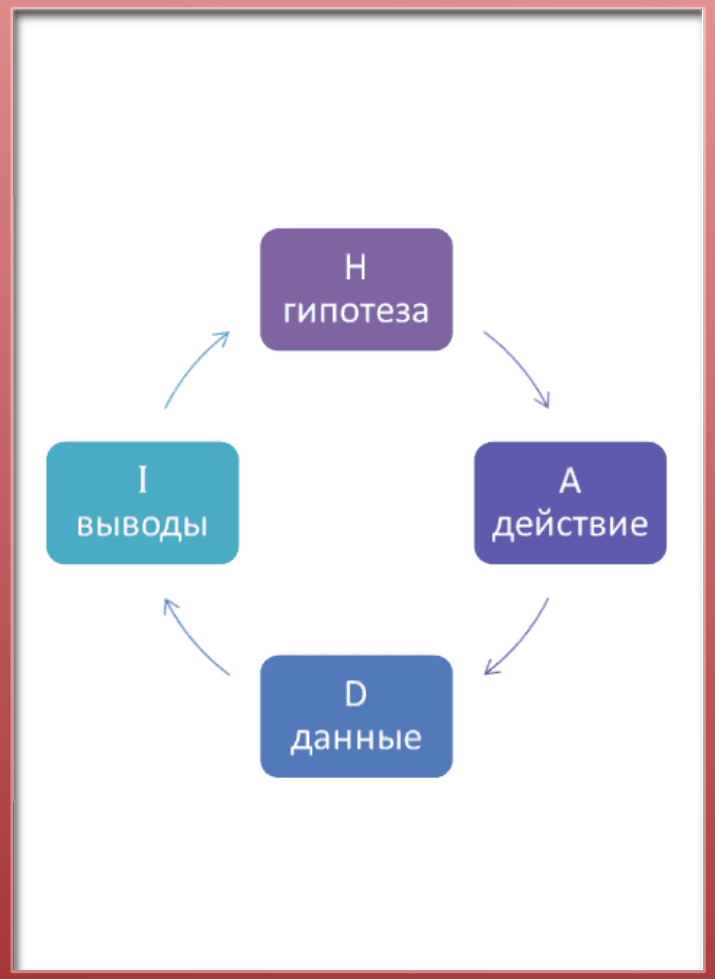
БОЛЬ
И ЕЁ УСТРАНЕНИЕ

Основные проблемы при разработке интерфейсов и предлагаемый способ их решения с использованием современных технологий web разработки.



ПРОБЛЕМА: ДЛИННЫЙ ЦИКЛ ОТЛАДКИ

- Редактирование кода.
- Запуск приложения.
- Проверка и обнаружение проблем.
- Исследование причины проблем.

И так для любой опечатки.

Чем быстрее разработчик поймёт где и почему ошибся, тем быстрее он всё реализует.

ПРОБЛЕМА: ГДЕ ЧТО ЛЕЖИТ?

- Как быстро перейти к определению сущности?
- Как быстро найти все места использования сущности?

Детективные расследования снижают полезную продуктивность.

ПРОБЛЕМА: КАК ЭТО ИСПОЛЬЗОВАТЬ?

- Что это за объект я тут получил?
- Какие поля и методы есть у объекта?
- Какие параметры принимает функция?
- Какие вообще есть классы и функции?
- Правильно ли я использую сущность?

Ответы требуют времени, иногда продолжительного.

