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

****

**- пользователя;**

**- менеджера проекта;**

**- дизайнера;**

**- верстальщика;**

**- серверного программиста.**

Со стороны пользователя приложение должно быть с удобным интерфейсом, лёгкой навигацией, приятным дизайном, быстро работать(пользователь не хочет ждать долгую загрузку), он может легко найти нужную информацию или услугу. Плохое приложение обладает противоположным свойствами, раздражает пользователя, ему не приятно проводить время в нем.

Со стороны менеджера проекта хорошее приложение решает все задачи поставленные руководством, оно нравится пользователю, его легко поддерживать существующей команде, оно приносит прибыль для компании.

Со стороны дизайнера выполняет задачи поставленные проект менеджером, помогает пользователю в выполнении своих задач, обладает дизайном вызывающем интерес не только пользователей, но и коллег, его легко реализовывать разработчикам.

Со стороны верстальщика приложение должно обладать валидной версткой, точное соответствовать техническому заданию, хорошо адаптироваться к разным устройствам, обладать доступностью, программная часть структурированным, хорошо читаемым кодом кодом.

Со стороны серверного программиста хорошее приложение должно быть безопасным, проводить корректные валидацию данных, корректно обрабатывать запросы на сервер, отдавать данные.

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

У меня не было такого опыта, но я думаю в крупных приложениях в первую очередь нужно продуманная архитектура, чтобы можно было расширять функциональность, тестируемость кода, разделение на модулию.

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

Presentational Сomponents — это компонент обычно ответственный визуальную часть(View), он не обрабатывает данные, у него нет зависимотей от других компонентов, просто рендерит html, получает данные и преобразует их в визуальную часть приложения, в него могут входить другие Presentational Сomponents и Сontainer Сomponents, в react получает данные через props, при использовании новых хуков может иметь свой state и управлять lifecycle.

Сontainer Сomponent — это контейнер ответственный за внутреннюю обработку данных, которые в дальнейшем используются в компоненте, может свои Dom элементы и стили него могут входить другие Presentational Сomponents и Сontainer Сomponents, в react использует в виде класса для удобства обработки данных.

Данное разделение удобно для разделения логики view и обработки данных, изоляции определённой логики и решении определённой узкоспециализированной задачи, многократное использование, уменьшается повторение кода.

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

В js используется наследование с помощью прототипов, у каждого объекта есть свойство prototype, которое содержит все доступные методы и свойства. Объекты используют prototype, чтобы наследовать свойства и методы. Движок js ищет свойства или методы в самом объекте, потом в prototype, потом в других объекты которые были наследованы по той же цепочке. Таким образом мы можем создать объект и наследовать его свойства и методы цепочкой:

function Class(){};

Class.prototype.method = function(){

//code...

}

const class = new Class();

class.method();

Я писал небольшой редактор изображений на чистом js с использованием классов и наследования, но уже с ES6 синтаксисом.

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

Чаще всего используются mocha.js, jest.js, но конечно их больше. Я писал только на jest. Тестировал node.js запросы на jest.

**7.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?**

Попрошу уточнить тз у проект менеджера, в это время можно начать реализовать основной функционал формы, передачи данных на сервер.

**8.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?**

Читаю статьи на medium, подкасты Вебстандарты, обучающие курсы на youtube(Traversy media, лекции: Timur Shemsedinov, блог Вадима Макеева), записи конференций. Занимаюсь английским языком.

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

Я начинающий фронтенда разработчик(js, react, redux, html, css), знаю основы node.js, но личных проектов пока на писал на node.

Примеры работ:

1.Weather app: <https://dindeni.github.io/weather> - react, redux  
 Исходники: <https://github.com/dindeni/weather>  
 2.Simple image editor: <https://dindeni.github.io/editor> - vanilla JS  
 Исходники: <https://bitbucket.org/dimdeni/imageeditor>