1 слайд

**Добрый день**, нас зовут **Литвинов** Александр и **Водяной** Алексей, и мы бы **хотели рассказать** Вам о важности **использования** такого понятия как **codе-style**

**2 Слайд**

**Для чего же** и зачем нам необходим **codу-style**

Конечно, самое важное это **воспитание технологической культуры**

Это наша цель**.**

Чтобы достичь указанную цель необходимо решить следующие **задачи**

Выявить **суть проблемы**;

Провести **анализ** проблемы;

Найти возможные **способы решения** проблемы;

Подвести **ито**г.

3 слайд

Что же такое **codу-style.**

Рассмотрим на примере **обычного текста**

«Мы открываем страницу, изучаем ее, декомпозируем на блоки, идентифицируем разные части – легко отделяем свойства от функций, уровни изоляции методов и свойств, находим константы и так далее.»

Мы **идентифицируем весть текст** по **блокам**, и поэтому важным аспектом стилизации кода или текста является приведение всего к одинаковому виду.

Это **ключевой фактор, который характеризует читабельность** кода.

4 слайд

Почему это важно? Потому что большую часть своего рабочего времени **программист читает код.** Какой-то прямой корреляции не обнаружено, по большей части это зависит от квалификации, языка (например, Python построен на отступах, и неправильно структурированный код просто не будет работать), качества кода и т.д. Но от **50% времени уходит на чтение своего и чужого кода.** Если код **сложный,** то показатель может достигать 75%, а если код совсем **плох**, то 95% времени может уходить на чтение и попытку понять, куда нужно добавить одну строчку, чтобы поправить какой-то недочет. А теперь вопрос. Что вы сделаете, если увидите, что ваш код тратит **75% времени** на чтение диска или на выделение **памяти двусвязным спискам**? Программист постарается **оптимизировать** этот процесс, попробует поменять алгоритм, применить более эффективную структуру хранения и т.д. И можно было бы сказать мол, ну это **же мозг**, он значительно **мощнее любого компьютера** и может хранить около петабайта данных.

5 слайд

Можно подойти к **решению** этой проблемы также, как мы подходим к проблемам обучения **нейронных сетей**.   
  
Есть **2 способа** ускорить чтение кода:  
  
Постоянно **наращивать базу знаний** разработчика по тому, как может выглядеть код. Это значит непрерывное обучение, которое постоянно наращивает мощность сети.

Привести весь **код** к одному стандарту, задать тот самый **code style**

Разумеется, довольно очевидно, что первый способ – **очень трудозатратный**. В этом случае программисту нужно постоянно читать репозитории. А учитывая текучку кадров, появление новых технологий и практик, это становится практически невозможно. Методом исключения, **остается стилизация кода**, которая снизит аффект на декомпозицию и оставит только аффект на бизнес логику.

6 слайд

**code style** это форма социального контракта внутри вашего community в компании или обществе. Вы **договариваетесь о том**, как писать код, с учетом специфики бизнеса, образования, самого кода, архитектуры и планов по изменению кодовой базы. Эта договоренность означает, что соблюдают ее все. Командная работа подразумевает не столько работу вместе, сколько работу в интересах команды. Это не коворкинг, это общая цель к которой идут все. Как следствие, одной из целей обозначается **чистота кода как инструмент дальнейшего развития**.

7 слайд

Код читается напрямую и легко. Не нужно особо разбираться в бизнес логике, чтобы понять функциональную нагрузку при таком нейминге. В этом примере мы просто берем весь репозиторий, потом используем builder, чтобы создать коллекцию, потом применяем фильтр. Не нужно даже вникать внутрь кода, чтобы понять, что происходит. Как чтение книги по заголовкам.

8 слайд

Хоть это уже и должно быть очевидно, но нужно как-то подчеркнуть основные моменты.   
Code style:  
  
1. **Обеспечивает линейное развитие проекта и не влияет на объем кодовой базы.** Если вы обеспечили историческое написание понятного кода, то, сколько бы не приходило и не уходило разработчиков, вы всегда имеете **равное качество кода**, что позволяет проекту динамично расти вне зависимости от его объема.  
  
2. **Значительно ускоряет процесс адаптации новых программистов.** Если ваш код написан понятно, то даже **новый специалист** очень быстро разберется с определением блоков и начнет приносить пользу. Есть такое понятие, как **точка самоокупаемости** сотрудника. Это точка, с которой сотрудник начинает приносить только пользу. А если код написан понятно, новым сотрудникам не нужно разбираться в бизнес-логике, ему достаточно научится читать ваш код. И чем быстрее он это сделает, тем **быстрее перестанет** задавать тонны **вопросов** остальным специалистам, отнимая у них время. А время специалистов, которые прошли точку самоокупаемости, значительно дороже для команды и компании с точки зрения потенциальной ценности приносимой продукту.

Это же относится **к обучению**, чем понятнее ваш код, тем проще обучить ему ученика. Проще обяснить, меньше вопросов  
  
3. **Убирает зависимость от частностей.** Не нужно будет постоянно спотыкаться о чью-то **оригинальность** и **специфическое оформление**. Представьте реакцию **эксперта**, который проверяет вашу **олимпиаду, ЕГЭ или иной проект**, а если вас таких оригинальных за сегодня было **больше 100**  
4. **Минимизирует эффект ментального блокера при изучении нового кода.** Ваш мозг меньше сопротивляется, т.к. не нужно вникать в чужую стилистику. Ментальных ресурсов на чтение понятного кода нужно намного меньше.  
  
5. **Минимизирует репутационные потери.** Очень скоро, после прибытия в новую компанию, **программист** начнет **делиться впечатлениями** с бывшими коллегами и друзьями. И он **либо** скажет, что тут все **круто**, либо выделит **негативные моменты** в работе с кодом. В каком-то смысле тут получается HR-бонус: если вы хотите, чтобы с вами работали крутые программисты, делайте хороший проект.

Не секрет, что частой причиной ухода становится усталость от постоянного перепиливания некачественной кодовой базы.  
  
6. **Формирует и воспитывает культуру разработки.** Задача программиста лежит на более низком уровне, чем будущее всей компании, но важно донести понимание, что **понятность и читаемость** кода сейчас влияет на **динамику дальнейшей разрабо**тки. Если код трудночитаем и не стандартизирован, с болью и страданиями поддается рефакторингу и масштабированию, то с ростом кодовой базы проекта, будет падать скорость разработки. Чем больше **некачественного кода**, тем **сложнее писать новый**, тем медленнее развивается продукт, тем сложнее компании наращивать обороты и тем сложнее ей платить вам больше денюжек, потому что много денюжек уходит на то, чтобы обеспечивать жизненный цикл проекта все новыми и новыми сотрудниками, который основное время онбоардинга тратят не на то, чтобы приносить пользу компании, а на классификацию наркотиков, под которыми этот код был написан.

9 слайд

Вы можете видеть **источники** из которых мы брали информацию, можете с ней ознакомиться.

Так же: Мы бы хотели выразить свою **благодарность** нашему преподавателю Илье Рудольфовичу Дединскому.

На этом у нас все! Спасибо за уделенное внимание,

Мы готовы ответить на ваши вопросы, если они у вас конечно есть.