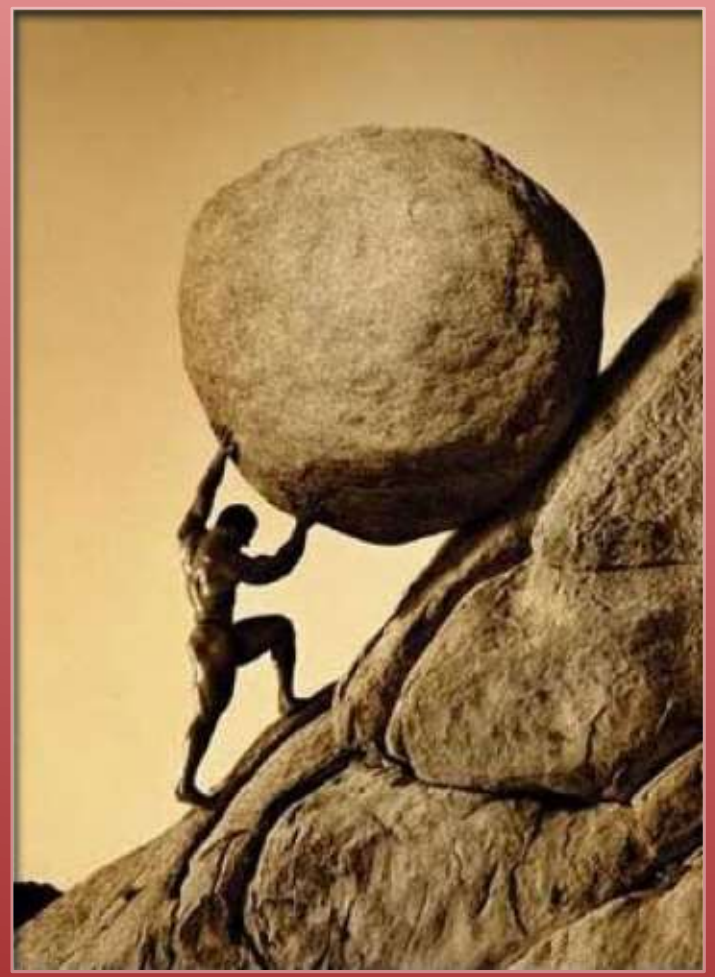
РЕШЕНИЕ: СТАТИЧЕСКАЯ ТИПИЗАЦИЯ

- Проверка кода по мере редактирования.
- Быстрая навигация по коду.
- Контекстные подсказки и документация.

TypeScript = современный JavaScript + типизация.

ПРОБЛЕМА: МНОГО КОДА

- Много ошибок.
- Медленная разработка.
- Медленная работа приложения.
- Требуется больше тестов.



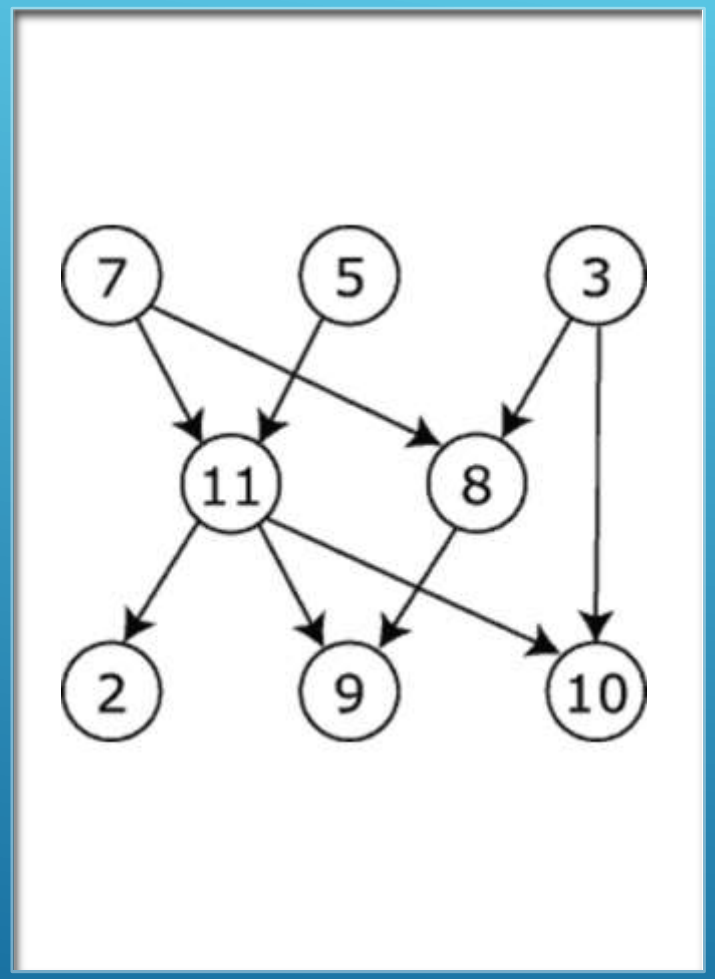
ПРОБЛЕМА: РАСТУЩАЯ СЛОЖНОСТЬ

- По мере роста приложения всё сложнее вносить в него изменения не разломав.
- Требуется учитывать всё большее число состояний одновременно, разбросанных по разным частям приложения.
- Множество внутренних соглашений неизбежно начинают противоречить друг другу (например, что в каком порядке следует вызывать).

ПРОБЛЕМА: ОБНОВЛЕНИЕ ПРИЛОЖЕНИЯ

- Требуется перезагрузка всей страницы.
- Пользователь выпадает из контекста работы (сбрасываются позиции скроллингов, открытые панели, редактируемый текст).
- Разные пользователи работают с разными версиями приложения одновременно.

