Министерство сельского хозяйства Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Брянский государственный аграрный университет»

Институт энергетики и природопользования
Кафедра информатики, информационных систем и технологий

ФЕДЬКОВА Н.А.

Современные технологии разработки программного обеспечения

Методическое пособие

Брянская область, 2022

УДК 004.45 (07)
ББК 32.973-018.2
С 56

Современные технологии разработки программного обеспечения: методическое пособие / сост.: Н. А. Федькова. – Брянск: Изд-во Брянский ГАУ, 2022. – 58 с.

Издание окажет помощь студентам направления подготовки 09.04.03 Прикладная информатика при подготовке по дисциплине «Современные технологии разработки программного обеспечения».

Рекомендовано к изданию.

Рецензенты: старший преподаватель кафедры информационных систем и технологий Милютина Елена Михайловна

Содержание

Введение

Пособие направлено на изучение современных методов и средств проектирования информационных систем. Предусматривается изучение CASE-средств, как программного инструмента поддержки проектирования информационных систем (ИС). Курс предусматривает изучение: состава и структуры различных классов экономических ИС как объектов проектирования; современных технологий проектирования ИС и методик обоснования эффективности их применения; содержания стадий и этапов проектирования ИС и их особенностей при использовании различных технологий проектирования; целей и задач проведения предпроектного обследования объектов информатизации; методов моделирования информационных процессов предметной области; классификацию и общие характеристики современных CASE-средств.

Научной основой курса являются методологии системного анализа и моделирования, позволяющие на этапе создания информационной системы решить следующие основные задачи:

- обеспечение требуемой функциональности системы и адаптивности к изменяющимся условиям ее функционирования;
- проектирование реализуемых в системе объектов данных;
- проектирование программ и средств интерфейса (экранных форм, отчетов), которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии реализации проекта, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры, параллельной обработки, распределенной обработки данных и т.п.

Программой курса предусматривается изучение CASE-инструментов поддержки проектирования информационных систем. Практикум дисциплины включает в себя задания для освоения учащимися инструментальных средств разработки и анализа функциональных и информационных моделей деятельности экономических объектов (предприятий и учреждений), являющихся основой проектирования информационных систем.

Дисциплина «Методология и технология проектирования информационных систем» имеет целью ознакомить учащихся с информационными технологиями анализа сложных систем и основанными на международных стандартах методами проектирования информационных систем, обучить студентов принципам построения функциональных и информационных моделей систем, проведению анализа полученных результатов, применению инструментальных средств поддержки проектирования экономических информационных систем.

Интегрированная среда разработки

1. Основные понятия.
2. История.
3. Обзор IDE.
4. Java IDE для быстрой веб-разработки.

1. Основные понятия

Интегрированная среда разработки, ИСР (англ. Integrated development environment — IDE), также единая среда разработки, ЕСР — комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО).

Среда разработки включает в себя:

- текстовый редактор,
- Транслятор (компилятор и/или интерпретатор),
- средства автоматизации сборки,
- отладчик.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для использования при объектно-ориентированной разработке ПО. ИСР обычно предназначены для нескольких языков программирования — такие как IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code::Blocks, Xcode или Microsoft Visual Studio, но есть и IDE для одного определённого языка программирования — как, например, Visual Basic, Delphi, Dev-C++.

Частный случай ИСР — среды визуальной разработки, которые включают в себя возможность наглядного редактирования интерфейса программы.

Использование ИСР для разработки программного обеспечения является прямой противоположностью способу, в котором используются несвязанные инструменты, такие как текстовый редактор, компилятор, и т. п. Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволяет разработчику сделать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. Однако так как ИСР является сложным программным комплексом, то среда разработки сможет качественно ускорить процесс разработки ПО лишь после специального обучения. Для уменьшения барьера вхождения многие достаточно интерактивны, а для облегчения перехода с одной на другую интерфейс у одного производителя максимально близок, вплоть до использования одной ИСР.

ИСР обычно представляет собой единственную программу, в которой проводится вся разработка. Она, как правило, содержит много функций для создания, изменения, компилирования, развертывания и отладки программного

обеспечения. Цель интегрированной среды заключается в том, чтобы объединить различные утилиты в одном модуле, который позволит абстрагироваться от выполнения вспомогательных задач, тем самым позволяя программисту сосредоточиться на решении собственно алгоритмической задачи и избежать потерь времени при выполнении типичных технических действий (например, вызове компилятора). Таким образом, повышается производительность труда разработчика. Также считается, что тесная интеграция задач разработки может далее повысить производительность за счёт возможности введения дополнительных функций на промежуточных этапах работы. Например, ИСР позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках.

Большинство современных ИСР являются графическими. Но первые ИСР использовались ещё до того, как стали широко применяться операционные системы с графическим интерфейсом — они были основаны на текстовом интерфейсе с использованием функциональных и горячих клавиш для вызова различных функций (например, Turbo Pascal, созданный фирмой Borland).

2. История

Первые ИСР были созданы для работы через консоль или терминал, которые сами по себе были новинкой: до того программы создавались на бумаге, вводились в машину с помощью предварительно подготовленных бумажных носителей (перфокарт, перфолент) и т. д.

Dartmouth BASIC был первым языком, который был создан с ИСР, и был также первым, который был разработан для использования в консоли или терминале. Эта ИСР (часть Dartmouth Time Sharing System) управлялась при помощи команд, поэтому существенно отличалась от более поздних, управляемых с помощью меню и горячих клавиш, и тем более графических ИСР, распространённых в XXI веке. Однако она позволяла править исходный код, управлять файлами, компилировать, отлаживать и выполнять программы способом, принципиально подобным современным ИСР.

Maestro I — продукт от Softlab Munich, был первой в мире интегрированной средой разработки для программного обеспечения в 1975 г.[2] и, возможно, мировым лидером в этой рыночной нише в течение 1970-х и 1980-х годов. Он был установлен у 22000 программистов во всем мире. До 1989 года 6000 копий было установлено в Федеративной Республике Германия. Ныне Maestro I принадлежит истории и может быть найден разве что в Музее Информационной технологии в Арлингтоне.

Одной из первых ИСР с возможностью подключения плагинов была Softbench.

3. Обзор IDE

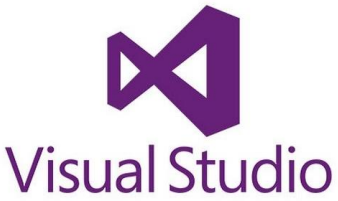
IDE — это не просто текстовый редактор. В то время как текстовые редакторы для кода, такие как Sublime или Atom, предлагают множество удобных функций, таких как подсветка синтаксиса, настраиваемый интерфейс и расши-

ренные средства навигации, они позволяют только писать код. Для создания функционирующих приложений как минимум нужен компилятор и отладчик.

IDE включает в себя эти компоненты, как и ряд других. Некоторые из них поставляются с дополнительными инструментами для автоматизации, тестирования и визуализации процесса разработки. Термин «интегрированная среда разработки» означает, что предоставляется все необходимое для превращения кода в функционирующие приложения.

Ознакомьтесь с приведенным ниже списком функций и недостатков каждой из 10 лучших IDE.

1. Microsoft Visual Studio.



Microsoft Visual Studio – это интегрированная среда разработки, цена которой варьируется от $699 до $2900. Множество версий этой IDE способны создавать все типы программ, начиная от веб-приложений и заканчивая мобильными приложениями, видеоиграми. Эта линейка программного обеспечения включает в себя множество инструментов для тестирования совместимости. Благодаря своей гибкости Visual Studio является отличным инструментом для студентов и профессионалов.

Поддерживаемые языки: Ajax, ASP.NET, DHTML, JavaScript, JScript, Visual Basic, Visual C#, Visual C++, Visual F#, XAML и другие.

Особенности:

– огромная библиотека расширений, которая постоянно увеличивается;
– IntelliSense;
– настраиваемая панель и закрепляемые окна;
– простой рабочий процесс и файловая иерархия;
– статистика мониторинга производительности в режиме реального времени;
– инструменты автоматизации;
– легкий рефакторинг и вставка фрагментов кода;
– поддержка разделенного экрана;
– список ошибок, который упрощает отладку;
– проверка утверждения при развертывании приложений с помощью ClickOnce, Windows Installer или Publish Wizard.

