[Draft] HARD SKILLS v.2.0

Хайлайт - новый материал

Компетенция / уровень	Стажер	Junior	Junior+	Middle	Middle+	Senior	Senior+
Контекст выполнения, this, области видимости, замыкания		Определение контекста выполнения, случаи применения Вычисление контекста, его значение по умолчанию Сохранение контекста выполнения через замыкание; Потеря контекста; Виды объявления функций, переменных Всплытие переменных Псевдомассив arguments	Замыкание в циклах и условных конструкциях, возможные проблемы	Лексическое окружение Механизм поиска переменных Подмена существующего контекста	Определение контекста при обработке браузерных событий Обнуление контекста Синтаксические особенности стрелочных функций	IIFE Каррирование Принципы работы сборщика мусора в V8 Процесс обработки кода в V8: • AST • Предварительный и полный лексический анализ • Just-in-time • Компиляция, интерпритация • Деоптимизация скомпилированного кода	Оптимизации в V8: • Скрытые классы • Исправление типов аргументов • Определение общих сущностей в глобальной области видимости
Прототипы, классы		Определение прототипа, класса	Наследование классов	Цепочка прототипов, механизм поиска свойств Особенности прототипов стрелочных функций	Реализовывать функции-фабрики Реализация прототипного наследования Создание объектов без прототипа, способы и область применения	Преобразование классов движком при создании Приватные и статичные поля классов Примеси Шаблон делегирования	Производительность, расширение прототипа против замыкания
DOM		Методы поиска DOM узлов Методы изменения содержимого в DOM узлах Методы добавления и удаления DOM узлов Основные браузерные события	Всплытие и погружение событий Нативная обработка форм и полей	Живые коллекции, методы обработки и преобразования в массив Предотвращение механизма всплытия, делегирование событий	MutationObserver Selection и Range	WebComponents • ShadowDom • Custom elements • HTML templates	Особенности CSS при работе с WebComponents
Асинхронность		JS - однопоточный и синхронный язык	XMLHttpRequest, fetch	Механизм Event loop Как работают таски/микротаски Как используются: • callback • Promises	Callbackhell, методы борьбы с ним	async/await Особенности обработки цепочки callback-ов Promis-а после отлавливания ошибки	Функции-генераторы, область их применения Создание бесконечных циклов функциями-генераторами, область применения

Браузер и его инструменты		devTools network breakpoints	localStorage iframe	WebSocket cookies	Безопасность cookie Установка cookies сервером, httpOnly	Service workers Web-workers Web APIs (Payment Request API, Push API, Web Share API и т.д.) indexDB	Worklets PWA
Регулярки		Назначение регулярных выражений Создание регулярных выражений через литерал	Основные спец. символы: \d, \s и другие, в том числе границы: ^, $ и т.д. Методы работы со строкой: replace, match, search	Глобальный объект RegExp и его методы Группировки *(x)* Наборы и диапазоны Квантификаторы – *x*, x?* и другие	Границы слова в регулярном выражении Обратные ссылки	Оптимизация производительности регулярных выражений (жадный и ленивый поиск, критический возврат) Опережающие и ретроспективные проверки	Поиск на заданной позиции
Обработка исключений, логирование, дебаг		try-catch Методы console (log, error, warn и т.д.)	Дебаг клиентского кода через браузер	redux devtools	Chrome DevTools: application, sources	Настройка логирования ошибок (sentry, prometheus и др.)	Дебаг серверного кода через браузер Понимание принципа выделения программных узлов, которые необходимо логировать
React		Назначение React Компоненты, свойства: Props, State, однонаправленный поток данных Особенности синтаксиса JSX	Хуки: встроенные и пользовательские Жизненный цикл компонента Компоненты высшего порядка	Virtual DOM Способы оптимизации React-приложения Фрагменты Предохранители Порталы и как ими пользоваться React router, компонент withRouter React Context SSR	Compound components Form managers	render-props Механизм Reconciliation	Архитектура Fiber Использование WebComponents в React, область применения Server components
Redux		Redux, область применения Связывание react с redux action, dispatch Иммутабельность store	Комбинирование State и store в компоненте, по какому принципу надо разделять данные	redux-saga, redux-thunk Альтернативные подходы к оргнизации store - Feature-first + redux-ducks	Redux-toolkit	Библиотеки для осуществления нормализации данных и упрощенной работы с иммутабельными структурами (immutable, normalizr и тд)	Преимущества и минусы иммутабельност и данных в js
TS		TypeScript, область применения	Как типизация приводит к уменьшению багов Основные типы данных в TS	Interface, types Файлы декларации .d.ts Механизм typeGuard Преобразование типов (extend, |, &, UtilityTypes) Generics Keyof, typeof	Optional chaining, nullish coalescing Conditional types, mapped types	Template Literal Types Перегрузка функций Reference types Использование tsc	Декораторы Mixins

