# Квалификационные задания для разработчиков JavaScript

## Level I

### Q.1

а) Пользователя: интуитивно понятный интерфейс, минимальные усилия для получения нужной информации, не навязчивая анимация, минимальное ожидание отклика от приложения, нет «вырви глаз» цветовой вакханалии, быстрая загрузка приложения

б) Менеджера проекта: понятные требования заказчиков к проекту, сроки

в) Дизайнера:  продуманная архитектура проекта и навигация, гармоничная цветовая гамма, гибкая верстка проекта(для внесения изменений :D)

г) Верстальщика:  продуманная архитектура проекта и навигация, адаптивность

д) Серверного программиста: документация к серверной части приложения, тесты, единообразие, стабильность

### Q.2

К сожалению, опыта в разработке крупных многостраничных сайтов у меня нет. Однако, если функциональность может меняться со временем, то реализация должна быть как можно гибче ;D

### Q.3

Presentational Components – компоненты, которые отображают данные. Не манипулируют данными, обычно не имеют собственного стейта(State). Имеют свою разметку и стили.

Container Components – компоненты, которые работают с данными. Обычно не имеют собственной резметки и стилей. Предоставляют данные и поведение для других компонентов. Зачастую имеют собственный стейт(State)

Данный подход деления компонентов позволяет легче переиспользовать компоненты

Плюсы:

- вы упрощаете понимание app и UI
- периспользование (переиспользование компонентов с новой логикой или наоборот)
- нет дублирования разметки

Минусы:

- нужно заранее продумать, какие данные потребуются компонентам и как передать их

**Q.4**

В плане наследования JavaScript работает лишь с одной сущностью: объектами. Каждый объект имеет внутреннюю ссылку на другой объект, называемый его **прототипом.** У объекта-прототипа также есть свой собственный прототип и так далее до тех пор, пока цепочка не завершится объектом, у которого свойство prototype равно null. Это называется прототипным наследованием.

ECMAScript 6

```
class Polygon {
        constructor(height, width) {
                this.height = height;
                this.width = width;
        }
}


class Square extends Polygon {                    ← наследуемся от Polygon
        constructor(sideLength) {
                super(sideLength, sideLength);  ← вызов конструктора родителя
        }
        area() {
```

```
            return this.heigth * this.width;
        }
        setSideLength(newLength) {
            this.height = newLength;
            this.width = newLength;
        }
}

var square = new Square(2);
```

ECMAScript 5

```
function Polygon (heigth, width) {
        this.name = height;
        this.width = width;
}

function Square(sideLength) {
        Polygon.call(this,  sideLength,  sideLength);   ← вызываем родителя
}
Square.prototype = Object.create(Polygon.prototype);  ← задаем прототип
Square.prototype.area = function() {
        return this.height * this.width;
}
Square.prototype. setSideLength = function() {
        this.height = newLength;
        this.width = newLength;
}
```

Из опыта реализации наследования без использования js-фреймворков:

Наследование реле в слоте(однорукий бандит)

**Q.5**

Cypress, Nightmare, CasperJS, TestCafe, Nightwatch – Дядюшка гугл подсказал.

К сожалению, какого-либо опыта в данной области не имею.

**Q.6**

Уточню детали у проект менеджера.

**Q.7**

Сниппеты, Eslint, тесты(в моем случае это пока что в теории), ide, различные расширения редактора кода

**Q.8**

habr.com

medium.com

mozilla MDN

в данный момент, интересов к другим областям знаний у себя не наблюдаю ;(

**Q.9**

Зовут меня Александр, мне 24 года. Проживаю в г. Петрозаводске р. Карелия. Имею высшее образование(бакалавр). Спокойный, неконфликтный, умею работать в команде, способен быстро обучаться.

Из опыта работы - это развитие и поддержка https://vk.com/che_umniy

# Level II

https://github.com/abutyrev/route-editor