ЛЕКЦИЯ

«ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ»

ВОПРОСЫ ЛЕКЦИИ

1. Эволюция методов программирования.
2. Основные этапы технологии программирования.
3. Алгоритмы и программы.
4. Популярные языки программирования.

ЛИТЕРАТУРА:

1. Трофимов, В. В. Алгоритмизация и программирование: учебник для академического бакалавриата / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2019. — 137 с. — (Бакалавр. Академический курс. Модуль). — ISBN 978-5-534-07834-3. — Текст : электронный // ЭБС Юрайт [сайт]. с. 31-36 — URL: https://www.biblio-online.ru/bcode/423824/p.31 (дата обращения: 10.01.2020).

2. Зыков, С. В. Программирование: учебник и практикум для академического бакалавриата / С. В. Зыков. — Москва : Издательство Юрайт, 2019. — 320 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-02444-9. — Текст : электронный // ЭБС Юрайт [сайт]. с. 2 — URL: https://www.biblio-online.ru/bcode/433432/p.2 (дата обращения: 10.01.2020).

1. Эволюция методов программирования

Давайте рассмотрим для начала несколько важных понятий. Начнем с главного. Что такое "язык программирования"? У данного понятия есть несколько определений. Вот некоторые из них:

1. Язык программирования (ЯП) − это формальная знаковая система, предназначенная для описания команд (инструкций) и данных, которые могут быть обработаны электронно-вычислительной машиной (ЭВМ).

2. Язык программирования – это набор правил, которые определяют, как написанная компьютерная программа выглядит и что компьютер может сделать под ее управлением.

3. Язык программирования − формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

Как было сказано в определении под номер три, языки программирования являются формальными, или искусственными языками. Они обеспечивают взаимодействие пользователя и ЭВМ.

Формальный язык определён формальной грамматикой. А. Н. Хомский предложил четыре типа формальных грамматик: неограниченные, контекстно-зависимые, контекстно-свободные и регулярные. Языки программирования определяются контекстно-свободными грамматиками при условии, что символами алфавита являются токены (объекты, создающиеся из лексем[1] в процессе лексического анализа), образованные по правилам регулярной грамматики.

Естественные языки (ЕЯ) определяются грамматиками общего вида и в этом смысле отличаются от ЯП, хотя языки программирования высокого уровня внешне похожи на ЕЯ. Как и все языки, ЯП имеют собственный алфавит, синтаксис и семантику.

Алфавит − это конечный набор символов для конкретного языка программирования.

[1] Лексема - минимальная смысловая единица для языка программирования (например: константа, ключевые слова и т.п.). В прикладном программировании токены и лексемы могут не различаться.

Синтаксис определяет правила образования токенов и правила формирования последовательность токенов.

Семантика − это формальное содержание (смысл) последовательности токенов.

Важно понимать, что ЯП характеризуются синтаксической однозначностью (например, в них нельзя менять местами определённые слова) и ограниченностью (имеют строго определённый набор слов и символов). Впрочем, мы уже немного углубились в теорию, давайте начнем по порядку.

1.1 Неструктурированное программирование

Технологией (методом) программирования называется совокупность методов и средств, используемых в процессе разработки программного обеспечения (ПО).

До середины 1960-х гг. преимущественно использовалась неструктурированная, «стихийная» технология программирования. Структура первых простейших программ состояла из программы, написанной на машинном языке (в двоичных или шестнадцатеричных кодах) и обрабатываемых ею данных.

Появление машинно-ориентированных языков (ассемблеров) позволило программистам вместо кодов использовать мнемонические обозначения кодов операций и символические имена данных. Программы стали «читаемыми».

Появление языков программирования высокого уровня (FORTRAN, ALGOL) позволило снизить уровень детализации операций. Большим достижением этих языков стала возможность использования подпрограмм. Теперь структура программы состояла из основной программы, области глобальных данных и набора подпрограмм. Недостаток такой структуры — возрастание вероятности искажения части глобальных данных какой-либо подпрограммой при увеличении количества подпрограмм. Для сокращения таких ошибок было предложено размещать в подпрограммах локальные

данные. Появилась возможность совместной разработки ПО несколькими программистами.