Недостатки: поскольку Visual Studio является супертяжелой IDE, для открытия и запуска приложений требуются значительные ресурсы. Поэтому на некоторых устройствах внесение простых изменений может занять много времени. Для простых задач целесообразно использовать компактный редактор илисредство разработки PHP.

2. NetBeans.

Бесплатная среда разработки с открытым исходным кодом. Подходит для редактирования существующих проектов или создания новых. NetBeans предлагает простой drag-and-drop интерфейс, который поставляется с большим количеством удобных шаблонов проектов. Среда в основном используется для разработки Java приложений, но можно устанавливать пакеты, поддерживающие другие языки.

Поддерживаемые языки программирования: C, C++, C++ 11, Fortan, HTML 5, Java, PHP и другие.

Особенности:

- интуитивный drag-and-drop интерфейс;
- динамические и статические библиотеки;
- интеграция нескольких сессий GNU-отладчика с поддержкой кода;
- возможность осуществлять удаленное развертывание;
- совместимость с платформами Windows, Linux, OS X и Solaris;
- поддержка Qt Toolkit;
- поддержка Fortan и Assembler;
- поддержка целого ряда компиляторов, включая CLang / LLVM, Cygwin, GNU, MinGW и Oracle Solaris Studio.

Недостатки: эта бесплатная среда разработки потребляет много памяти, поэтому может работать медленно на некоторых ПК.

3. PyCharm.

PyCharm разработан командой Jet Brains. Пользователям предоставляется бесплатная версия Community Edition, 30-дневная бесплатная ознакомительная версия Professional Edition и годовая подписка за $213 — $690 на версию Professional Edition. Комплексная поддержка кода и анализ делают PyCharm лучшей IDE для Python-программистов.

Поддерживаемые языки: AngularJS, Coffee Script, CSS, Cython, HTML, JavaScript, Node.js, Python, TypeScript.

Особенности:

- совместимость с операционными системами Windows, Linux и Mac OS;
- поставляется с Django IDE;
- легко интегрируется с Git, Mercurial и SVN;

– настраиваемый интерфейс с эмуляцией VIM;
– отладчики JavaScript, Python и Django;
– поддержка Google App Engine.

Недостатки: пользователи жалуются, что эта среда разработки Python содержит некоторые ошибки, такие как периодически не работающая функция автоматического заполнения, что может доставить определенные неудобства.

4. IntelliJ IDEA.

Еще одна IDE, разработанная Jet Brains. Она предлагает пользователям бесплатную версию Community Edition, 30-дневную бесплатную ознакомительную версию Ultimate Edition и годовую подписку на версию Ultimate Edition за $533 — $693. IntelliJ IDEA поддерживает Java 8 и Java EE 7, обладает обширным инструментарием для разработки мобильных приложений и корпоративных технологий для различных платформ. Если говорить о цене, IntelliJ является прекрасным вариантом из-за огромного списка функций.

Поддерживаемые языки программирования: AngularJS, CoffeeScript, HTML, JavaScript, LESS, Node JS, PHP, Python, Ruby, Sass, TypeScript и другие.

Особенности:

– расширенный редактор баз данных и дизайнер UML;
– поддержка нескольких систем сборки;
– пользовательский интерфейс тестового запуска приложений;
– интеграция с Git;
– поддержка Google App Engine, Grails, GWT, Hibernate, Java EE, OSGi, Play, Spring, Struts и других;
– встроенные средства развертывания и отладки для большинства серверов приложений;
– интеллектуальные текстовые редакторы для HTML, CSS и Java;
– интегрированный контроль версий;
– AIR Mobile с поддержкой Android и iOS.

Недостатки: эта среда разработки JavaScript требует времени и усилий на изучение, поэтому может оказаться не лучшим вариантом для начинающих. В ней есть много сочетаний горячих клавиш, которые нужно просто запомнить. Некоторые пользователи жалуются на неуклюжий интерфейс.

5. Eclipse.

Бесплатный и гибкий редактор с открытым исходным кодом. Он может оказаться полезен, как для новичков, так и для профессионалов. Первоначально создаваемый как среда для Java-разработки сегодня Eclipse имеет широкий диапазон возможностей благодаря большому количеству плагинов и расшире-

ний. Помимо средств отладки и поддержки Git / CVS, стандартная версия Eclipse поставляется с инструментами Java и Plugin Development Tooling. Если вам этого недостаточно, доступно много других пакетов: инструменты для построения диаграмм, моделирования, составления отчетов, тестирования и создания графических интерфейсов. Клиент Marketplace Eclipse открывает пользователям доступ к хранилищу плагинов и информации.

Поддерживаемые языки: C, C++, Java, Perl, PHP, Python, Ruby и другие.

Особенности:

– множество пакетных решений, обеспечивающих многоязычную поддержку;

– улучшения Java IDE, такие как иерархические представления вложенных проектов;

– интерфейс, ориентированный на задачи, включая уведомления в системном треке;

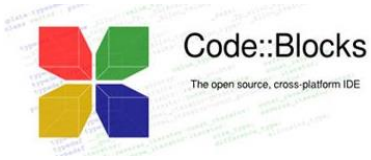
– автоматическое создание отчетов об ошибках;

– параметры инструментария для проектов JEE;

– интеграция с JUnit.

Недостатки: многие параметры этой среды разработки могут запугать новичков. Eclipse не обладает всеми теми функциями, что и IntelliJ IDEA, но является IDE с открытым исходным кодом.

6. Code::Blocks.



Еще один популярный инструмент с открытым исходным кодом. Гибкая IDE, которая стабильно работает на всех платформах, поэтому она отлично подходит для разработчиков, которые часто переключаются между рабочими пространствами. Встроенный фреймворк позволяет настраивать эту IDE под свои потребности.

Поддерживаемые языки: C, C++, Fortran.

Особенности:

– простой интерфейс с вкладками открытых файлов;

– совместимость с Linux, Mac и Windows;

– написана на C++;

– не требует интерпретируемых или проприетарных языков программирования;

– множество встроенных и настраиваемых плагинов;

– поддерживает несколько компиляторов, включая GCC, MSVC ++, clang и другие;

– отладчик с поддержкой контрольных точек;

– текстовый редактор с подсветкой синтаксиса и функцией автоматического заполнения;

– настраиваемые внешние инструменты;

– простые средства управления задачами, идеально подходящие для совместной работы.

Недостатки: относительно компактная среда разработки Си, поэтому она не подходит для крупных проектов. Это отличный инструмент для новичков, но продвинутые программисты могут быть разочарованы ее ограничениями.

7. Aptana Studio 3.

Самая мощная из IDE с открытым исходным кодом. Aptana Studio 3 значительно улучшена по сравнению с предыдущими версиями. Поддерживает большинство спецификаций браузеров. Поэтому пользователи этой IDE могут с ее помощью быстро разрабатывать, тестировать и развертывать веб-приложения.

Поддерживаемые языки: HTML5, CSS3, JavaScript, Ruby, Rails, PHP и Python.

Особенности:

– подсказки для CSS, HTML, JavaScript, PHP и Ruby;

– мастер развертывания с простой настройкой и несколькими протоколами, включая Capistrano, FTP, FTPS и SFTP;

– возможность автоматической установки созданных приложений Ruby и Rails на серверы хостинга;

– интегрированные отладчики для Ruby и Rails и JavaScript;

– интеграция с Git;

– простой доступ к терминалу командной строки с сотнями команд;

– строковые пользовательские команды для расширения возможностей.

Недостатки: есть проблемы со стабильностью, и она работает медленно. Поэтому профессиональные разработчики могут предпочесть более мощную HTML среду разработки.

8. Komodo.

Предлагает бесплатную 21-дневную ознакомительную версию, полная версия стоит $99 – $1615 в зависимости от редакции и лицензии. Komodo поддерживает большинство основных языков программирования. Удобный интерфейс позволяет осуществлять расширенное редактирование, а небольшие полезные функции, такие как проверка синтаксиса и одноступенчатая отладка, делают Komodo одной из самых популярных IDE для веб и мобильной разработки.

