

АРТЕЛЬ

семейство языков программирования

ООО «Незабудка Софтвер»

КТО МЫ

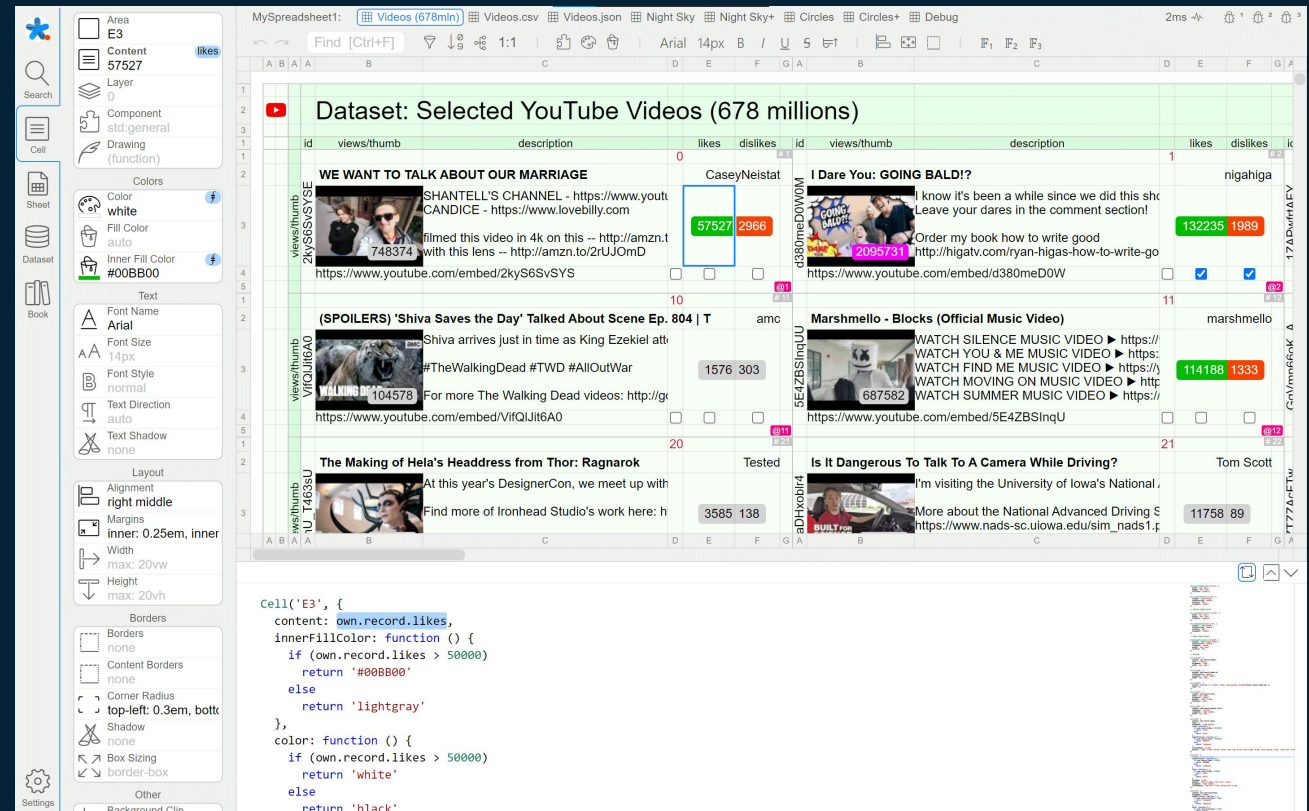
- Выпускники БГУИР разных лет (МРТИ до 1993 года)
 - Факультет КСиС, кафедра ПОИТ
- Сотрудники компании «Незабудка Софтвер»
 - Резидент Парка высоких технологий (ПВТ)
- Преподаём в БГУИР по совместительству
- В настоящее время реализуем проект электронных таблиц для больших данных
- Из больших проектов в прошлом делали САПР микроэлектроники
- Авторы патентов в области распределенных баз данных и поиска по текстовым шаблонам
- Публиковали книги по программированию в среде Delphi и C++ Builder