В 1960-х гг. разразился «кризис программирования». Причина — несовершенство неструктурированного программирования. Использовался метод программирования «снизу-вверх», т.е. сначала разрабатывались простые подпрограммы, а затем строилась сложная программа путем их сборки. При сборке программы появлялось большое количество ошибок согласования, а при их исправлении появлялись новые ошибки.

1.2 Процедурное и модульное программирование

В результате исследовательских работ 1960—1970-х гг. была разработана технология процедурного (или структурного, модульного) программирования. Процедурное программирование основано на модели построения программы как иерархии процедур.

Для изучения процедурного программирования в 1971г. Н. Виртом был создан язык программирования Pascal, нашедший большое применение в университетах. В конце 1970-х гг. был создан «классический» язык С Б. Кернигана и Д. Ритчи. На этом языке были написаны фактически все новые операционные системы и системные программные продукты.

Технология процедурного программирования основана на использовании следующих методов (приемов) программирования:

1) метод декомпозиции (нисходящего проектирования), т.е. разделение программы на процедуры простейшей структуры и представление программы в виде иерархии процедур;

2) метод модульной организации, т.е. группировка процедур и обрабатываемых ими данных в модули, которые программируются и компилируются отдельно.

3) метод структурного программирования процедур, т.е. разделение процедур на вложенные блоки, использование операторов ветвления и циклов, форматирование текста процедуры.

1.3 Объектно-ориентированное программирование (ООП)

Применение информационных систем в экономике и управлении привело к появлению больших по объему программ. Область применения программного обеспечения постоянно расширялась, процессы управления, подлежащие автоматизации, усложнялись. Проблема сложности программного обеспечения решалась путем дробления программы на отдельные процедуры и уменьшения их размера для удобства работы и повышения читабельности программы. При таком подходе трудно описать реальные объекты предметной области во всем их многообразии, их поведение и взаимосвязи между ними.

В начале 1980-х гг. Б. Страуструп разработал язык C++, ставший первым промышленно используемым языком, использующим объектно-ориентированный подхода к программированию.

В 1991 г. нидерландским программистом Гвидо ван Россумом был разработан язык программирования Python, включающий в себя как процедурные, так и объектно-ориентированные возможности.

В 1995 г. фирмой Sun Microsystems был разработан на основе языков C и C++ новый язык Java, используемый для создания интерактивных Web-страниц и в разработке приложений на базе Internet, а также для реализации ПО устройств, взаимодействующих по сети.

Ключевым понятием ООП является понятие объекта. Объекты являются программными компонентами, моделирующими элементы реального мира.

Основными принципами ООП являются:

1)Абстракция данных (выделение существенных признаков объекта, отличающих его от других объектов);

2)Инкапсуляция (каждый объект полностью описывается совокупностью своих свойств и методов);

3)Наследование (позволяет создавать новые классы на основе существующих);

4)Полиморфизм означает способность объектов классов, связанных наследованием, реагировать различным образом на одно и то же сообщение.

Итак, объектно-ориентированный подход к программированию имеет ряд существенных преимуществ перед структурным подходом:

1. программирование в понятиях, максимально приближенных к предметной области;
2. многократное использование написанного кода;
3. сокращение времени разработки и отладки программ.

Однако следует отметить и ряд недостатков объектно-ориентированного подхода:

1. Значительная глубина абстракции может привести к созданию «многослойных» приложений, где выполнение объектом требуемого действия сводится к множеству обращений к объектам более низкого уровня, что сказывается на производительности системы.
2. Инкапсуляция снижает скорость доступа к данным.
3. Код, относящийся к классам-потомкам, может находиться не только в этих классах, но и в их классах-предках. Это приводит к снижению скорости трансляции и выполнения программы.
4. Обеспечение полиморфизма приводит к необходимости связывать методы, вызываемые программой в процессе исполнения программы, на что тратится дополнительное время.

1.4 Декларативное программирование

Рассмотренные выше подходы к программированию реализуются в рамках так называемого императивного стиля программирования. Этот стиль предполагает, что программа представляет собой последовательность команд (инструкций), выполняя которые компьютер решает поставленную задачу. Другими словами, при этом подходе программа должна объяснить компьютеру, как решать задачу.

Помимо императивного, широкое распространение получил декларативный стиль программирования. В рамках данного стиля программа представляет собой совокупность утверждений, описывающих фрагмент предметной области или сложившуюся ситуацию. Таким образом, описывается результат, а не методы его достижения. Человек описывает решаемую задачу, а поиском решения занимается система программирования.