Поддерживаемые языки: CSS, Go, JavaScript, HTML, NodeJS, Perl, PHP, Python, Ruby, Tcl и другие.

Особенности:

– настраиваемый многооконный интерфейс;
– интеграция контроля версий для Bazaar, CVS, Git, Mercurial, Perforce и Subversion;
– профилирование кода Python и PHP;
– возможность развертывания в облаке благодаря Stackato PaaS;
– графическая отладка для NodeJS, Perl, PHP, Python, Ruby и Tcl;
– автоматическое заполнение и рефакторинг;
– стабильная производительность на платформах Mac, Linux и Windows.

Недостатки: бесплатная версия среды разработки программного обеспечения не включает в себя все функции. В то же время премиум версия явно стоит своих денег.

9. RubyMine.

Еще одна премиум IDE, разработанная компанией Jet Brains. Предлагается 30-дневная бесплатная ознакомительная версия, полная версия стоит $210 — $687 в год. Удобная навигация, логичная организация рабочего процесса и совместимость с большинством платформ делают RubyMine одним из популярных инструментов для разработчиков.

Поддерживаемые языки: CoffeeScript, CSS, HAML, HTML, JavaScript, LESS, Ruby и Rails, Ruby и SASS.

Особенности:

– сниппеты кода, автоматическое заполнение и автоматический рефакторинг;
– дерево проектов, которое позволяет быстро анализировать код;
– схема модели Rails;
– просмотр проекта Rails;
– RubyMotion поддерживает разработку под iOS;
– поддержка стека включает в себя Bundler, pik, rbenv, RVM и другие;
– отладчики JavaScript, CoffeeScript и Ruby;
– интеграция с CVS, Git, Mercurial, Perforce и Subversion.

Недостатки среды разработки: чтобы RubyMine работала бесперебойно, компьютеру требуется не менее 4 ГБ оперативной памяти. Некоторые пользователи также жалуются на отсутствие опций настройки GUI.

10. Xcode.

Набор инструментов для создания приложений под iPad, iPhone и Mac. Интеграция с Cocoa Touch делает работу в среде Apple простой, вы можете включать такие сервисы, как Game Center или Passbook, одним кликом мыши. Встроенная интеграция с сайтом разработчика помогает создавать полнофункциональные приложения «на лету».

Поддерживаемые языки: AppleScript, C, C++, Java, Objective-C.

Особенности:

– элементы пользовательского интерфейса можно легко связать с кодом реализации;

– компилятор Apple LLVM сканирует код и предоставляет рекомендации по решению проблем производительности;

– панель навигации обеспечивает быстрое перемещение между разделами;

– Interface Builder позволяет создавать прототипы без написания кода;

– пользовательский интерфейс и исходный код можно подключить к сложным прототипам интерфейсов всего за несколько минут;

– редактор версий включает в себя файлы журнала и хронологии;

– распределение и объединение процессов удобно при командной работе;

– Test Navigator позволяет быстро тестировать приложения в любой момент разработки;

– автоматически создает, анализирует, тестирует и архивирует проекты благодаря интеграции с сервером OX X;

– рабочий процесс настраивается с помощью вкладок, поведения и фрагментов;

– библиотека инструментов и каталог ресурсов.

Недостатки инструментальной среды разработки: для запуска Xcode нужен компьютер от компании Apple. А для загрузки создаваемых приложений в Apple Store – лицензия разработчика.

4. Java IDE для быстрой веб-разработки

Для создания лучшего Java-приложение, которое будет быстрым, стабильным и надежным потребуется не только опытный разработчик с отличными навыками программирования, но и правильная среда разработки Java.

Java-IDE действительно необходима, а особенно такая, которая поддерживает сразу несколько языков программирования.

– NetBeans – лучшая Java-IDE, развиваемая Oracle (бесплатное программное обеспечение);

– IntelliJ IDEA – лучшая IDE для Java – разработчиков (премиум Java-IDE);

– Eclipse – популярная IDE для Java-разработчиков (бесплатное программное обеспечение);

– Android Studio – Java-IDE, предназначенная для Android-разработчиков;

– JDeveloper – бесплатная IDE для упрощения разработки Java-приложений;

– DrJava – компактная среда разработки Java-программ;

– JEdit – продвинутый редактор кода для Java-разработчиков;

– MyEclipse – Java-IDE полного стека для веб-разработки;

– JCreator – простая IDE для Java-разработки;

– GreenFoot – онлайн Java-инструктор и IDE.

1. NetBeans – лучшая Java-IDE, развиваемая Oracle (бесплатное программное обеспечение)

На сегодняшний день лучшая среда разработки Java, созданная разработчиками для разработчиков с целью сделать рабочий процесс максимально простым и эффективным.

С помощью NetBeans можно создавать не только корпоративные веб-приложения, но и мобильное программное обеспечение, приложения для ПК. Кроме этого, NetBeans позволяет повторно использовать рабочее пространство проекта на нескольких платформах, таких как Windows, Linux, Mac OS X и Solaris от Oracle.

NetBeans решает большинство проблем, с которыми разработчик может столкнуться в процессе работы. Эта IDE позволяет управлять всеми задачами: анализом, проектированием, отладкой, модульным тестированием, управлением исходным кодом и развертыванием.

В разрабатываемом коде могут быть не выявленные ошибки, которые трудно найти. Но встроенные инструменты NetBeans, такие как анализ статического кода, интеграция плагина FindBug помогут выявить и устранить сложные проблемы в коде. Кроме этого, отладчик NetBeans предоставляет возможность быстро перемещаться по коду, устанавливать контрольные точки, добавлять заметки, делать снэпшоты и отслеживать исполнение кода.

Также IDE поставляется с мощным профайлером кода, который помогает оптимизировать производительность приложения (*скорость работы и потребление памяти*). Кроме этого, она включает в себя визуальный отладчик, позволяющий корректировать создаваемые пользовательские интерфейсы, без необходимости правки кода.

Стоит уделить внимание инструменту Maven, поддерживаемому NetBeans. С его помощью добавление зависимостей производится всего в несколько кликов. Это позволяет избежать лишней загрузки / перестроения индексов.

Давайте посмотрим, за что еще NetBeans считается одной из лучших Java IDE.

Она поддерживает HTML5, CSS3, JavaScript (ES 5/6) и Angular JS. NetBeans также поддерживает автоматической завершение для этих языков, а также директивы Angular.

NetBeans позволяет разрабатывать мобильные приложения с помощью таких платформ, как Cordova и PhoneGap. А также с помощью GlassFish и WebLogic развертывать любые веб-приложения и тестировать конечный функционал. Они добавляют к основным возможностям ядра поддержку Java 8.

NetBeans поддерживает все популярные веб-приложения (Spring / Struts / Wicket) и библиотеки, такие как PrimeFaces, RichFaces, ICEfaces.

2. IntelliJ IDEA — лучшая IDE для разработчиков Java (премиум Java-IDE).

Быстрый и надежный инструмент для веб-разработки на Java. Эту среду разработки для Java под Windows развивает компания JetBrains. Она постоянно обновляет и дополняет ее, чтобы она соответствовал потребностям рынка.

IntelliJ IDEA поставляется в двух редакциях. Бесплатная версия предназначена для студентов, преподавателей и разработчиков плагинов. Платная — для организаций, разрабатывающих коммерческие приложения.

IntelliJ Community Edition.

Бесплатная версия поддерживает такие платформы, как Android, Swing и JavaFX, Java, Groovy, Kotlin, Scala, Go, Dart, Erlang и Python.

Кроме этого, данная IDE является довольно компактной и содержит несколько других функций, таких как выполнение тестов JUnit / TestNG, отладка, автоматическое завершение кода, проверка кода, рефакторинг кода, Ant и визуальный графический редактор GUI.

Также можно воспользоваться четырьмя типами систем контроля версий: Git / GitHub, SVN, Mercurial и CVS. Доступен плагин Docker, обеспечивающий поддержку развертывания веб-приложений. Бесплатная версия также включает в себя Decompiler и ByteCode для выполнения расширенного анализа.

IntelliJ Ultimate Edition.