Nezaboodka*

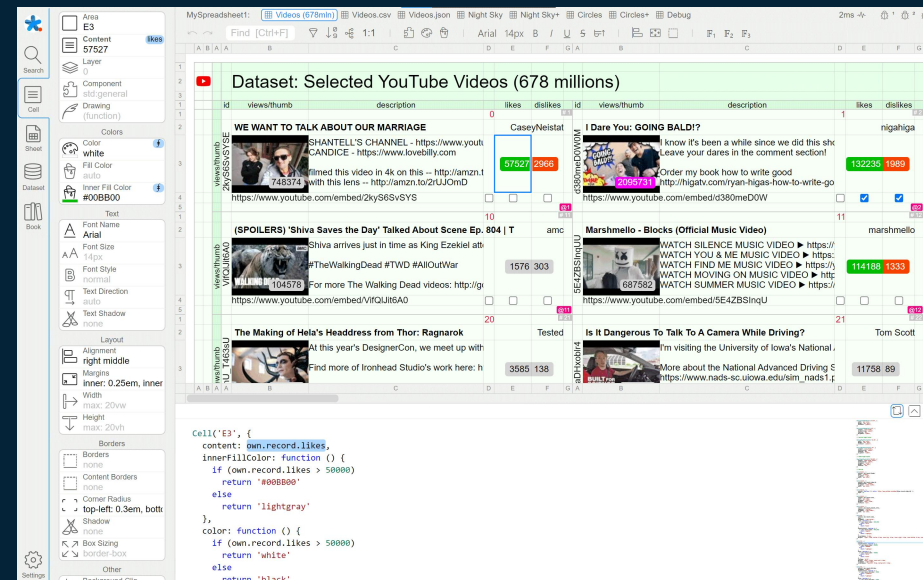
- 2019 год. Проект Бусел (англ. Boosel) — таблицы для больших данных

- Серверная часть:
 - Веб-сервер, .NET;
 - Незабудка — объектная распределённая СУБД с поддержкой ACID-транзакций.
- Требовался язык автоматизации:
 - пользовательского интерфейса;
 - запросов на стороне СУБД;
 - более мощный, чем язык формул Excel с функциями МИН, МАКС и др.



История (2/2)

- Встроили язык Python.
 - Для платформы .NET имелась реализация (Iron Python).
 - Встроили как Web Assembly на стороне браузера.
 - Всё работало невыносимо медленно из-за взаимодействия объектов Python и JavaScript.
 - Если пользовательский интерфейс на русском, то почему автоматизация на английском?
- Решили создать новый язык прикладного программирования с русской и английской лексикой.
 - Это же не очень долго, правда?
 - Первое внедрение — Бусел.
 - Сами «едим свою стряпню».
- Назвали — **Артель**
 - Слово русского происхождения
 - На английском и языках с латиницей — **Artel**:
 - Asynchronous, Reactive, Transactional, Extensible Language
 - — Но артель женского рода. — А это не важно



Постановка задачи

- Русская и английская лексики
 - Как насчёт китайского или арабского (написание справа налево)?
- Возможность решения реальных прикладных задач:
 - Веб-приложения для браузера с пользовательским интерфейсом
 - Веб-службы, серверы приложений, интегрированные среды и редакторы
- Отказ от своей платформы
 - Популярные платформы являются бесплатными и открытыми
 - Богатейший набор готовых бесплатных библиотек
 - Не нужно самим разрабатывать сборщик мусора.
- Целевые платформы:
 - JavaScript в веб-браузере и NodeJS
 - .NET
 - Java
- Возможность писать на одном языке для всех целевых платформ
 - Доступ ко всем возможностям целевой платформы при необходимости
 - Использование программных библиотек на русском, английском и других языках
- Интеграция с существующими репозиториями и менеджерами пакетов
 - NPM, NuGet, др.
- Пригодность для обучения программированию на родном языке



Требования к языку (1/2)

