Ответы на вопросы к квалификационному заданию для разработчиков Javascript.

Q1. Расскажите, чем, на ваш взгляд, отличается хорошее клиентское приложение от плохого с точки зрения пользователя, менеджера проекта, дизайнера, верстальщика и серверного программиста.

Хорошее клиентское приложение с точки зрения

1. Пользователя:
   - Имеет интуитивно-понятный интерфейс.
   - Корректно отображается на любом устройстве.
   - Загружается быстро, без видимого построения элементов.
   - Нет зависаний/фризов при взаимодейтсвии со страницей.
   - Идентично или схоже отображается в различных браузерах.

2. Менеджера проекта:
   - Соответствует заданному ТЗ или макету.
   - Имеет возможно быстрого внесения правок.
   - Имеет положительный отзыв у заказчика.
   - Не создает сложностей другим участникам разработки.

3. Дизайнера:
   - Близко или идентично созданному макету.
   - Все интерактивные элементы которого совпадают с задумкой дизайнера.

4. Верстальщика:
   - Имеет поясняющие комментарии для лучшей поддержки кода.
   - Содержит понятный и легкочитаемый код.
   - Соответствует единому подходу к написанию разметки, стилей и скриптов.

5. Серверного программиста:
   - Не перегружает сервер "качественно"(часть вычислительных операций берут на себя скрипты на клиентской стороне).
   - Не перегружает сервер "количественно"(присутствует компановка изображений и файлов для единой загрузки, имеется предзагрузка файлов).

Q2. 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.

Основной особенностью разработки крупных многостраничных сайтов, функциональность которых может меняться в процессе реализации и поддержки, является применение архитектуры или системы распределения данных.

Также важно использовать модульность для быстрого переиспользования независимых блоков.

Помимо соблюдения принципа DRY немаловажным является использование системы контроля версий для быстрого перехода между нужными этапами разработки.

Для поддержки будет полезным использование единых стандартов разработки.

В последние годы стало популярно разрабатывать SPA-приложения, которые имею множество преимущест перед MPA/MVC.

При разработке сайтов я использую сборщик Webpack, библиотеку React, контейнер состояний Redux, css препроцессор Sass(sass), плагины в IDE/Тектовом редакторе для быстрой замены частей кода во всем проекте, линтеры для js и css.

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 (чаще просто Component) - это компонент, который несет в себе сугубо презентационную составляющую и показывает как выглядит условный блок. Этот компонент не зависит от остальной части приложения и получает данные и колбеки через props. В большинстве случаев записывается как функция и не имеет собственного состояния.

Container Component (чаще просто Container) - это компонент, который несет в себе всю логику взаимодействия и показывает как условный блок работает. Этот компонент почти не содержит разметки. Предоставляет данные и поведение для дочерних компонентов.

Подобная компонентная архитектура обеспечивает более простое переиспользование компонентов. Можно быстро изменять внешний вид компонентов не касаясь логической составляющей.

С появлением React Hooks использование функциональных компонентов стало еще более актуальным, т.к. исчез их основной минус – невозможность использования методов жизненного цикла(кроме getSnapshotBeforeUpdate и componentDidCatch).

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

В JavaScript нет классического наследования, а вместо него используется прототипное наследование - объекты наследуются от других объектов. К тому же в нем нет классического понятие классов, хотя в ES6 ввели подобный синтаксический сахар, который скрывает явную работу с конструкторами и прототипированием.

Опыта реализации JS-наследования в больших проектах у меня нет, но был опыт создания небольших игр для лучшего понимания работы наследования.

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

Для написания тестов end-to-end во фронтенде можно использовать фреймворки Mocha, Jasmine или Jest. В своих проектах на React я использую Jest и утилиту Enzyme.

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?

Отмечу у себя, что это спорный момент, отправлю сообщение с вопросов дизайнеру и верстальщику и продолжу работу над проектом. Если моя работа будет походить к концу, а ответа от дизайнера и верстальщика не будет, то спрошу у проектного менеджера или тимлида.

Если в команде есть соглашение, что все пробелы подобного рода заполняются через соблюдение правил style guide, то сделаю в соответствии с ними.

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

- Таск-раннер - Gulp
- Сборщик модулей - Webpack
- Система контроля версий - Git
- Пакетный менеджер - npm
- React Developer Tools
- Redux DevTools
- Препроцессор css - Sass
- Шаблонизатор и препроцессор html - Pug
- Линты: JSlint и Stylelint
- Проверка совместимости свойств и параметров - сервис.caniuse.
- Проверка актуальных разрешений экранов – statcounter.
- Тестирование через эмуляцию – browserstack.
- Плагины для VS Code: emmet, prettier, eslint, git history, htmltagwrap, TODO hightlight, autofilename и прочие
- Горячие клавиши в IDE/текстовом редакторе.
- Тестирование JavaScript - Jest (+ Enzyme)

Q8. Какие ресурсы вы используете для развития в профессиональной сфере? Какие ещё области знаний, кроме тех, что непосредственно относятся к работе, вам интересны?

Примеры ресурсов:

Справочники: developer.mozilla, learn.javascript, ...

Поиск решений: stackoverflow, toster, ...

Статьи: habr, medium, ...

Курсы: ~~изучаю сливы платных кусров~~

Youtube каналы: freeCodeCamp, ITVDN, loftblog, skillbox, codedojo, Хауди Хо, Гоша Дударь, ...

Мне ближе JS-программирование, чем верстка.

Помимо javascript мне интересен UX-дизайн, английский язык, back-end разработка, алгоритмы и математика.

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

С ранних лет увлекаюсь информатикой, математикой и программированием. Начинал свой путь с изучения Basic и Fortran. Позже перешел на Pascal, C и затем на C++. В старших классах увлекся web разработкой и понял, что это мое.

С тех пор пробовал писать на десятке языков, от python и php до prolog и языков ассемблера.

Учебу в университете рассматривал для себя только на кафедре связанной с IT, поэтому поступил в МГТУ им Н.Э. Баумана на "Компьютерные системы и сети".

Почти все свободное время уделяю изучению новых технологий и практике. За последний год изучил более 10 платных курсов, связанных с JavaScript (React, Redux, Node.js и т. д.).

Крупных работ в портфолио нет, часть проектов залиты на github.