Для начинающих программистов бесплатная IDE предоставляет широкие возможностями, которые помогут им сделать первые шаги в веб-разработке на Java. Более продвинутые пользователи, скорее всего, предпочтут платную версию, которая дает возможность использовать продвинутые функции.

Это превосходная среда разработки Java, поддерживающая многие языки программирования, например JavaScript, TypeScript, SQL, CSS, LESS, Sass, Stylus, CoffeeScript, ActionScript, Ruby и PHP.

Для профессиональной веб-разработки вам понадобится версия Ultimate. Она объединяет в себе такие фреймворки, как Spring MVC, J2EE (JSF / JAX-RS / CDI / JPA), Grails, Griffin, React, Angular JS, Node.js, Django, Flask, CMS (Drupal / WordPress / Laravel).

Платная версия имеет дополнительную поддержку контроля версий, например, TFS, Perforce, ClearCase и Visual SourceSafe. В ней доступно множество вариантов настройки стратегии развертывания с использованием Tomcat, TomEE, GAE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Jetty и Virgo.

В этой версии реализована поддержка NPM, WebPack, Gulp и Grunt. Она включает в себя такие инструменты, как диаграммы (UML / Dependencies), матрица зависимостей.

3. Eclipse — популярная IDE для Java-разработчиков (бесплатное программное обеспечение).

Это современная, кроссплатформенная и свободно распространяемая IDE с открытым исходным кодом, доступная для корпоративной веб-разработки. В 1998 году IBM Software Group решила создать IDE, которая смогла бы закрепиться на рынке и стать одним из его лидеров. Это привело к созданию Eclipse.

Интересно, что Eclipse сама по себе является примером лучших Java-приложений, поскольку написана на Java. Она всегда была стабильной полнофункциональной платформой высочайшего качества для разработки современных веб-приложений.

Eclipse обладает всеми необходимыми функциями, которые должна иметь идеальная Java IDE:

– Поддержка Java 8.0 и 9.0;

– рефакторинг кода, редактирование кода с проверкой запроса, инкрементная компиляция, перекрестные ссылки, автоматическое предложение вариантов кода;

– интегрированный статический анализ кода;

– интеллектуальное завершение кода и быстрое исправление;

– удобство и производительность;

– поддержка Windows / Linux / Mac OS X.

Актуальная на данный момент версия среды разработки для языка Java — Neon. В ней было реализовано много новых функций:

– редактор Java показывает шаблон по умолчанию для размещения +ve / -ve проверок;

– теперь IDE помогает классифицировать уровень опасности для конкретных ошибок в коде;

– Code Assist поддерживает шаблоны подстроки;

– появилась возможность настраивать нулевой анализ на основе аннотаций для использования нескольких наборов типов аннотаций;

– теперь отладчик принимает дополнительные аргументы отладки, а просмотр ресурсов операционной системы позволяет использовать в отладке информацию о процессах;

– моделирование позволило внести значительные улучшения в генерирование кода, редакторы моделей, сравнение моделей и пользовательские графические редакторы.

Eclipse содержит много функций, позволяющих экономить время. Вы даже можете создавать собственные шаблоны кода, чтобы повысить скорость разработки.

4. Android Studio — Java IDE, созданная специально для Android-разработчиков.

Современная IDE для разработки приложений Java и Android. Это относительно новый продукт, выпущенный в середине 2013 года. Созданная на базе IntelliJ IDEA, Android Studio обеспечивает оптимальные условия для разработки дизайна и рабочего кода.

Основное предназначение инструмента — ускорение процесса разработки приложений для любого Android-устройства. До его выхода разработчикам приходилось полагаться на Eclipse и плагин ADT. Из-за этого операционная система Android отставала от Apple. После выхода Android Studio популярность Android выросла.

IDE поддерживает целый ряд полезных функций: интеллектуальное редактирование кода, отладка, модульное тестирование и профилирование кода. Рассмотрим каждую из них подробнее.

Функции среды разработки Java для Android:

– мгновенное обновление и запуск, чтобы изменения кода немедленно вносились в запущенное приложение без его перезапуска;

– встроенный эмулятор легко устанавливается и запускает приложения на различных конфигурациях устройств;

– редактор нового поколения позволяет писать качественный код, экономит время и повышает производительность разработки; Можно использовать расширенное автоматическое завершение кода, рефакторинг и проверку кода;

– встроенная система сборки (Gradle) помогает в автоматизации сборки, формировании зависимостей и подготовке пользовательских конфигураций сборки;

– Gradle — это система сборки под различные устройства. Она позволяет создавать приложения, работающие на всех Android -устройствах;

– простая интеграция с системами контроля версий, например, GitHub и SVN;

– возможность подготовки сборки для исполнения на сервере CI, таком как Jenkins и Bamboo;

– шаблоны готового кода помогают упростить и ускорить разработку приложений;

– можно проверить свое приложение с помощью JUnit 4 и регрессировать пользовательский интерфейс с помощью Espresso Test Recorder;

– отслеживание ошибок в коде с помощью встроенного инструмента анализа;

– также доступны другие встроенные инструменты, такие как Layout Editor, Vector Asset Studio, APK-анализатор и редактор переводов.

– Android Studio — это одна из лучших сред разработки Java, которую стоит попробовать для разработки как стационарных, так и мобильных приложений.

5. JDeveloper — бесплатная среда для упрощения разработки Java-приложений.

Мощная Java IDE с открытым исходным кодом от Oracle. Она поддерживает все этапы создания приложения.

JDeveloper включает в себя усовершенствованный редактор кода, который повышает скорость разработки с помощью аудита кода, интегрированного модульного тестирование и профилирования. Также визуальный редактор можно использовать для программирования на SQL, XML, PHP, JavaScript, HTML и CSS.

JDeveloper оптимизирована для управления приложениями J2EE, базами данных, веб-службам REST / SOAP, мобильными приложениями, компонентами и приложениями Oracle Fusion Middleware.

Данная IDE содержит ряд встроенных инструментов для ускорения разработки. Один из них — интегрированный сервер WebLogic, который позволяет запускать, тестировать и отлаживать J2EE-приложения. А также браузер SQL и редактор PL / SQL, которые помогут в построении запросов, просмотре баз данных и создании отчетов; редактор WSDL, который ускоряет разработку SOAP и REST.

JDeveloper также включает в себя два инструмента отслеживания — встроенные генераторы схем XSD / XML и инструменты для запуска тестов.

6. DrJava — простая среда для разработки Java-программ.

Простая и адаптивная среда разработки Java-программ. Первоначально ее задача заключалась в том, чтобы познакомить новичков с Java через интерактивное обучение. Позже были добавлены дополнительные функции для продвинутых пользователей.

DrJava объединяет такие функции, как интеллектуальный редактор кода, интерактивный терминал для оценки выходных данных программы, отладчик исходного кода и инструмент запуска модульных тестов.

IDE поддерживает Java 8 и более новые версии. И еще одно полезное дополнение — интеграция с инструментом Jacoco для охвата кода. Когда вы запускаете модульные тесты, он генерирует отчеты, содержащие в себе ссылки, указывающие на охват кода.

По нашему мнению, DrJava лучше всего подходит для пользователей, которые недавно начали изучать Java. Они смогут развивать свои навыки Java - программирования, а затем перейти на интегрированные среды разработки Java NetBeans, Eclipse или IntelliJ.

7. JEdit — продвинутый редактор для Java-разработчиков.

Интуитивный редактор кода, используемый Java-программистами на протяжении многих лет. Он поддерживает Mac OS X, OS / 2, Unix, VMS и Windows.

Как и в случае с Eclipse, для разработки JEdit использован язык программирования Java. И это также одно из лучших Java-приложений, доступных для свободного использования на основе лицензии GPL 2.0.

UDE предлагает ряд современных функций:

– JEdit включает в себя встроенный язык создания макросов для автоматизации любой повторяющейся задачи. Также можно использовать доступные макросы;

– JEdit-пакеты с интерактивным менеджером плагинов для поиска и загрузки необходимых плагинов.

8. MyEclipse — полноценная Java IDE для веб-разработки

MyEclipse является одним из инструментов, призванных сделать процесс разработки более эффективным. MyEclipse развивает компания Genuitec, которая создала его на базе Eclipse.

