Тулинг

Status	Done
URL	https://youtu.be/1lUggEp9Y7k

IDE / Terminal (настройка)

Что лучше?

- Локально: IDE или терминал на выбор
- Удаленно: терминал

Что полезно/важно для терминала?

- Поддержка unicode/emoji и клавиш Esc, F1…F12
- Для MacOS лучше всего iTerm2
- Интерпретатор команд: bash, zsh
- Автокомплит: bash-completion, git completion, npm completion
- Midnight commander, Far - файловые менеджеры

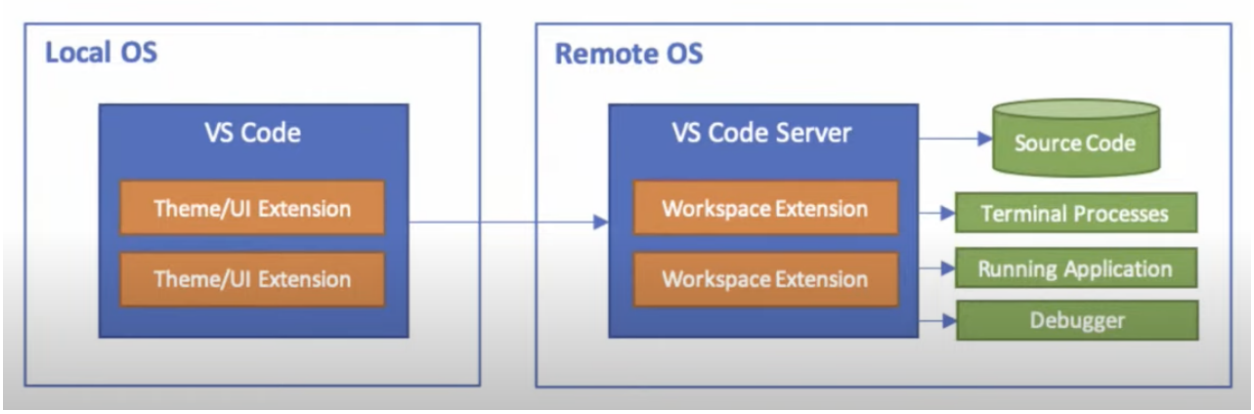
- vim

Удаленная разработка

Терминал

- ssh - безопасное подключение к серверу по TCP. Проблема: соединение может прерываться
- tmux, mosh: решают проблему с отключениями
- Midnight commander, vim

VS Code Remote developer



CodeServer

- Независимое решение

JetBrains

- JetBrains Gateway

Еще варианты

- Linux X Window + WebStorm

- Windows + Windows Server + RDP (remote desktop protocol)

Типичный круг задач

Система контроля версий

- git --help
- MacOS, Linux: ставим автодополнения
- git-prompt - вывод полезной информации о ветке (на больших репозиториях может тормозить)
- Без тормозов:
  - PS1 = “\w \$(__git_ps1 ‘(%s)’)\$ (для bash)
  - PROMPT = ‘%2/ $(__git_ps1 “(%s)”) $’ (для zsh) ?

WebStorm + Git

- Можно использовать Partial Commit (разбивать изменения на несколько коммитов)
- История изменения файла
- Interactive Rebase
- Hotkeys
- Похожая работа в Git/SVN/etc.
- Annotate(blame)
- Update project в 1 нажатие клавиши === stash + pull + unstash + resolve conflicts
- Удобное дерево файлов с изменениями
- Просмотр diff
- Минус - есть много памяти

VS Code + Git

- GitLens - обязательно ставить
- История изменения файлов
- Interactive Rebase
- Code > Preferences > Settings
  - Git Allow Force Push
  - Git: Auto Stash

Сборка

- Все команды сборки пишем в package.json
- Запуск: npm run myCommandName
- npm run выводит все возможные команды
- Автокомплит в bash
- Кроссплатформенность:
  - NODE_ENV=development webpack не работает в Windows cmd
  - Совместимость: никаких shell-скриптов, только вызов npm-пакетов или своего js
  - shelljs, dotenv, cross-env
  - cross-env NODE_ENV=development webpack будет везде работать одинаково
- Совместимость с новыми node.js
  - fsevents v1, chokidar v1 v1 не поддерживаются node v14++
- Совместимость с Apple M1/M2

Webstorm

- Знает про npm, webpack и др
- Запуск команд из секции scripts в package.json
- Быстрый переход к редактированию команды
- Автокомплит и валидация

VS code

- Знает про npm
- Запуск команд из секции scripts в package.json
- Автокомплит и валидация (послабее)

Навигация по коду

Терминал

- Помним структуру проекта
- Используем файловый навигатор
- Поиск по содержимому: пропускаем .git, node_modules
- Поиск разной степени умности и скорости: grep, ack, ag

WebStorm

- Поиск заранее по индексированному содержимому (быстрый, но долго индексирует и плохо работает с необычными ФС)
- Поиск файла по части имени
- Дерево проекта
- Go to definition/implementation
- Поиск/замета фрагмента AST (Search/Replace Structurally) - поиск с игнорированием форматирования

VS Code

- Поиск файла по части имени
- Дерево проекта
- Go to definition/implementation/reference

