей части только вёрстку, иногда в нём может храниться состояние для конкретных элементов внутри компонента.

Container Component – компонент, описывающий логику (в том числе работу с состоянием приложения и http-запросами), которая в качестве параметров передаётся в Presentational Component.

Данный подход позволяет упорядочить работу с компонентами, повысить их читабельность, упростить тестирование, а в случае с Presentational Component – даёт возможность безболезненного переиспользования компонентов.

На мой взгляд, плюсы данного подхода очевидны. Поскольку в команде он используется в малой степени, его минусов я разглядеть не смог. Может быть, в дальнейшем он мне в чём-то и покажется неудобным, но это, как говорится, не точно.

***Q4. Как устроено наследование в JS? Расскажите о своем опыте реализации JS-наследования без использования фреймворков.***

Принцип наследования в js работает с объектами. Он заключается в возможности объекта B наследовать свойства объекта A (объект A выступает в роли прототипа объекта B). У объекта-прототипа так же может быть прототип и эта иерархическая цепочка заканчивается (начинается =) ) глобальным js-объектом Object, чей прототип является null. У Object уже есть набор свойств и методов, которые будут наследоваться всем создаваемым объектам. Таким образом, наследование реализуется по умолчанию каждый раз при создании объектов (массивов, функций – так как их прототипом так же является Object). Все вновь создаваемые объекты наследуют свойства их прототипа - Object. Но также можно создавать собственные прототипы. Например, когда нам необходимо использовать одни и те же свойства у разных объектов. Для этого создаётся объект-прототип (конструктор), в который записываются необходимые свойства, которые потом можно «раздать» новым объектам.

Без использования фреймворков мы реализуем js-наследование каждый раз, когда создаём новый объект, массив или функцию =)

В работе наследование используется редко. В основном на этапе инициализации – для создания различных типов хранилищ состояния приложения. Также наследование происходит при создании React-компонента (React же не фреймворк =) )

***Q5. Какие библиотеки можно использовать для написания тестов end-to-end во фронтенде? Расскажите о своем опыте тестирования веб-приложений.***

К сожалению, в работе тестированию клиентской части не уделяется должное внимание, поэтому мой опыт ограничивается прохождением интенсива «JavaScript, уровень 2» в HTML-Академии и решением тестового задания для FunBox.

На интенсиве использовались Mocha и Chai. При решении задания я использовал Enzyme и Jest.

***Q6. Вам нужно реализовать форму для отправки данных на сервер, состоящую из нескольких шагов. В вашем распоряжении дизайн формы и статичная верстка, в которой не показано, как форма должна работать в динамике. Подробного описания, как должны вести себя различные поля в зависимости от действий пользователя, в требованиях к проекту нет. Ваши действия?***

В данной ситуации я сперва обговорю данный момент с менеджером проекта. После его предложений – с тимлидом. На основании совместного опыта построю модель поведения формы, предложу её менеджеру. И так до момента, пока вопросов не останется. Затем приступлю к работе.

Возможно, в зависимости от сложности формы, реализую форму самостоятельно, после чего предоставлю результат менеджеру и тимлиду для проверки.

***Q7. Расскажите, какие инструменты помогают вам экономить время в процессе написания, проверки и отладки кода.***

Для написания кода я использую Visual Studio Code. В целях сохранения единого стиля кода на проекте используется tslint (в тестовом задании я настроил eslint) и плагины для редактора для автоматического исправления ошибок. Проверку кода удобно проводить тестированием (по причине, описанной в пункте Q5, на работе как таковой проверки кода не происходит – только на этапе сборки проекта, если падают ошибки). Отладку кода я провожу логгированием через терминал или консоль разработчика в браузере.

***Q8. Какие ресурсы вы используете для развития в профессиональной сфере? Приведите несколько конкретных примеров (сайты, блоги и так далее).***

***Какие ещё области знаний, кроме тех, что непосредственно относятся к работе, вам интересны?***

Основными источниками информации являются официальные документации библиотек, фреймворков, технологий. По возможности выполняю задачи на [www.codewars.com](http://www.codewars.com), и прохожу курсы на [www.codecademy.com](http://www.codecademy.com), в том числе и по таким языкам как Java, Python, Ruby для большего понимания принципов ООП.

Слежу за деятельностью HTML-Академии. Планирую, при необходимости, проходить полноценные интенсивы для повышения квалификации.

Из блогов и проч.: портал <https://habr.com/ru>, youtube-каналы: [Traversy Media](https://www.youtube.com/channel/UC29ju8bIPH5as8OGnQzwJyA), [ITVDN](https://www.youtube.com/channel/UCzxRv9BtqrM946JmaMLtv_w), [Max Frontend](https://www.youtube.com/channel/UCqJyAVWwIqPWKEkfCSP1y4Q) – помог с тестами, [teach-in](https://www.youtube.com/channel/UCimGOl6wSQWDbBFoVMQGzMg) – отличный канал с лекциями преподавателей МГУ и многие другие, которые попадаются по необходимости.

Также слежу за блогом [TaleWorlds](https://twitter.com/taleworlds). Хотя это и игровая индустрия, но его прочтение вдохновляет меня на разработку.

Кроме разработки - увлекаюсь историей (больше - отечественной). Как хобби и саморазвитие – музыка (меломан, но главное предпочтение – рок, играю на гитаре, нередко с братом на студиях) и литература (отечественная классика).

***Q9. Расскажите нам немного о себе и предоставьте несколько ссылок на последние работы, выполненные вами.***

В школе учился отлично до 9 класса, потом учился хорошо. Поступил в МАИ и РЭА Плеханова, выбрал последнее. В университете работал, играл в КВН. После учёбы отслужил в армии. Потом нелёгкая занесла в РЖД, где прозябал до начала 2018 года. С вёрсткой познакомился в 8 классе в Детском доме творчества на Полянке. Первый сайт был посвящён игре Commandos, который, к сожалению, не сохранился. В университете работал с контентом сайта вуза. Около года назад устроился фронтенд-разработчиком в компанию Программный Продукт, работаю там по сей день.

Общительный, весёлый, дружелюбный, неконфликтный =)

Поскольку проекты, в которых я принимаю/принимал участие являются либо внутренними порталами, либо ещё не отправленными в релиз, ссылки на них предоставить, к сожалению, не могу.

Из последних наработок:

[Шаблон для сборки на webpack](https://github.com/dimitriskot/webpack-template)

[Шаблон для разработки на Js с React](https://github.com/dimitriskot/react-native-template) (слово «native» затесалось случайно, это не React Native)

[Шаблон для разработки на Typescript с React](https://github.com/dimitriskot/react-typescript-template)

[Шаблон для разработки на Coffeescript с React](https://github.com/dimitriskot/react-coffeescript-template)

[Тестовое задание на Js с Vue](https://github.com/dimitriskot/XDTest) (при трудоустройстве в Программный Продукт)

Крайний учебный проект по Js в HTML-Академии:

[Угадай мелодию](https://github.com/dimitriskot/425695-guess-melody)

Спасибо за уделённое внимание моему отклику, очень надеюсь на понимание и дальнейшее сотрудничество!