Данная среда разработки для языка Java поддерживает корпоративную разработку, веб-разработку и веб-разработку полного цикла.

Рассмотрим функции, которые она предоставляет:

– Расширенная поддержка J2EE для всех известных фреймворков, таких как Spring, JPA, JSF, Jquery и Cordova;

– Интеллектуальное редактирование кода с подсказками, мгновенная проверка, подсветка синтаксиса, удобная справка (панель инструментов с хлебными крошками и мини карта);

– Полное управление циклом развития проекта;

– Встроенная поддержка MySQL, SQL Server и Sybase;

– Встроенный WebSphere для запуска, тестирования и отладки приложения «на лету»;

– Другие серверы приложений, такие как WebLogic, Apache Tomcat, GlassFish и Derby, поддерживаются по умолчанию из коробки;

– Создание динамических веб-приложений с использованием TypeScript и Angular 2 (ES6);

– Тонкая поддержка для разработки и тестирования RESTful веб-сервисов;

– Возможность предварительного просмотра результата исполнения кода с помощью встроенного инструмента CodeLive. Также можно перейти к источнику любого элемента на веб-странице;

– Кроссбраузерное тестирование и поддержка мобильных эмуляторов;

– Обсуждение обновлений кода через интеграцию Slack.

MyEclipse содержит функции, которые могут повысить производительность любого Java-разработчика.

9. JCreator — простая среда для Java- разработки

Является одной из лучших сред разработки Java, которая предоставляет разработчикам интерактивный опыт разработки. В этом плане она предоставляет больше возможностей, чем любой другой редактор кода.

Развитием JCreator занимается IT-компания Xinox Software. IDE доступна в двух вариантах: Lite версия с ограниченными возможностями и Pro edition с полным набором функций.

JCreator включает в себя множество полезных функций:

– Интерфейс Easy Project Management похож на Visual Studio;

– пользовательская цветовая схема для организации кода;

– возможность настроить и использовать в проектах нескольких профилей JDK;

– выбор шаблона проекта для быстрого запуска;

– поддержка средств сборки и управления версиями, таких как Ant и CVS;

– возможность запускать приложение как апплета с помощью JUnit или в терминале;

– встроенные инструменты для вызова внешних функций и утилит. Форматирование Java-кода, компилятор RMI.

При создании этой IDE должное внимание было уделено простоте использования, скорости, производительности и отличному пользовательскому интерфейсу. Во многом авторы ориентировались на Microsoft Visual Studio.

10. GreenFoot — онлайн-инструктор Java и IDE.

Специализированный редактор кода, который предназначен для изучения Java новичками в интерактивном режиме. Он формирует среду для генерации симуляции графических программ и игр, написанных на Java.

IDE подробно документирует каждую функцию, предоставляя документацию, как новичкам, так и опытным пользователям. Это помогает быстро развивать свои навыки.

Эта среда разработки Java имеет понятный пользовательский интерфейс и включает в себя одно окно для моделирования всех объектов и классов приложения.

У GreenFoot есть своя целевая аудитория, к которой относятся студенты, преподаватели и тренеры. При этом в ней реализованы все функции, необходимые для Java- разработчика:

– Редактор GUI позволяет добавлять классы одним кликом мыши. Но можно добавлять и другие элементы, в зависимости от конкретного варианта использования;

– создаваемые классы легко расширяются или наследуются. Это же можно делать и с помощью визуального редактора;

– GreenFoot поддерживает большое количество библиотек изображений, сгруппированных по таким категориям, как животные, объекты, здания, люди и символы;

– в данной IDE реализованы функции управления проектами, поддержки кода, автоматического завершения, подсветки синтаксиса и другие инструменты.

Такие интерактивные функции принесли GreenFoot большое количество поклонников. Она выступает в качестве отличной платформы для обучения Java, с помощью которой начинающие программисты могут создавать стабильные, надежные и масштабируемые Java-приложения.

11. Visual Studio Code.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для

кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code — редактор кода, с поддержкой более 30 языков программирования и форматов файлов, а также обладающий рядом дополнительных, полезных возможностей.

Основные возможности и преимущества программы.

1. Visual Studio Code поддерживает работу с TypeScript, JavaScript, Node.js и Mono.
2. Имеются встроенные отладчик и командная строка.
3. Поддержка практически всех языков программирования.
4. Наличие встроенной библиотеки элементов кода.
5. Автозавершение при вводе кода.
6. Добавление в библиотеку собственных сниппетов.
7. Подсветка синтаксиса.
8. Одновременная работы с несколькими проектами.
9. Поддержка многооконного и двухпанельного режимов.
10. Расширение функционала с помощью плагинов.
11. Интеграция с Visual Studio Team Services, GitHub и GIT.
12. Наличие встроенных средств для тестирования, сборки, упаковки и развертывания приложений.
13. Публикация созданных программных продуктов в Microsoft Azure (через посредство Visual Studio Team Services).
14. Интегрированная система подсказок.
15. Командная работа над проектами.
16. Широкий набор настроек и кроссплатформенность.

Visual Studio Code — это редактор исходного кода. Он поддерживает ряд языков программирования, подсветку синтаксиса, IntelliSense, рефакторинг, отладку, навигацию по коду [17], поддержку Git и другие возможности. Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON файлы (например, пользовательские настройки). Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

Visual Studio также позволяет заменять кодовую страницу при сохранении документа, символы перевода строки и язык программирования текущего документа.

С 2018 года появилось расширение Python для Visual Studio Code с открытым исходным кодом. Оно предоставляет разработчикам широкие возможности для редактирования, отладки и тестирования кода.

На март 2019 года посредством встроенного в продукт пользовательского интерфейса можно загрузить и установить несколько тысяч расширений только в категории «programming languages» (языки программирования).

Visual Studio Code имеет поддержку плагинов, доступных через Visual Studio Marketplace. Они могут включать в себя дополнения к редактору, поддержку дополнительных языков программирования, статические анализаторы кода.

С мая 2019 года доступен закрытый тест редактора Visual Studio Online на основе VS Code. Он поддерживает все расширения и IntelliCode. Не нужно путать это с репозиторием для DevOps, который также не так давно запустила Microsoft

12. Sublime Text.

Sublime Text — проприетарный текстовый редактор. Поддерживает плагины на языке программирования Python.

Разработчик позволяет бесплатно и без ограничений ознакомиться с продуктом, однако программа уведомляет о необходимости приобретения лицензии.

Некоторые возможности:

- быстрая навигация (Goto Anything);
- командная палитра (Command Palette);
- API плагинов на Python;
- одновременное редактирование (Split Editing);
- высокая степень настраиваемости (Customize Anything).

Поддержка языков. Sublime Text поддерживает большое количество языков программирования и имеет возможность подсветки синтаксиса для C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL и XML.

В дополнение к тем языкам программирования, которые включены по умолчанию, пользователи имеют возможность загружать плагины для поддержки других языков.

Менеджер пакетов. Sublime Text может быть оснащён менеджером пакетов, который позволяет пользователю находить, устанавливать, обновлять и удалять пакеты без перезагрузки программы. Менеджер поддерживает установленные пакеты в актуальном состоянии, загружая новые версии из репозиториев. Кроме того, он предоставляет команды для активации и деактивации установленных пакетов.

Интерфейс. Редактор содержит различные визуальные темы, с возможностью загрузки дополнительных.

Пользователи видят весь свой код в правой части экрана в виде миникарты, при клике на которую можно осуществлять навигацию.

Есть несколько режимов экрана. Один из них включает от 1 до 4 панелей, с помощью которых можно показывать до четырёх файлов одновременно. Полноценный (free modes) режим показывает только один файл без каких-либо дополнительных меню вокруг него.

Выделение столбцов и множественная правка. Выделение столбцов целиком или расстановка нескольких указателей по тексту, что делает возможным мгновенную правку. Указатели ведут себя, будто каждый из них — один в тексте. Команды типа перемещение на знак, перемещение на строку, выборка текста, перемещение на слово или его части (CamelCase, разделённый дефисом или подчёркиванием), перемещение в начало/конец строки и т.д., влияют на все указатели независимо и сразу, позволяя править сложно-структурированный текст быстро, без использования макрокоманд или регулярных выражений.