Рефакторинг

> 📕 “Рефакторинг”, Мартин Фаулер

Терминал

- Безопасный рефакторинг:
  - Покрыть код тестами
  - Убедиться, что тесты действительно проверяют код
    - Jest: expect.hasAssertions()
    - await promise.rejects
    - плагины для ESLint
  - Убедиться, что тесты не падают
  - Изменить код
  - Убедиться, что тесты не падают

Webstorm умеет рефакторить код

- Гарантия правильности кода после рефакторинга
- Вынос константы
- Преобразование логических выражений

- Замена function на arrow function и обратно
- Деструктурирование
- Удаление лишних скобок
- Подсветка одинаковых участков кода
- Замена if на ?, инверсия ветвей
- и тд

VS Code: Плагины

- JavaScript Booster
- Intelli Refactor
- Intelli Code

Линтеры

Терминал

- EsLint, Stylelint
- Никаких TSLint и JSHint!
- Запуск линтеров из скриптов npm
- Watcher: eslint-watch, chokidar-cli (автопроверка при сохранении)
- Как правильно игнорировать ESLint:
  - `/*eslint-disable*/` - выключает до конца файла, поэтому так делать плохо
  - `/*eslint-disable max-len*/` - тоже до конца файла
  - `//eslint-disable-next-line max-len` - так хорошо

Webstorm

- Поддержка из коробки предлагает исправления

VS Code

- Поддержка плагинами
- Поддержка fix
- Suppress rule for current line

В любом варианте

- Для гарантии должен быть линтер в одном из мест:
  - В pre-commit/pre-push hook
  - В скрипте сборки проекта локально и в CI

Тесты

- Jest, Mocha
- Запуск тестов из скриптов npm
- Watcher: jest --watch, chokidar-cli

Webstorm

- Распознавание и запуск тестов Jest, Mocha, Karma
- Обработка стектрейсов ошибки
- Обработка статусов тестов
- Сниппет для it

Отладка

Терминал

- Запускаем node.js с флагами
  - —inspect[=[host:]port] (активировать инспектор на заданных host:port)
  - —inspect-brk[=[host:]port](активировать инспектор на заданных host:port, не выполнять скрипт и ждать подключения)
- Подключаемся из Chrome DevTools

WebStorm

- Можно прямо в редакторе работать отладчиком и точками останова (все, что есть в Chrome DevTools)

VS Code

- Можно прямо в редакторе работать отладчиком и точками останова (все, что есть в Chrome DevTools)

Полезные ссылки

Devtools (Chrome)

Как открыть?

- Через контекстное меню
- Через пункт в верхнем меню
- Через хоткей F12
- Через специфичный хоткей (подписаны в верхнем меню)

- Если сайт сопротивляется, можно вставить урл в страницу с уже открытыми devTools

Панель Elements

- Можно выделять элементы DOM-дерева
- Можно менять стили и атрибуты прямо в dev-tools
- На некоторых элементов видны бейджики (grid, flex)
- Кога наводимся мышкой на селектор, подсвечиваются все подходящие элементы
- Можно искать Cmd+F (по классу или по подстроке)
- Можно формить состояние (hovered и тд)
- Можно делать скриншоты
- Scroll into view - прокручивание до нужного элемента
- Брейкпоинты, когда js пытается модифицировать дерево
- Можно видеть блочную модель
- Можно видеть перекрытие
- Можно смотреть все обработчики событий (и удалять их)

Панель Console

- Видим тут сообщения об ошибках
- Можем запускать свой код
- $0 показывает выделенный элемент

Панель Sources

- Исходники
- Минифицированные скрипты (но можно нажать pretty-print)
- Cmd+P - поиск
- Брейкпоинты (Хорошо ставить Pause on uncaught exceptions)
- Можно добавить в ignore-list
- Точки останова:
  - Простая
  - Условная (работает при условии типа i === 50)
- LogPoint - в этой строчке будет выведено какое-то сообщение (какое захотим, как console.log, но без пересборки)
- Панель Overrides - можем открыть какой-то файл и сохранить его отредактированным
- Есть возможность остановить выполнение при ajax-запросе
- Можно ставить брейкпоинт из консоли: debug(someFunction)
  - Отмена - undebug(someFunction)

Панель Network

- Чекбокс Preserve log - отмена очистки при обновлении (полезно при редиректах)
- disable cash - полезно при локальной разработке (если смотрим продакшн, лучшее отключить)
- Управление замедлением сети (принципиальное отличие: пакеты не теряются)
- Колонки можно добавлять
- По колонкам можно сортировать

- Сложный поиск можно делать через Filter (например смотреть, что загружено из кэша, фильтровать по размеру, типу и тд)

Панель Performance

- Анализ производительности
- Есть запись (до полной загрузки + несколько секунд)
- Видим, что когда происходило (скриншоты и цвета)
- Свой внутренний Network
- Отметки Web Vitals
  - FP - First Faint
  - FCP - First Contentful Paint
  - DCL - DOM Content Loaded
  - L - Loaded
  - LCP - Largest Content Part
- Main - основной поток выполнения браузера

Панель Memory

- Анализ работы с памятью

Панель Coverage

- Какая часть кода реально загружается и какая используется

Панель Application

- Нужно для работы с service workers

- Storage
- Cookies
- Web SQL
- Back/Forward cache - возможность быстро возвращаться на предыдущую страницу

Панель Lighthouse

- Анализ по категориям (что можно улучшить)

React developer Tools

- Добавляет вкладки Components и Profiler(в проде не работает)
- В Profiler можно смотреть отдельные рендеры