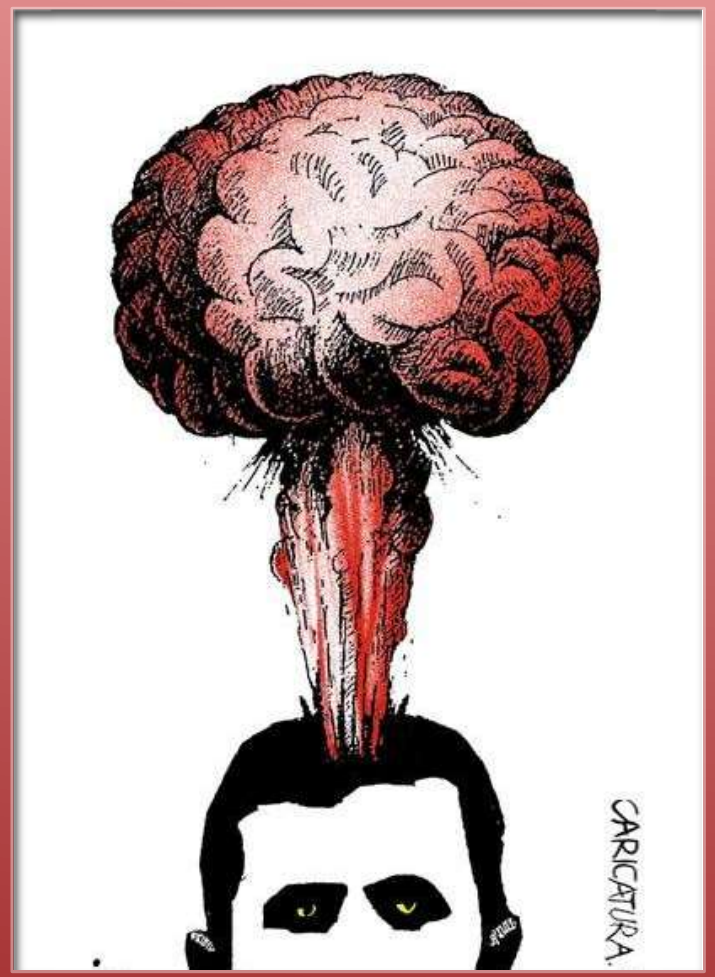
Обычно об этих проблемах не думают, просто некоторым пользователям иногда не везёт.



РЕШЕНИЕ: РЕАКТИВНАЯ АРХИТЕКТУРА

- Описание состояний вместо действий.
- Надёжная каскадная синхронизация состояний.
- Эффективное обновление состояний.
- Минимум обслуживающего кода.
- Обновление приложения налету незаметно для пользователя.

По ссылке статья о принципах работы



ПРОБЛЕМА: СЛОЖНОСТЬ ОСВОЕНИЯ

- Много разных сложных абстракций.
- Не очевидные соглашения.
- Много запутанного кода.
- Средства отладки зачастую не помогают.
- Требуются специалисты высокого класса.
- Низкая отдача от специалистов низкого класса.

ПРОБЛЕМА: НЕУНИВЕРСАЛЬНОСТЬ

- Несколько решений одной задачи.
- Сложно добавить функционал в готовое решение.
- Компонент умеет слишком много вообще и слишком мало из того, что требуется.
- Адаптивная реализация требует дополнительного существенного времени.

ПРОБЛЕМА: ПРОИЗВОДИТЕЛЬНОСТЬ

- Легко сделать медленно.
- Сложно сделать эффективно.

РЕШЕНИЕ:
ПРОСТОЙ И ЭФФЕКТИВНЫЙ ФРЕЙМВОРК

- Минимум абстракций и соглашений.
- Статическая типизация.
- Реактивная архитектура.
- Высокая эффективность.
- Компактный размер.
- Простота компоновки.
- Динамическая адаптация к возможностям устройства.
- Высокая настраиваемость.

ПРОБЛЕМЫ EXT JS

- Длинный цикл отладки.
- Где что лежит?
- Как это использовать?
- Много кода.
- Растущая сложность.
- Обновление приложения.
- Сложность освоения.
- Неуниверсальность.
- Производительность.

ПРОБЛЕМЫ SAP UI5

- Длинный цикл отладки.
- Где что лежит?
- Как это использовать?
- Много кода.
- Растущая сложность.
- Обновление приложения.
- Сложность освоения.
- Неуниверсальность.
- Производительность.

ПРОБЛЕМЫ ANGULAR JS

- Длинный цикл отладки.
- Где что лежит?
- Как это использовать?
- Растущая сложность.
- Сложность освоения.
- Неуниверсальность.
- Производительность.

РИСКИ СВОЕГО РЕШЕНИЯ:

- Основной разработчик может оказаться самоуверенным идиотом и не добиться поставленных целей.
- Требуется время на разработку ядра, основных компонент, документирование.
- Требуется дополнительное обучение сотрудников при переходе с других фреймворков.
- Потребуются дополнительные усилия для согласования предлагаемого решения с заказчиками.

БОНУСЫ СВОЕГО РЕШЕНИЯ:

- Требуется меньше людей для того же объёма работ.
- Требуются специалисты менее высокого класса.
- Реализованный функционал легко переносится между проектами.
- Требуется меньше времени на реализацию.