Автодополнение. Когда пользователь набирает код, Sublime Text, в зависимости от используемого языка, будет предлагать различные варианты для завершения записи. Редактор также автоматически завершает созданные пользователем переменные.

Подсветка синтаксиса и высокая контрастность. Тёмный фон Sublime Text предназначен для увеличения контрастности текста. Основные элементы синтаксиса выделены разными цветами, которые лучше сочетаются с тёмным фоном, нежели со светлым.

Поддержка систем сборки. Sublime Text позволяет пользователю собирать программы и запускать их без необходимости переключаться на командную строку. Пользователь также может настроить свою систему сборки и включить автоматическую сборку программы каждый раз при сохранении кода.

Заготовки (сниппеты). Сохранение фрагментов часто используемого кода, ключевые слова для их запуска.

Переход по файлам. Навигационный инструмент, который позволяет пользователям перемещаться между файлами, а также внутри них, с помощью нечёткого поиска.

Другие особенности

– дополнительно реализована функция авто сохранения, помогающая пользователям не потерять проделанную работу;

– настраиваемые комбинации клавиш и инструмент навигации позволяют назначать свои комбинации клавиш для меню и панелей инструментов (только для первой версии, во второй и третьей — Command Palette);

– возможность поиска по мере набора используется для поиска в документе;

– функция проверки синтаксиса работает подобным же образом, проверяя корректность прямо во время ввода;

– есть возможность автоматизации с помощью макросов и повтора последних действий;

– команды редактирования, включая редактирование отступов, переформатирование параграфов и объединение строк.

Использование системы управления версиями GIT

Вопросы:

1. Основные понятия.
2. Отличия распределённых систем управления версиями.
3. Основные возможности и особенности Git.
4. Слияние и разделение ветвей.

1. Основные понятия

Git – это распределённая система управления версиями файлов. Код программы написан в основном на языке C. Проект был создан Линусом Торвальдсом в 2005 году для управления разработкой ядра Linux и, как и GNU/Linux, является свободным программным обеспечением (ПО), при этом стороннее использование подчиняется лицензии GNU GPL версии 2. Вкратце данное соглашение можно охарактеризовать как ПО со свободным кодом, которое должно развиваться открыто, т.е. любой программист вправе продолжить совершенствование проекта на любом его этапе. За свое недолгое время существования данная система была введена многими ведущими разработчиками. Git используется в таких известных Linux-сообществу проектах, как Gnome, GNU Core Utilities, VLC, Cairo, Perl, Chromium, Wine.

Системы управления версиями (Version Control Systems) – это программное обеспечение, призванное автоматизировать работу с историей файла (или группы файлов), обеспечить мониторинг изменений, синхронизацию данных и организовать защищенное хранилище проекта. Короче говоря, основная задача систем управления версиями – упростить работу с изменяющейся информацией. Разберем общий вид разработки на примере.

Предположим, есть некий проект, который вы разрабатываете, несколько отделов программистов и вы – координатор (или руководитель). По отношению к системе контроля, будь то сервер (если речь идет о централизованной системе) или локальная машина, любой разработчик проекта ограничен только правами доступа на изменение и/или чтение версий файлов данного хранилища. В любой момент вы можете сделать откат данных до необходимой вам версии. Вы, как координатор, можете ограничить доступ определенным пользователям на обновление версии файла. Также СУВ предоставляет интерфейс наблюдения и поиска версий файлов. Например, можно создать запрос: “Где и когда менялся данный кусок кода?”.

Система предполагает защищенное хранение данных, т.е. любой хранимый в ней блок имеет множество клонов. Так, например, при повреждении какого-либо файла вы своевременно можете заменить его копией. Для уменьшения объема данных проекта часто используется дельта-компрессия – такой вид хранения, при котором хранятся не сами версии файла, а только изменения между последовательными ревизиями.

2. Отличия распределённых систем управления версиями

Распределённые системы управления версиями – это СУВ, главной парадигмой которых является локализация данных каждого разработчика проекта. Иными словами, если в централизованных СУВ все действия, так или иначе, зависят от центрального объекта (сервер), то в распределенных СУВ каждый разработчик хранит собственную ветвь версий всего проекта. Удобство такой системы в том, что каждый разработчик имеет возможность вести работу независимо, время от времени обмениваясь промежуточными вариантами файлов с другими участниками проекта. Рассмотрим эту особенность, продолжая предыдущий пример.

У каждого разработчика на машине есть свой локальный репозиторий – место хранения версий файлов. Работа с данными проекта реализуется над вашим локальным репозиторием, и для этого необязательно поддерживать связь с остальными (пусть даже и главными) ветвями разработки. Связь с другими репозиториями понадобится лишь при изменении/чтении версий файлов других ветвей. При этом каждый участник проекта задает права собственного хранилища на чтение и запись. Таким образом, все ветви в распределенных СУВ равны между собой, и главную из них выделяет координатор. Отличие главной ветви лишь в том, что на неё мысленно будут равняться разработчики.

3. Основные возможности и особенности Git

Стоит сказать, что система если и не произвела фурор, то немного всколыхнула сообщество в области СУВ своей новизной и предложила новый путь развития. Git предоставляет гибкие и простые в использовании инструменты для ведения истории проекта.

Особенностью Git является то, что работа над версиями проекта может происходить не в хронологическом порядке. Разработка может вестись в нескольких параллельных ветвях, которые могут сливаться и разделяться в любой момент проектирования.

Git – довольно гибкая система, и область её применения ограничивается не только сферой разработки. Например, журналисты, авторы технической литературы, администраторы, преподаватели вузов вполне могут использовать её в своем роде деятельности. К таковым задачам можно отнести контроль версий какой-либо документации, доклада, домашних заданий.

Выделим основные отличия Git от других распределенных и централизованных СУВ.

Архитектура Git. SHA1 (Secure Hash Algorithm 1) – это алгоритм криптографического хеширования. Каждый файл вашего проекта в Git состоит из имени и содержания. Имя – это первые 20 байтов данных, оно наглядно записывается сорока символами в шестнадцатеричной системе счисления. Данный ключ получается хешированием содержимого файла. Так, например, сравнив два имени, мы можем почти со стопроцентной вероятностью сказать, что они имеют одинаковое содержание. Также, имена идентичных объектов в разных ветвях (репозиториях) – одинаковы, что позволяет напрямую оперировать данными. Хорошим дополнением сказанному выше служит ещё то, что хеш позво-

ляет точно определить поврежденность файлов. Например, сравнив хеш содержимого с именем, мы можем вполне точно сказать, повреждены данные или нет. Далее под именем мы будем понимать имя файла, а строку символов будем называть SHA1-хешем.

Стоит упомянуть о так называемых коллизиях. “Вполне точно определить поврежденность” означает, что существуют такие файлы, различные по содержанию, SHA1-хеш которых совпадает. Вероятность таких коллизий очень мала, и, по предварительной оценке, равна 2 в -80-й степени (~ 10 в -25-й степени). Точной оценки нет, так как на данный момент мировому сообществу не удалось эффективно расшифровать данную криптографическую схему.

Объекты Git. Работу с версиями файлов в Git можно сравнить с обычными операциями над файловой системой. Структура состоит из четырех типов объектов: Blob, Tree, Commit и References; некоторые из них, в свою очередь, делятся на подобъекты.

Blob (Binary Large Object) – тип данных, который вмещает лишь содержимое файла и собственный SHA1-хеш. Blob является основным и единственным носителем данных в структуре Git. Можно провести параллель между данным объектом и инодами (inodes) в файловых системах, поскольку их структура и цели во многом схожи.

Дерево (Tree) – тип данных, который содержит:

- собственный SHA1-хеш;
- SHA1-хеш blob’ов и/или деревьев;
- права доступа Unix-систем;
- символьное имя объекта (название для внутреннего использования в системе).

По своей сути объект является аналогом директории. Он задает иерархию файлов проекта.

Commit – тип данных, который содержит:

- собственный SHA1-хеш;
- ссылку ровно на одно дерево;
- ссылку на предыдущий commit (их может быть и несколько);
- имя автора и время создания commit’a;
- имя коммитера (commiter – человек, применивший commit к репозиторию, он может отличаться от автора) и время применения commit’a;
- произвольный кусок данных (блок можно использовать для электронной подписи или, например, для пояснения изменений commit’a).