- Переводы имён на разные естественные языки
 - Имена с дефисами, нормализация имён для совместимости с целевой платформой
- Модульное программирование
 - Единица компиляции — пакет — папка с исходными файлами
 - Пакет создаёт пространство имён
 - Возможность подключения/использования/импорта пакета целиком
- Объектно-ориентированное программирование
 - Взаимодействие с программами на языках C#, Java, TypeScript
 - Решение проблемы разных принципов совместимости интерфейсов в .NET/Java и TypeScript
 - Возможность написания независимого от платформы кода
- Обобщённое программирование
 - Обобщённые типы коллекций: Массив, Список, Множество, Словарь
- Функциональное программирование
 - Функциональный тип, безымянные функции, замыкания
- Реактивное транзакционное программирование
 - Наблюдаемые переменные (сигналы)
 - Реактивные функции и транзакции



Требования к языку (2/2)

- Безопасность
 - Отказ от значения «пусто» (null, undefined) для ссылочных типов
 - Союзные типы (объединение множеств значений разных типов)
 - Анализ путей выполнения программы
 - Контроль инициализации переменных до использования
 - Попутная типизация, сужение типов
 - Исчерпывающие проверки вариантов
- Переменные
 - изменяемые
 - неизменяемые — однократно инициализируемые
 - с хранимым значением
 - вычисляемые
- Функции
 - Именованные параметры
 - Отказ от var-параметров
 - Возврат нескольких значений из функции
 - Асинхронные функции (задачи)



Терминология на русском

- Как назвать логический тип?
 - Логическое, Булево, Да-Нет
- Как назвать литералы для значений:
истина/ложь, истинно/ложно, да/нет
 - Мы не знаем истину. В программе нет лжи.
 - Высказывание в один момент времени истинно, а в другой момент ложно.
 - Литералы не нужны в выражениях:
если условие-соблюдено тогда ...
 - Литералы используются в командах присваивания и в редакторах свойств:
конец-игры = да
- Как назвать тип вещественных чисел?
 - Вещественное, Дробное, Число
 - Тип Целое не вызывает вопросов
- Как правильно: ключевые слова или служебные слова?
- Как правильно: операторы, инструкции или команды?
- Как лучше: сокращать имена или не сокращать?

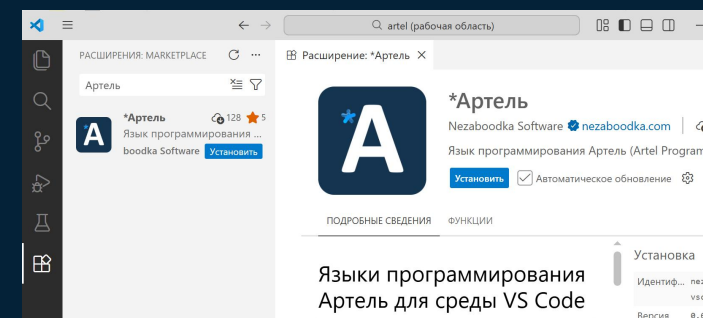
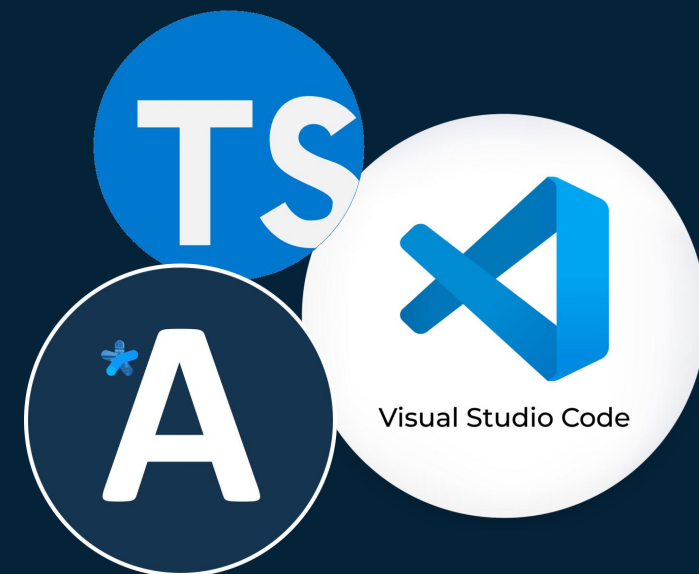


На какие языки опирались?

