**Q1 Отличия хорошего клиентского приложения от плохого с точки зрения:**

**а) Пользователя.**

Для пользователя основными критериями хорошего web-приложения является логическая понятность и удобство использования приложения. Визуальная привлекательность интернет-ресурса тоже немаловажна, но не должна противоречить логики сайта. Например, пользователям, заходящим на информационный портал совершенно необязательно видеть параллакс-эффект, т.к. он не только отвлекает от важных новостей, но и увеличивает время обращения с сайтом, что неприемлемо для информационных порталов. Затронув такой фактор как время обращения с web-приложением хотелось дополнить эту тему тем, что многие web-разработчики в угоду красивой визуальной обертки создают очень медленные сайты. Для пользователей с небыстрым интернетом или медленным компьютером этот аспект крайне важен, и медленный сайт может навсегда отпугнуть подобных пользователей.

**б) Менеджера проекта.**

Являясь прямым посредником между заказчиками и разработчиками web-приложения менеджеру проекта важно видеть сайт не только как продавцу готового изделия, но и как его покупателю. Тем самым, для менеджера проектов важна как репрезентативность клиентского приложения, так его логическая простота и ясность для дальнейшей работы разработчиков с этим приложением.

**в) Дизайнера.**

Смотря через призму взгляда дизайнера, хороший сайт — это не просто красивая обертка, это web-приложение, полностью отвечающее своим идейным представлением и максимально дающее требуемый результат.

**г) Верстальщика.**

Зависимость верстальщика от дизайнера останавливает критерии выбора хорошего клиентского приложения, как имеющего гибкий дизайн, упрощающий процесс адаптивной и кроссбраузерной верстки, а также максимально беспроблемной его посадки под CMS.

**д) Серверного программиста.**

Для серверного программиста крайне важна работа web-приложения как целого организма, имеющего четкую организацию его жизнедеятельности, а значит и понятную логическую составляющую его поведения. Для хорошего web-приложения важно понимать поведение этого организма на длительный срок с дальнейшей возможностью доработки определенных структур его жизнедеятельности.

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

Любой крупный проект, необязательно в области сайтостроения, требует детальной проработки технического задания. На этом этапе важно определиться с концепцией сайта, его потребностями и целями. Проработав все детально, необходимо произвести оценку трудозатрат на создание сайта, а также сроков его выпуска. В зависимости от этой оценки создать команду исполнителей и распределить роли в ней с учетом всех этапов разработки web-приложения. После этого можно непосредственно приступать к разработке сайта, обеспечивая максимальный обмен информацией между разработчиками текущего проекта. В крупных проектах большую роль играет тестирование и seo-оптимизация web-приложения, желание сэкономить на специалистах «универсалах» может пагубно сказаться на развитие проекта в дальнейшем. После выполнения всех этапов разработки следует еще раз вернуться к техническому заданию и максимально пристально проверить на соответствие требованиям и целям заказчиков к сайту. Уточнить еще раз детали, и лишь потом производить дальнейшие финальные обновления проекта.

**Q3 О Presentational Сomponents и Сontainer Сomponents.**

К сожалению, ранее не был знаком с разделением компонентов на Presentational Сomponents и Сontainer Сomponents. Дублировать информацию из просторов интернета без какого-либо практического навыка в этой области считаю бессмысленным.

**Q4 Наследование JS.**

Использовал наследование JS только при обучении на практических работах.

А если говорить про наследование, то изначально (до ES6) наследование осуществлялось через свойство protototype, которому присваивалась ссылка на новый объект, созданного из protototype базового объекта.

Однако, в ES6 упростили эти процедуры, и появилась возможность наследования с помощью расширения (extends) классов, где после указания расширения достаточно переопределить необходимые методы и свойства базового класса и унаследовать все остальные методы и свойства. Это упрощение («синтаксический сахар») сделало более понятное восприятие кода, написанного на JS.

**Q5 Библиотеки для написания тестов end-to-end во фронтенде.**

Не имею опыта в данном направлении.

**Q6 Форма для отправки данных на сервер.**

Для валидации формы использовал бы инструменты JS. Для обязательных полей сделал бы проверку на пустую строку, для электронной почты проверку на формат записи. Если есть телефон проверка на ввод цифр, а не других символов. При соблюдении условий отправил бы на отдельный php-скрипт сервера, где обработал текстовые данные через strip\_tags и trim, а числовые через abs и int. Ну а дальше зависит от самого сайта, если это какой-то крупный форум, то обработанные данные надо будет отправить в базу данных, если какой-то небольшой сайт, то можно сохранить в обычном текстовом файле.

**Q7 Инструменты, помогающие экономить время в процессе написания, проверки и отладки кода.**

Из текстовых редакторов использую Sublime Text, из IDE – Visual Studio. В качестве таск-менеджера пользуюсь Gulp, при необходимости применения php, Gulp использую вместе c OpenServer. В зависимости от задач использую сетку Bootstrap. Редко использую jQuery (больше люблю нативный JS).

**Q8 Ресурсы для развития в профессиональной сфере.**

Ну, во-первых, использую исходники (документацию с официальных сайтов). Периодически захожу на сайт [w3schools.com](https://www.w3schools.com). Просматриваю ответы на toster.ru. Смотрю уроки на ютубе.

**Q9 Обо мне**

Вся информация обо мне на моем личном сайте toshlyandiya.ru