Данный объект призван хранить снимок (версию) группы файлов в определенный момент времени, можно сравнить его с контрольной точкой. Commit’ы можно объединять (merge), разветвлять (branch) или, например, установить линейную структуру, тем самым отражая иерархию версий проекта.

Reference – тип данных, содержащий ссылку на любой из четырех объектов (Blob, Tree, Commit и References). Основная цель его – прямо или косвенно указывать на объект и являться синонимом файла, на который он ссылается. Тем самым повышается понимание структуры проекта. Очень неудобно опери-

ровать бессмысленным набором символов в названии, ссылку же, в отличие от SHA1-хеша, можно именовать так, как удобнее разработчику.

Из ссылок, в свою очередь, можно выделить ряд подобъектов, имеющих некоторые различия: Ветвь, Тег. Рассмотрим их.

Ветвь (Head, Branch) – символьная ссылка (Symbolic link), которая указывает на последний в хронологии commit определенной ветви и хранит SHA1-хеш объекта. Является типом данных журналируемых файловых систем. Данный вид объекта определяется не в самом Git, а наследуется от операционной и файловой систем. Ветвь используется как синоним файла, на который она ссылается, т.е. Git позволяет оперировать ею напрямую. Можно позволить себе не задумываться о том, работаете ли вы с последней версией или нет.

Тег (tag) – тип данных, который в отличие от ветвей неизменно ссылается на один и тот же объект типа blob, tree, commit или tag. Его, в свою очередь, можно разделить на легковесный (light tag) и тяжеловесный или аннотированный (annotated tag). Легкий тег, кроме неизменности ссылки, ничем не отличается от обычных ветвей, т.е. содержит лишь SHA1-хеш объекта, на который ссылается, внутри себя. Аннотированный тег состоит из двух частей:

- первая часть содержит собственный SHA1-хеш;
- вторая часть состоит из:
- SHA1 объекта, на который указывает аннотированный тег;
- тип указываемого объекта (blob, tree, commit или tag);
- символьное имя тега;
- дата и время создания тега;
- имя и e-mail создателя тега;
- произвольный кусок данных (данный блок можно использовать для электронной подписи или для пояснения тега).

Иными словами, проект в Git представляет собой набор blob’ов, которые связаны сетью деревьев. Полученная иерархическая структура может, в зависимости от времени, быть отражена в виде commit’ов – версий, а для понимания их структуры в Git присутствуют такие объекты, как ссылки. Исключая действия со ссылками, почти вся работа с объектами системы максимально автоматизирована изнутри. Отталкиваясь от механизма ссылок, мы приходим к следующей идее – работать именно над группами файлов. По мнению автора, мысль является ключевой в философии Git. Задав, например, операцию для данного commit’а, она рекурсивно отработает свою часть по дереву, на которое ссылается. Являясь расширением общепринятого взгляда “действие над каждым файлом”, нововведение упрощает реализацию и подход со стороны программиста над повседневными задачами СУВ, такими как слияние/разделение ветвей, опять же рекурсивно автоматизируя процесс. Данный подход прост для понимания, быстро работает и гибок в реализации своих целей. Многие из этих черт достигаются благодаря Unix-ориентированности системы, т.е. оперируя стандартными устройствами, Git опирается на уже имеющиеся в операционной системе решения.

Проясним момент хранения данных. Содержание файлов разных версий в хронологии занимает довольно много памяти. Так, например, в проекте из двадцати файлов двадцати версий архив будет весить в 20 раз больше (возможно, порядка сотни мегабайтов), а что будет, если количество и тех, и других в 10 раз больше (вроде бы ненамного)? Размер занятого пространства возрастет в 100 раз (т.е. примерно 1 ГБ). В реальных задачах скорость роста занимаемой памяти далеко не линейно зависит от времени. Для решения данной проблемы существует несколько оптимизаций:

– каждый объект Git хранится в виде обыкновенного архива (tar.gz);

– для всей иерархии файлов применяется последовательная дельта-компрессия.

Разберем на примере.

У вас есть трехлетняя история вашего проекта, в ней порядка тысячи файлов и ста версий. Если в определенный момент нужно будет обратиться к самой ранней версии, Git придется разархивировать дельта-компрессию всей истории файла. Неутешительно, но на данный процесс может уйти до полудня. Git предлагает делать так называемые контрольные точки, т.е. хранить недельта-архивированный файл через некоторое количество версий, которое назовем глубиной компрессии. Тогда в нашем примере вся история сужается до некоторого наперед заданного количества дельта-компрессий, разархивировав которые, можно взглянуть на любую версию в хронологии. Заметим, что дельта-компрессию наиболее целесообразно использовать над одними видами ближайших в иерархии объектов, для этого репозиторий необходимо отсортировать соответственно по типу и размеру. Данный ряд операций, описанных в этом пункте, выполняет функция git-repack (и git-gc, которая её содержит).

4. Слияние и разделение ветвей

Данный вопрос очень трудоемок и насыщен, в связи с чем введем понятия слияния и разделения только в общих чертах. Снова обратимся к примеру.

Представим себе момент разработки проекта, когда главной поставленной целью является скорость работы программы. Один из возможных тактических вариантов решения – разбить разработчиков на две группы, каждая из которых будет решать одну и ту же задачу. При этом ветвь истории проекта должна раздвоиться. Данная процедура называется ветвление (branch). Действие разветвления ветви – это простое создание её копии, которая впоследствии будет иметь свою историю.

Пусть мы получили два уже законченных результата одной и той же задачи, над которой работали две группы программистов. Как нам быть? Посмотреть, чей код быстрее и надежнее? Это слишком просто, но не всегда лучший выход. Хорошее решение – это, немного разобравшись в коде и файлах, разбить их на подзадачи или блоки кода. И только тогда уже выявлять сильные и слабые стороны данных кусочков. Конечно, этот вариант подходит только в том случае, когда вы заранее предусмотрели, что впоследствии сможете собрать все эти частицы воедино. Случай, когда вы сами разрабатываете код, улучшая и исправляя некоторые ошибки, равнозначен приведенному примеру. Данный

процесс объединения двух целых в одно называется слияние (merge). Процесс объединения двух версий и есть ключевой момент ведения проекта. Как бы то ни было, стоит избегать автоматизированного исполнения данной операции. Отличительная черта Git – это максимально достоверный и довольно быстрый способ решения задачи ветвления.

К достоинствам системы можно отнести:

1. Unix-ориентированность.
2. Идеологическая выдержанность (следуя правилам использования системы, очень сложно попасть в безвыходную ситуацию или получить то, чего вы не ожидали).
3. Высокая производительность (это одно из самых явных достоинств системы, плата за которое есть «Идеологическая выдержанность» и «Unix-ориентированность»).
4. Интеграция Git со сторонними СУВ, такими как Subversion, Mercurial, …
5. Управление группой файлов (системе нет необходимости рассматривать изменения в каждом файле по отдельности, она запоминает любые изменения всего проекта, и, если вдруг вам понадобится проследить единичные изменения, она выдаст ровно ту часть, которая связана с данным файлом).
6. Операция слияния (максимально автоматизированная реализация сложной задачи).
7. К недостаткам отнесем:
8. Unix-ориентированность (стоит отметить отсутствие зрелой реализации Git на не Unix-системах).
9. Необходимость периодического выполнения команды git-gc (пакует группы файлов и удаляет те, которые не связанны ссылками).
10. Коллизии хеширования (совпадение SHA1 хеша различных по содержанию файлов).

Фреймворки для быстрой разработки интернет приложений

Вопросы:

1. Основные понятия.
2. Фреймворки для быстрой разработки интернет приложений.

1. Основные понятия

Фре́ймворк (иногда фреймво́рк; англицизм, неологизм от framework — остов, каркас, структура) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Употребляется также слово «каркас», а некоторые авторы используют его в качестве основного, в том числе, не базируясь вообще на англоязычном аналоге. Можно также говорить о каркасном подходе как о подходе к построению программ, где любая конфигурация программы строится из двух частей.

1. Постоянная часть — каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнёзда, в которых размещается вторая, переменная часть.