- Языки, с которыми имеется большой профессиональный и педагогический опыт:
 - Object Pascal
 - C++, C#, Java, TypeScript
 - Python
- Языки, которые исследовали, но не использовали профессионально:
 - Oberon, Eiffel
 - Kotlin, Swift, Go, Rust, Dart, Carbon.
- Языки с русской лексикой:
 - 1С:Элемент, Тривиль.

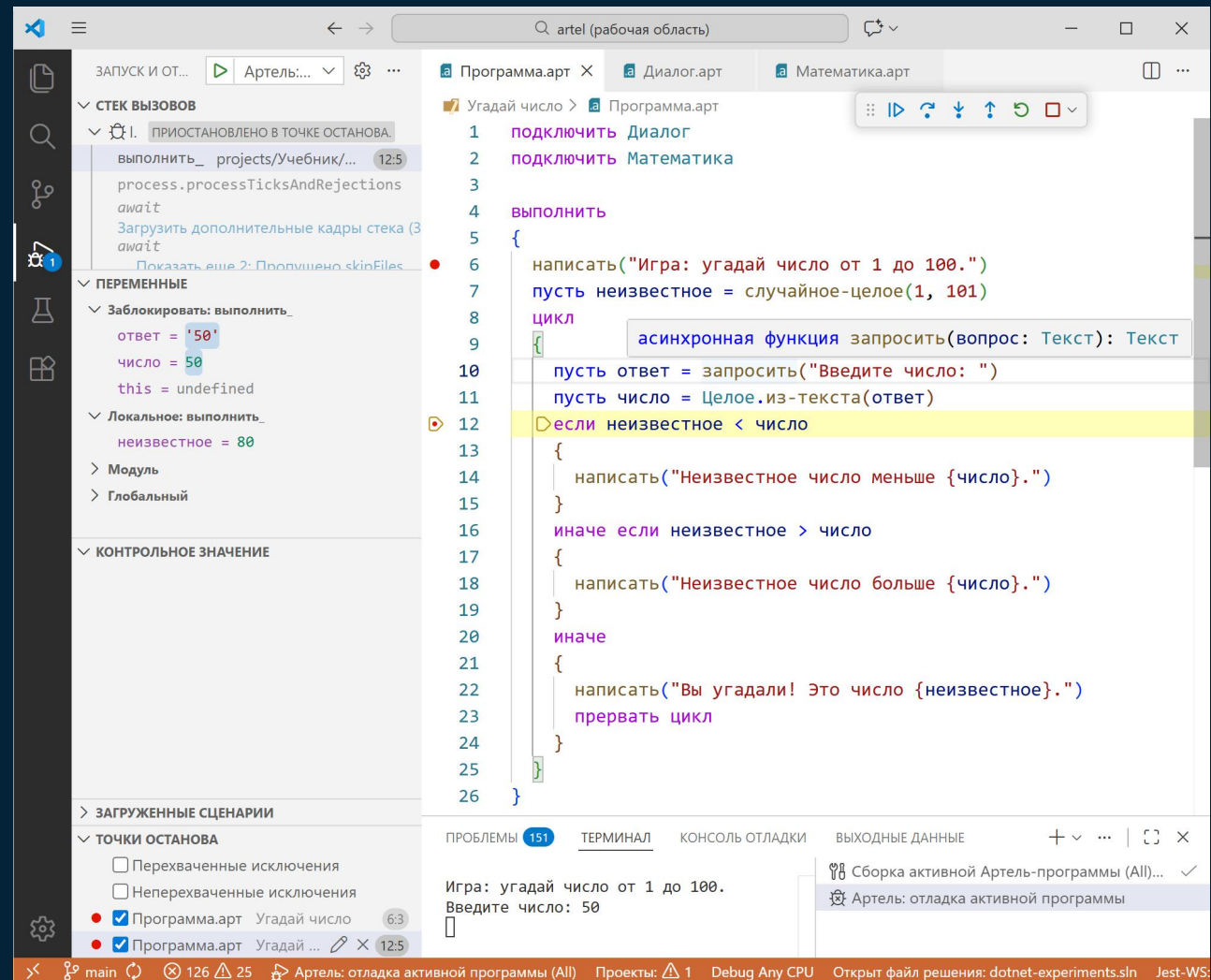
Какой нам нужен компилятор

- Программировать на языке без среды невозможно
 - Будем ли разрабатывать свою среду? **Нет**.
 - В какой среде программируем сами? **VSCode**.
 - Она бесплатная, открытая, с поддержкой русского языка? **Да**.
 - Её редактор кода можно встроить в Бусел? **Да, MonacoEditor**.
 - Что важнее всего программисту при написании и чтении кода? **Навигация**.
- На каком языке разрабатывать?
 - На языке первой целевой платформы — **TypeScript**.
- Изучали код открытых компиляторов: TypeScript, C#, Kotlin, Dart и др.
- Компилятор для среды сильно отличается от пакетного компилятора.
 - Работа с конкретным синтаксическим деревом, сохранение всех лексем
 - Изменение даже одного символа текста — перезапуск анализа
 - «Ленивый» анализ из любой точки дерева (куда смотрит указатель)
 - Асинхронный анализ, возможность быстрой его отмены
 - Первым делом — компилятор для среды, пакетный компилятор — **потом**.

