Темы для самообучения

ООП

- Статическое поле/метод.
- Инкапсуляция; модификаторы доступа: public, private, protected; свойство класса (геттер, сеттер).
- Наследование, Абстрактный класс.
- Интерфейс класса, имплементация интерфейса.
- Множественное наследование, проблема ромбовидного наследования, наследование через интерфейсы.
- Связность, связанность. Способы снижения связанности.
- В каких случаях стоит применять композицию, а в каких наследование, при проектировании.
- Ассоциация, Агрегация, Композиция, разница между ними.
- UML диаграмма классов.
- Шаблоны: Издатель-подписчик (Publish–subscribe), Одиночка, Фабричный метод.
- MVC.
- СОЛИД, первые четыре принципа (поверхностное представление).

Верстка

- **bem**. Все верстка должна быть по БЭМу, показать в примерах. Без инструментов, только методология (именование и декомпозиция)
- сверствать 2-3 колонки всеми способами способами: **float, inline-blocks, position**, tables. Опционально: flex
- отцентрировать блок. По горизонтали, по вертикали. Без js и с.
- **примеры transition и animation**, включа варианты с аппаратным ускорением
- примеры адаптивной верстки: на телефоне, планшете и десктопе. media-queries, виртуальные пиксели, подход s, m, l, xl
- подключение к проекту кастомного шрифта, с разными начертаниями
- пример испольования спрайтов, для иконок, под ретину

JavaScript

- понимание и обоснование каждого из пунктов стайлгайда. Это 80% по JS.
- this, call, apply, bind.
- **prototype**, рассказ что это и как работает.
- **событийность**. Принцип использования и принцип работы событий в jQ и GCT, способ их эмуляции в js
- способы оптимизации работы с ДОМом. Список самых дорогих операций
- использование промизов на js
- **манипуляции с DOM** - поиск, добавление, изменение, удаление классов и узлов (в любой либе).

Чистый код

Чеклист представляет из себя оглавление из книги чистый код. Книга большая и очень полезная, но я обынчно рекомендую ее читать по оглавлению. Имя большинства глав – тезис. Если для вас он очевиден, и не подразумевает разночтений, и вы с ним согласны – можно не чиать. В противном случае читать обязательно. Можно не соглашаться, но делать это надо аргументировано. "А мне не нравится" – не аргумент.

Примеры согласия или несогласия с каждым из тезисов надо будет суметь показать в личном или эталонном коде.

Именование

- Имена должны передавать намерения программиста
- Избегайте дизинформации (контр-пример)
- Исопльзуйте осмысленные различия
- Используйте удобопроизносимые имена
- Выбирайте именная удобные для поиска
- Избегайте схем кодирования имен
- Избегайте мысленных преобразований
- ⚠️Имена классов и методов обязательны к прочтению, и пересказу
- Избегайте остроумия
- Выберите одно слово для каждой концепции
- Избегайте каламбуров
- Используйте имена из пространства решения
- Используйте имена из пространства задачи
- Добавьте содержательный контекст
- Не добавляйте избыточного контекста

Функции

- Компактность
- Правило одной операции
- Одинь уровень абстракции на функцию
- Аргументы функций
- ⚠️ Особеннсти аргументов функци (к прочтению)
- Избавтесь от побочных эффектов.
- Разделение команд и запросов.
- Не повторяйтесь
- ⚠️ Структурное прогаммирование.

Обязательный раздел для стажировки.

Комментарии

- Комментарии не компенсируют плохого кода.
- Юридические комментарии.
- Информативные комментарии.
- Представление намерений.
- Прояснение.
- Предупреждения о последствиях.

- TODO.
- Усиление.
- Комментарии Javadoc в общедоступных API
- ⚠️ Виды плохих комментариев

Форматирование

- Вертикальное форматирование
- Газетная метафора
- Вертикальное разделение концепций
- Вертикальное сжатие
- Вертикальные растояния
- Вертикальное упорядочивание.
- ⚠️ Google javascript styleguide форматирование имеет больший приоритет над книгой, как стандарт в компании. Оно лучше покрывает форматирование

Объекты и структуры данных

- ⚠️ Обязательны к прочетнию, пересказу и демонстрации в коде.
- Классы
- Строение класса
- Инкапсуляция
- Классы должны быть компактными
- Принцип единой ответственности
- Связность.
- Поддержание связности приводит к уменьшению классов.
- Структурирование у учетом изменений.
- Изоляция изменений.

Обязательный раздел для стажировки.

Node.js

- Модульная система - как работают module.exports и require
- Gulp - самые основные моменты
- Основные утилы ноды (fs, path etc)
- Express - как использовать app.use, отдача статики по разным путям, middleware (подключаемые и свои)
- Node 4 ES6 - arrow функции, const
- npm - установка пакетов, что делают -D, --save-exact, структура зависимостей в npm 3

JS асинхронность

- ES6 Promise - основные методы, создание промиса из функции с колбэком
- asyncawait - основные методы, использование вместе с промисами
- Последовательное/асинхронное выполнение нескольких асинхронных функций без превращения кода в кашу

БД

- Sequelize - CRUD операции, миграции, транзакции, связи между моделями
- Связи в БД вне ОРМ - внешние ключи, ON DELETE
- SQL в общих чертах

Linux

- Базовые команды: ls, cd, cat, tail, echo etc.
- Grep, регулярки

Теория

- MVC - Отношения между моделью, вьюхой и контроллером
- REST - принципы наименование методов, принцип stateless
- Используемые коды ошибок

GIT

- **к**ак добавить файлы в коммит? Как добавить чанки? Почему нельзя добавлять файлы по звезде или точке?
- **п**уш. Что такое фастфовард? Чем опасен пуш с форсом?
- **в**етвление. Как создать и удалить ветку, локально и в origin`е. Как привязать локальную ветку "a1" к удаленной ветке "b2", и пушить в нее? Как получить список комитов из локальной ветки, которую вы только что удалили? Сколько веток может указывать на один коммит?
- **ч**то такое мердж? Как его откатить? Какие бывают стратегия мерджа? Что такое ребейз? Зачем нужен пулл по ребейзу?
- **к**ак переименовать коммит? Как удалить последний? Как удалить коммит из середины и запушить его? В каких случаях так можно делать, а в каких нет?
- **к**ак пользоваться reflog?
- git successful. Нарисовать диаграмму с нуля, со всеми видами веток и допустмыми связями между ними.
- что такое хуки?

Консоль

- **с**оздать файл и директорию, удалить, скопировать, переместить
- **c**hmod
- **s**sh, scp
- **f**ind/grep
- **v**im: создать файл, сохранить, отредактировать, выйти (с сохранением и без)
- **a**pt-get, apt-cache
- **t**op/kill