Декларативное программирование может применяться во многих областях человеческой деятельности, к которым относятся:

1) создание систем искусственного интеллекта;
2) автоматическое доказательство теорем;
3) разработка экспертных систем и оболочек экспертных систем;
4) создание систем поддержки принятия решений;
5) разработка систем обработки естественного языка;
6) построение планов действий роботов.

Применение декларативного стиля в этих областях позволяет достичь значительно большей скорости разработки приложений, уменьшить размер исходного кода, создать более понятные, по сравнению с императивными языками, программы. Этот подход существенно проще и прозрачнее формализуется математическими средствами, поэтому программы легче тестировать и верифицировать.

К декларативному стилю в первую очередь относятся функциональный (Lisp) и логический подходы (Prolog) к программированию.

В функциональном языке существует некоторое множество базовых функций, на основе которых строятся все другие функции, как композиции базовых.

Логическое программирование возникло как упрощение функционального программирования для математиков и лингвистов, решающих задачи символьной обработки. Поиск ответа на запрос заключается в попытке логического вывода запроса на основании фактов и правил, имеющихся в базе.

1.5 Компонентные технологии

Опыт использования технологии объектно-ориентированного программирования выявил проблемы совместимости при компоновке объектов. Попытка изменения одного объекта приводила к необходимости перекомпиляции всего кода проекта. В результате разработка ПО была ограничена использованием только одного языка программирования и только одного компилятора.

Изменения в требованиях к информационному сервису, а также появление новых представлений о требуемых данных и функциях привели к усложнению ПО. Это потребовало создания автоматизированных технологий разработки и сопровождения ПО.

Потребность в разработке технологий, устраняющих указанные недостатки и убыстряющих написание программ, привела к появлению компонентных технологий программирования и CASE-технологий.

При компонентном подходе модель построения программы представляет собой совокупность отдельных двоичных объектов-компонентов — физически отдельно существующих частей программы, взаимодействующих между собой через стандартные двоичные интерфейсы.

Объекты-компоненты можно использовать в любом языке программирования, применять в одном или в разных процессах, на одном или на разных компьютерах.

Технологии программирования, использующие компонентный подход, разработаны на базе технологии COM (компонентная модель объектов) и на базе технологии создания распределенных объектов CORBA.

Технология COM — это модель взаимодействия типа «клиент — сервер». Клиент — это программа или объект, использующий другой объект. Клиент подсоединяется к объекту через интерфейс. Сервер — это местоположение объектов COM, подключаемых к приложению-клиенту.

Технология COM фирмы Microsoft и ее распределенная версия DCOM явились основой для разработки компонентных технологий программирования.

Технология CORBA использует принципы, аналогичные технологии COM. В ней также применяется модель взаимодействия «клиент — сервер», но организация взаимодействия производится с помощью специального посредника. Таким посредником выступает брокер запросов к объектам. Наиболее широко и достаточно долго использовался продукт VisiBroker фирмы Borland. Технологию можно применять для разработки распределенного ПО в разнородной вычислительной среде.

CASE-средства представляют собой инструменты разработки, автоматизирующие процессы создания и сопровождения ПО, включая этапы анализа и проектирования ПО, генерацию кода, тестирования, документирования и другие процессы.

Большинство CASE-технологий основано на методологии процедурного и объектно-ориентированного проектирования. В настоящий момент на рынке программного обеспечения насчитывается более 300 различных CASE-средств. Наиболее известными являются CAERwinProcess Modeler (ранее BPwin), CA ERwin Data Modeler (ранее ERwin) и т.д.

1.6 Перспективы развития технологий программирования

Развитие информационных технологий сопровождается появлением новых и совершенствованием существующих подходов к программированию.

По мнению многих исследователей, развитие языков программирования в ближайшее время будет двигаться в направлении все большей абстракции, изменения уровня детализации и наибольшего упрощения. Это приведет к повышению надежности процесса создания программ и уменьшению количества допускаемых разработчиками ошибок.

2. Основные этапы технологии программирования

Для решения задачи на компьютере требуется написать программу. Программа на языке высокого уровня состоит из исполняемых операторов и операторов описания.