NodeJS		NodeJs, зачем нужен	Плюсы использования NodeJs для фронтовых приложений	Основные встроенные модули (http, fs) и глобальные переменные окружения (global, process) Работа с загружаемыми модулями (npm, yarn, npx) Рутинг приложения Отдача статики	Отличие и особенности SQL и NoSQL баз данных Асинхронность, события Работа с конфигами для различных сред разработки	Кеширование (через node-cache, например) Подключение баз данных к серверу, CRUD	Параллелизация через workers_threads
Вёрстка		Блочная модель Семантика, базовые теги HTML5 Базовые CSS свойства Векторная и растровая графика. Преимущества и недостатки Типографика. Типы шрифтов. Свойства для изменения шрифтов. Подключение шрифтов	БЭМ Базовые псевдоклассы	Современные пре /постпроцессоры (SCSS, PostCSS) Flexbox Анимация Контекст наложения	Современные подходы к организации стилей (CSS-Modules, CSS-in-JS) Grid	browserlist Определение инструментов в зависимости от используемых браузеров Дизайн система	Доступность (aria, tabindex & etc.) Микроразметка
Сеть		ip-адрес и hostName Различие POST и GET Query-параметры	REST API, RESTFUL API	Понимание протоколов HTTP, HTTP2, HTTPS Принцип API first	Различие в подходах Long-Polling, Websockets and Server-Sent Events	Теория работы сетей. TCP /IP	Модель OSI
Производительность		Инструменты проверки производительности страницы (devtools, lighthouse)	Preload, prefetch Async, defer	Метрики клиентской производительности	Механизмы уменьшения размера бандла CDN	lazy-loading Оптимизация критичного пути	Метрики серверной производительности
Безопасность		eval, dangerouslySetInnerHTML Атрибут rel и его значения noreferrer, noopener, nofollow	Аутентификация и авторизация пользователя	CSP CORS XSS SQL-injections CSRF	npm audit TCP hijacking X-Frame-Options	OWASP top-10	Работа SSL/TLS Организация безопасности при реализации WebSocket-ов Безопасность JWT
Тестирование		Назначение unit тестов	Паттерны AAA, AAS unit тесты - jest	unit тесты - enzyme / React testing library Моки. Зачем нужны и как применять Стабы. Зачем нужны и как применять Принцип TDD Принцип BDD coverage, что определяет	e2e тесты, cypress	Интрументализация кода, назначение и инструменты	1-2 дополнительных инструмента для тестирования Понимать принцип выбора подходящего инстумента тестирования

Git		Назначение системы Git Базовые команды: • pull • push • commit • fetch	Команды: • init • config • checkout • merge	Сложные команды • amend • fixup • revert • cherry-pick • stash • reset • tag • log • diff • reflog	Подходы к организации пространства: • git-flow • github-flow • gitlab-flow	hooks switch restore grep	git-lfs bisect worktree
Docker		Что такое docker, зачем нужен	Базовые команды: • pull • run	Написание Docker-файла Команды для подключения к запущенному контейнеру, просмотр логов	Что такое multi-stage build docker-compose	Как устроены слои в docker-образе docker-swarm	Облачные платформы Модели взаимодействия с облачными платформами: • Paas • Iaas • Saas
CI/CD		Зачем нужен CI /CD	Прекоммитные проверки	Jenkins Mesos/Marathon	Понятие кластера серверов Зачем нужны системы управления кластерами	Конфигурировать имеющиеся системы Jenkins, Marathon и другие, используемые в банке	Знать несколько систем ci/cd отличных от банковских, их плюсы и минусы
Webpack		Что такое Webpack	Как работает конфигурация	Зачем нужны лоадеры Зачем нужны плагины Семантическое версионирование стартовые скрипты в package.json HMR Tree shaking Минификация	Разделение конфига на prod /dev bundle-analyzer Разделение бандла на чанки	Механизмы ускорения горячей/холодной сборки проекта Уметь пользоваться 1-2 другими инструментами для сборки (gulp, rollup) Понимать принцип выбора сборщика исходя из задачи	Module Federashion Как писать свои плагины для babel и postcss
Алгоритмы и структуры данных		Алгоритмы, критерии их оценки	Нотация Big O, зачем используется	Уровни Big O нотации, оценка сложности написанного алгоритма	Стандартные алгоритмы сортировки Стандартные подходы к решению алгоритмических задач (two pointers, backtracking, binary search)	Какие нотации помимо Big O используются для оценки сложности алгоритма Основные сущности для решения алгоритмических задач и методы работы с ними: • Графы (направленные и не направленные) • Стек • Hash-map • Связный список	Динамическое программирование
Разработка программного обеспечения		ООП, основные термины	Функциональное программирование, его отличие от ООП и применимость в JS	SOLID	MVC MVP MVVM	Подходы к разработке программного обеспечения: • Водопад • V-образная модель • Инкрементная модель • Итеративная модель • Спиральная модель	Парадигмы программирования: • Метапрограммирование • Императивное • Декларативное

Паттерны		Паттерны: • Одиночка • Фасад • Наблюдатель	Паттерны: • Адаптер • Команда • Декоратор	Паттерны: • Фабричный метод • Прототип • Заместитель	Паттерны: • Цепочка обязанностей • Абстрактная фабрика • Стратегия	Паттерны: • Посредник • Компоновщик • Снимок Dependency Injection	Паттерны: • Шаблонный метод • Приспособленец • Мост
Разработка программного обеспечения		Форматирование: • Вертикальное форматирование • Газетная метафора • Вертикальное разделение концепций • Вертикальное сжатие • Вертикальные расстояния • Вертикальное упорядочение Комментарии: • TODO • Комментарии Javadoc в общедоступных API	DRY KISS	YAGNI Основные принципы рефакторинга: • Код должен становиться чище, при этом не должна создаваться новая функциональность • Предварительно должны быть написаны тесты	Уверенно находить свойства "грязного" кода: • Избыточные комментарии • Неинформативные комментарии • Высокая связанность кода • Большое количество аргументов в функции • etc Применять различные методы рефакторинга: • Извлечение метода или переменной • Инкапсуляция полей • Разбиение условного оператора • etc	APO – избегание преждевременной оптимизации BDUF – масштабное проектирование превыше всего Применять метод рефакторинга через паттерны проектирования	Оценивать объемы рефакторинга, планировать его и грамотно организовывать Составление технического беклога по исправлению проекта