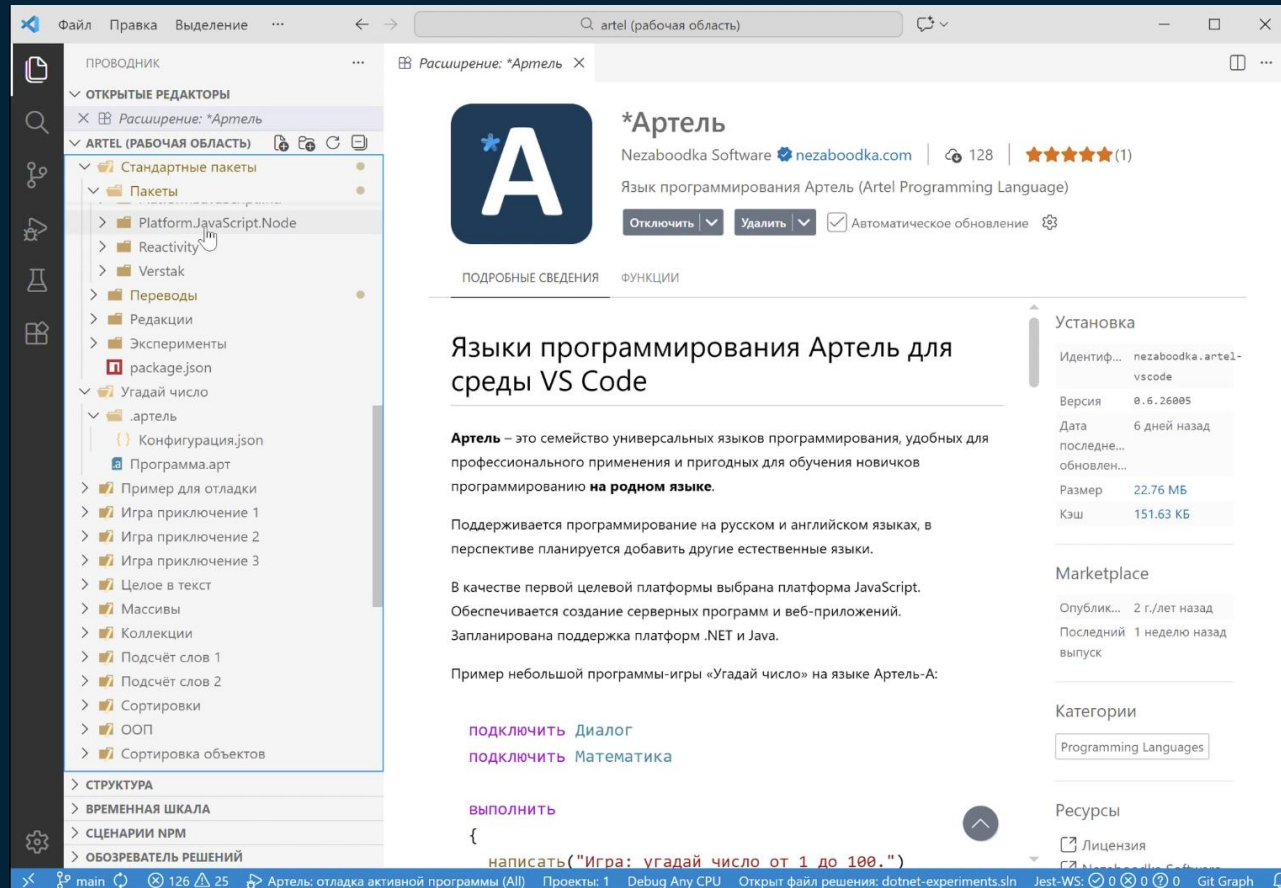


Этапы создания компилятора

1. Расширение к VSCode с анализом и подсветкой синтаксиса
 - Синтаксический анализ с восстановлением после ошибок
 - Связывание имён, определение типов выражений, показ ошибок
2. Простой генератор кода
3. Простой семантический анализ
 - Проверка совместимости типов
 - Выведение типа переменной на основе начального значения
 - Навигация по коду, поиск ссылок, подсказки, показ заголовков вызываемых функций, авто-дополнение имён
 - Переводы имён: русский, английский
4. Интеграция с платформами JavaScript и NodeJS
 - Интерфейсные и реализующие пакеты
 - Интеграция компилятора TypeScript, импорт библиотек из D.TS
5. Отладка
 - Поддержка карт исходных текстов и русских имён
 - Компилятор командной строки в репозитории NPM
6. Сложный семантический анализ
 - Перегруженные функции
 - Анализ потока управления: сужение типов, контроль инициализации переменных до их использования
 - Выведение аргументов типа обобщённых функций
 - Переводы текстов (локализуемый текст)



Расширение к VSCode. Игра «Угадай число»



Посмотреть пример «Угадай число» в редакторе VSCode по [ссылке](#)

Разные задачи — разные требования. Диалекты А и М

Артикул-А

- Более привычен программистам на языках TypeScript, Kotlin, C#, Java, Swift.