Исполняемый оператор задает законченное действие, выполняемое над данными. Примеры операторов: вывод на экран, занесение числа в память, выход из программы.

Оператор описания, как и следует из его названия, описывает данные, над которыми в программе выполняются действия. Примером описания (на естественном языке) может служить предложение «В памяти следует отвести место для хранения целого числа, и это место мы будем обозначать А».

Описания должны предшествовать операторам, в которых используются соответствующие данные. Операторы программы исполняются последовательно, один за другим, если явным образом не задан иной порядок.

Для того чтобы выполнить программу, написанную на языке программирования высокого уровня, требуется перевести ее на язык, понятный процессору — в машинные коды. Этим занимаются специальные программы, которые называются языковыми процессорами, или трансляторами. Различают два вида языковых процессоров: интерпретаторы и компиляторы.

Интерпретатор — это программа, которая получает исходную программу на языке высокого уровня и по мере распознавания его операторов выполняет описываемые ими действия.

Транслятор — это программа, которая получает на вход исходную программу и формирует на выходе программу на объектном языке программирования. В частном случае объектным кодом может служить машинный язык, и в этом случае полученную на выходе транслятора программу можно сразу же выполнить на компьютере.

В общем случае объектный язык необязательно должен быть машинным или близким к нему (автокодом). В качестве объектного языка может служить и некоторый промежуточный язык. Для промежуточного языка может быть использован другой компилятор или интерпретатор — с промежуточного языка

на машинный. Схему трансляции, когда исходная программа переводится на промежуточный язык, который затем интерпретируется, называют гибридной. Такая схема используется в языках Java и C #. Транслятор, использующий в качестве входного языка близкий к машинному, традиционно называют ассемблером.

Компилятор работает следующим образом. Получив на вход исходный текст программы, компилятор выделяет из него лексемы, а затем на основе грамматики языка распознает выражения и операторы, построенные из этих лексем. При этом компилятор выявляет синтаксические ошибки и в случае их отсутствия строит объектный модуль.

Каждый оператор языка переводится компилятором в последовательность машинных команд, которая может быть весьма длинной, поэтому языки типа Pascal и C ++ и называются языками высокого уровня.

Кроме того, компилятор планирует размещение данных в оперативной памяти в соответствии с описаниями величин, используемых в программе. Попутно он ищет синтаксические ошибки, т.е. ошибки записи операторов.

Компоновщик, или редактор связей, формирует исполняемый модуль программы, подключая к объектному модулю другие объектные модули, в том числе содержащие функции библиотек, обращение к которым содержится в любой программе (например, для осуществления вывода на экран). Таким образом, при компиляции трансляция и исполнение программы идут последовательно друг за другом. При интерпретации — параллельно. Один раз откомпилированная программа может быть сохранена во внешней памяти и затем исполняться многократно. На компиляцию машинное время тратиться больше не будет. Программа на интерпретируемом языке при каждом выполнении подвергается повторной трансляции. Кроме того, интерпретатор может занимать значительное место в оперативной памяти.

В настоящее время практически любая реализация языка представлена как среда разработки, которая включает в себя:

1) компилятор (или интерпретатор);

2) отладчик — специальную программу, которая облегчает процесс поиска ошибок;

3) встроенный текстовый редактор;

4) специальные средства для просмотра структуры программы;

5) библиотеку готовых модулей, классов, например, для создания пользовательского интерфейса (окна, кнопки и т.д.).

С помощью компьютера можно решать задачи различного характера, например, научно-инженерные, задачи разработки системного программного обеспечения, программ для обучения, для управления производственными процессами и т.д.

Разные задачи требуют различных технологий своей разработки, различающиеся принципами, количеством и составом этапов, областями применения и назначением. Для создания относительно простых программ успешно применяется классический жизненный цикл разработки, когда очередной этап начинается после полного завершения предыдущего. Обычно выделяют следующие этапы:

1) постановка задачи;

2) математическое описание задачи;

3) выбор и обоснование метода решения;

4) выбор структур данных и алгоритмов решения задачи;

5) составление (кодирование) программы;

6) тестирование и отладка программы;

7) анализ результатов.

Некоторые этапы могут отсутствовать, например, в задачах разработки системного программного обеспечения отсутствует математическое описание.

