Why I hate and love Angular at the same time

Я ненавижу и обожаю Angular одновременно. Two-way binding, MVWhatever архитектура, скорость разработки небольших приложений – как бальзам на душу после пыток от создания приложений на jQuery. Однако у Angular есть проблемы: свои специфические и общие для фреймворков.

Порог входа в Angular гораздо выше чем в javascript и html/css. Если тебя бросают на проект написанный на этом фреймворке, то ты можешь довольно быстро освоиться и работать многие месяцы, думая что ангуляр - это просто. Но настанет один день, когда тебе понадобится создать Angular приложение с нуля, и окажется что всё, что ты знал о том как работать со $scope, дергать сервисы и создавать директивы является одной из граней Angular. А для того, чтобы создать приложение с нуля, нужно разобраться в Routing, видах Dependency Injection, стратегиях разбивки приложения на модули, глобальной обработке бекендовых ошибок, и еще куче вопросов по архитектуре Angular приложения.

Другой коварный минус для молодых программистов, взявших Angular как первый фреймворк, заключается в том, что Agular не учит работать с DOM и ивентами. Наличие десятка встроенных директив и тысяч плагинов решают почти все задачи, избавляя разработчика от нужды разбираться с тем как js общается с DOM и что такое ивенты. Поэтому на смену jQuery-программистам приходят Angular-программисты.

Еще одна важная причина почему многие выбирают вместо Angular другой фреймворк (в основном React или Angular 2) – его тормознутость. Если вы выбрали Angular для большого приложения с большим количеством страниц, форм и одновременно отображаемых данных на странице – будьте готовы расставлять one-time binding’и, переписывать библиотеки на jQuery, работать с DOM в обход Angular, убивать вотчеры и удалять скоупы. Проблема в том, что Angular вешает вотчеры на всё подряд и из-за особенностей работы $digest цикла дёргает вотчеры по два раза каждый раз. Поэтому ваше приложение чаще всего начнёт тормозить не потихоньку, а неожиданно. Отрендерить 20 строк в таблице с вотчерами на данных - дело доли секунды. Отрендерить 1000 - уже слишком долго, чтобы можно было показывать кому-то.

Самый большой минус Angular как и многих других фреймворков, который я ненавижу – это десятки неочевидных непрописных правил работы с Angular приложением: разделения частей приложения на модули или нет, использование $rootscope с $broadcast/$emit или нет, правило запрета работы с DOM в Controller, именование встроенных Angular модулей в Controller’ах и директивах, использование Factory или Service, использование controllerAs и this в Controller’е вместо $scope итд. Причем на каждом новом проекте часть правил используют, часть намеренно игнорируют, а о части никто не знал.

Дебаг – бич всех современных фреймворков, и Angular - не исключение. Каждый раз когда возникает внутренняя ошибка Angular, которую не удалось отловить в самом ее начале, я точно знаю случилась беда. Найти причину будет очень сложно: отсылки из консоли на официальный сайт Angularа дают только примерные представления о природе ошибки. Точного места или хотя бы файла с ошибкой узнать не получится. Хуже всего исправляются ошибки Dependency Injection и Routing’а.

Для того чтобы исправить ошибку Dependency Injection лучше всего использовать следующую стратегию по разматыванию клубка зависимостей:

1. Убираем(закомменчиваем) подключения всех модулей в главном файле app.js
2. Убираем(закомменчиваем) подключения всех Angularовских js файлов (Контроллеров и Сервисов)
3. Проверяем чтобы приложение стартовало без ошибок, загружая только index.html
4. Возвращаем в app.js подключение первого модуля
5. Раскомменчиваем соответствующий js файл в index.html
6. Проверяем что модуль подгрузился и приложение работает
7. Повторяем шаги 4-5 пока не наткнемся на модуль по возвращении которого приложение перестает работать
8. Узнав проблемный модуль нужно определить причину ошибки: проверяем правильность названия модуля, проверяем подключение файла модуля в index.html и корректность указанного к нему пути, проверяем сам Dependency Injection внутри модуля на ошибки в названиях, при использовании array notation проверяем количество подключаемых зависимостей вне массива зависимостей и внутри него

В самых худших случаем на отлов такой ошибки может уйти час если знать где и как искать.

Проблемы с роутингом решаются немного проще. Если перейдя на какую-то страницу вы видите не ту страницу что нужно или вообще пустую страницу, то чаще всегда проблема в том что вы забыли прописать роутинг, или сделали ошибку настройках урлов роутинга, или синтаксическую ошибку в названии файла. Такие проблемы решаются следующим образом:

1. Находим страницу, для которой роутинг работает верно, допустим это страница login
2. Находим настройки роутинга для страницы login и создаем их копию с единственным отличием в урле, который указывает на ваш проблемный модуль, например на settings
3. Проверяем что url /login и /settings указывают в одно и то же место модуля login
4. Делаем абсолютную копию модуля login, меняя начала названия .js файлов, затем модуля, а затем кода внутри модуля
5. Отлавливаем ошибки на шаге 4 и исправляем их

Надеюсь этот пост поможет тем кто только начинает разбираться с Angular быстрее втянуться в этот фреймворк и подтолкнет не останавливаться и разбираться во всех его деталях, любить за плюсы и прощать минусы.