- Легче переключаться на языки с английской лексикой и обратно.

Артикул-М

- Меньше понятий, служебных слов и знаков операций.
- Ориентирован на встраивание и скрипты.

```
класс Лекция // по умолчанию нерасширяемый
{
  тема: Текст // неизменяемая после создания
  перем время: ДатаВремя

  создание(тема: Текст, время: ДатаВремя)
  {
    объект.тема = тема; объект.время = время
  }

  расширенная функция в-текст(): Текст
  {
    вернуть "{тема}, {время}"
  }
}
```

А

```
тип Лекция // по умолчанию расширяемый
пер тема: Текст
пер время: Дата-Время

операция создание(тема: Текст, время: Дата-Время)
  объект.тема = тема, объект.время = время
  всё

#замена-реализации // модификатор объявления
операция в-текст(): Текст
  возврат "{тема}, {время}"
  всё
всё
```

М

Состояние проекта

- Расширение к VSCode и Eclipse Theia
 - Подсветка, навигация, подсказки, рефакторинг и др.
 - Отладка
- Пакетный компилятор как NPM-пакет в пространстве @artel.
 - Управление пакетами через NPM и их публикация в репозитории NPMJS
- Целевая платформа JavaScript в браузере и NodeJS на стороне сервера
 - Перевод на русский значительной части программного интерфейса веб-браузера
- Документация
 - Обзоры, руководство, учебник
- Поиск площадки для Git-зеркала, чистка исходников
 - Лицензия Apache 2.0