3. Алгоритмы и программы

Термин «алгоритм» происходит от имени узбекского математика Аль-Хорезми (825 г.), который ввел правила выполнения четырех арифметических операций в десятичной системе счисления.

Алгоритмом называют точное предписание, которое задается вычислительному процессу и представляет собой конечную последовательность обычных элементарных действий, четко определяющую процесс преобразования исходных данных в искомый результат.

Перечислим основные свойства алгоритмов:

• определенность (детерминированность) — точность и понятность, обеспечивающие однозначность результата;

• результативность — получение искомого результата через конечное число этапов (шагов);

• дискретность — расчлененность алгоритма на отдельные этапы;

• массовость — пригодность алгоритма для решения всех задач данного типа.

Основными требованиями к алгоритму (как и к программе) являются компактность и минимальное время реализации.

Существуют следующие способы записи алгоритмов:

• псевдокод (формульно-словесный) — содержание последовательных этапов вычислений задается с нумерацией в произвольной форме на естественном языке; это искусственный, неформальный язык, но не язык программирования;

• блок-схемный — изображение структуры алгоритма в виде графического представления этапов процесса обработки данных с помощью специальных геометрических символов (блоков). Основными блоками, используемыми для записи алгоритма, являются начало/конец алгоритма, ввод/вывод данных, процесс, блок выбора решения;

• алгоритмические языки — непосредственная запись алгоритма решения задачи с помощью операторов языка.

Рассмотрим различные способы записи алгоритма для решения следующей задачи: ввести два числа, a и b. Переменной m присвоить значение наибольшего из этих чисел. Значение переменной m вывести. Алгоритм можно записать с помощью псевдокода или блок-схемы.

Запись алгоритма псевдокодом может выглядеть так:

1) ввести два числа: a и b ;

2) если a больше b, переменной m присвоить значение a, иначе b ;

3) вывести значение m.

Блок-схема, содержащая те же этапы, приведена на Рисунке 1.

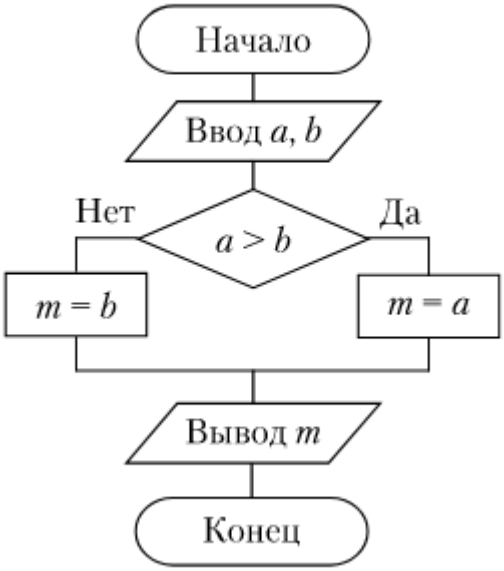

Рисунок 1 — Блок-схема алгоритма

Программа — это упорядоченная последовательность команд компьютера, необходимая для решения конкретной задачи.

Программы не имеют заранее строго регламентированного набора качественных характеристик. Перечислим основные характеристики программ

• алгоритмическая сложность (логика алгоритмов обработки информации);

• глубина проработки, полнота и системность реализованных функций программы;

• требования к операционной системе и техническим средствам со стороны программы;

• удобство освоения и эксплуатации программы.

Критериями качества программы являются:

• мобильность — независимость программы от программных и технических средств обработки данных;

• надежность — устойчивость в работе программы, точность выполнения функций обработки;

• эффективность — оценка расходов вычислительных ресурсов (объемов памяти для эксплуатации программы и т.п.);

• дружественность — обеспечение дружественного интерфейса для работы пользователя;

• модифицируемость — способность к внесению изменений и расширений функций обработки;

• коммуникативность — возможность интеграции с другими программами.

На качество программы влияют следующие факторы:

• маркетинг рынка и спецификация требований к программе — определение состава функций обработки данных программы, выбор пользовательского интерфейса, требования к комплексу технических и программных средств;

• проектирование структурной схемы программы — алгоритмизация процессов обработки данных, разработка структуры программы и базы данных, выбор метода и средств создания программы — технологии программирования;

• программирование (алгоритмизация, тестирование и отладка) — техническая реализация проекта программы, выполняемая с помощью выбранного инструментария технологии программирования

• эксплуатация и сопровождение программы — важный этап, связанный с устранением обнаруженных ошибок;

• распространение программы и завершение жизненного цикла — этап, связанный с постоянной программой маркетинговых мероприятий и заканчивающийся либо отсутствием спроса, либо изменением технической политики разработчика.

4. Популярные языки программирования

На данный момент существует множество языков программирования, но можно назвать только несколько из них, которые пользуются большой популярностью, например: JavaScript, Java и Python.

Python

Язык программирования Python (читается как "пайтон", но обычно в русской версии произносят "питон") - это высокоуровневый язык программирования, который может работать в режиме интерпретатора. Его простота, удобство пользования, относительно простые синтаксис и семантика, возможность использования объектно-ориентированного подхода привлекли множество программистов со всего мира. Он является бесплатным, что также служит весомым преимуществом.

Python подходит для решения большинства повседневных задач, будь то резервное копирование, чтение электронной почти или какое-нибудь игровое приложение. Язык программирования Python почти ничем не ограничен, поэтому также может использовать в крупных проектах. Например, Python интенсивно используется такими IT-гигантами, как Google и Yandex. К тому же простота и универсальность Python делают его одним из лучших языков программирования.

Java

Язык Java, официальной датой создания которого считается 23 мая 1995 г., является объективно-ориентированным языком программирования, разработанным в компании Sun Microsystems и приобретенным впоследствии известной компанией Oracle. Отличительной особенностью Java является возможность работать вне зависимости от архитектуры компьютера на любой виртуальной Java-машине, переводя высокоуровневое представление в байт-

код[2]. Язык Java является неотъемлемой частью масштабных ERP (от англ. enterprise resource planning - планирование ресурсов предприятия) систем.

Три ключевых элемента объединились в технологии языка Java:

- Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.
- Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты (не самостоятельные компоненты программного обеспечения). Апплеты - это программы на Java, которые, как правило, предназначены для того, чтобы загружаться посредством браузера, а затем работать в окне браузера. Они могут использоваться для создания богатых графикой и интерактивными возможностями пользовательских интерфейсов, которые не способны выразить средствами обычного языка разметки HTML. Сервлеты - программы на Java, которые работают на Web-серверах Java или серверах приложений Java. Как и программы CGI, сервлеты могут доставлять Web-службы непосредственно в браузер или действовать как промежуточное ПО, которое связывает браузер с серверными службами.
- Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих

[2] Байт-код Java — набор инструкций, исполняемых виртуальной машиной Java. Каждый код операции байт-кода — один байт. Используются не все 256 возможных значений кодов операций. 51 из них зарезервированы для использования в будущем.

классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

JavaScript

JavaScript был создан за 10 дней автором Бренданом Айком (Brandan Eich), работавшим на Netscape в далёком 1995 году. Первоначальная версия языка была ограничена только браузером Netscape и предлагала узкую функциональность, но со временем он продолжил развиваться частично благодаря сообществу разработчиков, которые не оставляли работу над ним.

JavaScript на сегодняшний день работает не только в разных браузерах, но также на различных устройствах, включая мобильные и настольные компьютеры.

JavaScript превратился из примитивного языка программирования в один из наиболее популярных инструментов в арсенале веб-разработчика.

Достоинства JavaScript:

- Его проще изучать, чем другие языки программирования.
- Ошибки проще выявить, а значит и справить.
- Он может привязываться к специальным элементам страниц или события вроде нажатия (click) или наведения мыши (mouseover).
- JS работает в разных браузерах и на разных платформах.
- Можно использовать JavaScript для вариации входных данных и снижения необходимости ручной проверки данных.
- Он делает сайт более интерактивным и привлекательным для посетителей.
- Он быстрее и легче, чем другие языки программирования.

Недостатки этого языка программирования:

1. Уязвим по отношению к эксплойтам (вредоносный код, использующий уязвимости программного продукта).

2. Может быть использован для запуска вредоносного кода на компьютере пользователя.
3. Не всегда поддерживается некоторыми браузерами или устройствами.
4. Фрагменты JS кода могут быть очень большими.
5. Может по разному отображаться на разных устройствах, что приводит к отсутствию целостности.