2. Сменные модули (или точки расширения).

Отличие от библиотеки. «Фреймворк» отличается от понятия библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. В то время как «фреймворк» диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять, согласно указанным требованиям. Пример программного фреймворка — C.M.F. (Content Management Framework), а пример библиотеки — модуль электронной почты.

Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — «фреймворк» может содержать в себе большое число разных по тематике библиотек.

Другим ключевым отличием «фреймворка» от библиотеки может быть инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова. Во «фреймворке» пользовательский код может реализовывать конкретное поведение, встраиваемое в более общий — «абстрактный» код фреймворка. При этом «фреймворк» вызывает функции (классы) пользовательского кода.

Фреймворк программной системы. Это каркас программной системы (или подсистемы). Может включать: вспомогательные программы, библиотеки кода, язык сценариев и другое ПО, облегчающее разработку и объединение разных компонентов большого программного проекта. Обычно объединение происходит за счёт использования единого API.

Примеры: веб-фреймворки, как PHP-фреймворки Zend Framework и Symfony, или Django, написанный на Python.

Фреймворк приложения. Одно из главных преимуществ при использовании «каркасных» приложений — «стандартность» структуры приложения. «Каркасы» стали популярны с появлением графических интерфейсов пользователя, которые имели тенденцию к реализации стандартной структуры для приложений. С их использованием стало гораздо проще создавать средства для автоматического создания графических интерфейсов, так как структура внутренней реализации кода приложения стала известна заранее. Для обеспечения каркаса, обычно используются техники объектно-ориентированного программирования (например, части приложения могут наследоваться от базовых классов фреймворка).

Одним из первых коммерческих фреймворков приложения был MacApp[en], написанный Apple для «Macintosh». Первоначально созданный с помощью расширенной (объектно-ориентированной) версии языка «Object Pascal», впоследствии он был переписан на «C++». Другие популярные каркасы для «Macintosh» включали:

Metrowerks PowerPlant[en] и MacZoop[en] (все основаны на Carbon);

WebObjects[en] от NeXT.

В различной степени фреймворки приложения представляют собой «Cocoa» для Mac OS X, а также свободные фреймворки, существующие как часть проектов Mozilla, OpenOffice.org, GNOME и KDE.

Microsoft создала похожий продукт для «Windows», который называется Microsoft Foundation Classes (MFC). На данный момент основным продуктом Microsoft для разработки ПО предлагается «.NET Framework».

Кроссплатформенными каркасами приложений (для операционных систем «Linux», «Macintosh» и «Windows») являются, например, widget toolkit[en], wxWidgets, Qt, MyCoRe[de] или FOX toolkit.

Фреймворк концептуальной модели. Абстрактное понятие структуры, которое используется в исследованиях для определения возможных способов решения проблемы или представления идеи.

Реализация фреймворка. «Фреймворк» определяется как множество конкретных и абстрактных классов, а также определений способов их взаимоотношения. Конкретные классы обычно реализуют взаимные отношения между классами. Абстрактные классы представляют собой точки расширения, в которых каркасы могут быть использованы или адаптированы.

Точка расширения — это та «часть» фреймворка, для которой не приведена реализация. Соответственно, каркас концептуальной модели состоит из концептуальных классов, а каркас программной системы — из классов языка программирования общего назначения.

Процесс создания фреймворка заключается в выборе подмножества задач проблемы и их реализаций. В ходе реализаций общие средства решения задач заключаются в конкретных классах, а изменяемые средства — выносятся в точки расширения.

2. Фреймворки для быстрой разработки интернет приложений

Веб-фреймворки сильно изменили мир программирования и стали неотъемлемой частью процесса разработки. Вы можете попробовать поискать информацию о них на сайтах, в статьях и книгах, но найдёте только общую и неоднозначную информацию — ничего, кроме бесконечных определений и сложных терминов, от которых закипает мозг. Пора наконец разобраться, что из себя представляют веб-фреймворки.

Веб-фреймворк — инструмент, облегчающий процесс написания и запуска веб-приложения. Вам не нужно самостоятельно писать кучу кода и тратить время на поиск потенциальных просчётов и ошибок.

На рассвете эры веб-разработки все приложения писались вручную, и только разработчик приложения мог изменить или развернуть его. Веб-фреймворки позволили выбраться из этой западни. С 1995 года вся морока, связанная с изменением структуры приложения, была приведена в порядок благодаря появлению общего подхода к разработке веб-приложений. В это время появились языки для веба. Сейчас их разнообразие позволяет выбрать подходящий как для статических, так и для динамических страниц. В зависимости от

поставленной задачи, вы можете выбрать один фреймворк, покрывающий все нужды, или совместить несколько.

2.1 Типы веб-фреймворков

У фреймворков есть две основные функции: работа на серверной стороне (бэкенд) и работа на клиентской стороне (фронтенд).

Фронтенд-фреймворки связаны с внешней частью приложения. Простыми словами, они отвечают за внешний вид приложения. Бэкенд отвечает за внутренне устройство приложения. Рассмотрим оба типа поподробнее.

Серверные фреймворки. Правила и архитектура таких фреймворков не даёт возможности создать веб-приложение с богатым интерфейсом. Они ограничены в своей функциональности, однако вы всё равно можете создавать простые страницы и разные формы. Также они могут формировать выходные данные и отвечать за безопасность в случае атак. Всё это определённо может упростить процесс разработки. Серверные фреймворки в основном отвечают за отдельные, но критически важные части приложения, без которых оно не сможет нормально работать. Вот несколько самых популярных фреймворков и языки, с которыми они работают:

- Django — Python;
- Zend — PHP;
- Express.js — JavaScript;
- Ruby on Rails — Ruby.

Клиентские фреймворки. В отличие от серверных, клиентские фреймворки никак не связаны с логикой приложения. Этот тип фреймворков работает в браузере. С их помощью можно улучшить и внедрить новые пользовательские интерфейсы. Фронтенд-фреймворки позволяют создавать разные анимации и одностраничные приложения. Все клиентские фреймворки отличаются по функциональности и использованию. Вот некоторые из них:

- Backbone+Marionette;
- Angular;
- Ember.js;
- Vue.js.

Все эти фреймворки используют JavaScript.

Многофункциональные фреймворки. Meteor известен как фулл-стек веб-фреймворк. Это значит, что он удовлетворяет почти все потребности как со стороны клиента, так и со стороны сервера, что делает Meteor чрезвычайно популярным. Вам не нужно тратить время на то, чтобы наладить взаимодействие между двумя фреймворками через REST API — вы можете просто выбрать Meteor и ускорить процесс разработки. Но это не главная особенность этого фреймворка. Обе стороны — серверная и клиентская — работают на одном языке, поэтому вы можете создавать и использовать для них один и тот же код. Следующая особенность — «режим реального времени» — когда вы что-то меняете в одном интерфейсе, изменения происходят и в остальных. В качестве примера можно взять документ или таблицу с общим доступом. Когда вы до-

бавляете комментарии или как-то изменяете содержимое, другие пользователи тоже это видят.

На этом о разделении на типы можно закончить, однако масштабы тоже важны. Фреймворки также отличаются по размеру. Существуют такие монструозные фреймворки, которые предлагают решения для всех задач.

Более легковесные варианты специализируются на решении конкретных задач — такие фреймворки называются микрофреймворками. Они не предоставляют «из коробки» всё, что нужно, однако иногда лучше разложить функциональность на несколько подоходов (фреймворки, микрофреймворки, библиотеки). Функциональность микрофреймворков можно расширять с помощью сторонних приложений и создавать небольшие проекты на их основе или совместить микрофреймворк с основным «большим» фреймворком.

Например, если ваше приложение основано на Django и вам нужны вебсокеты, то вы можете воспользоваться микрофреймворком aiohttp.

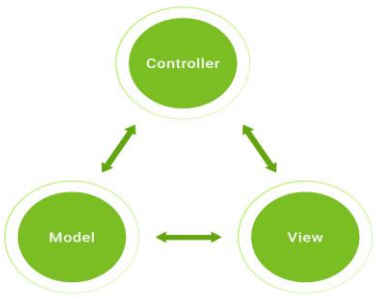
Другой пример: если ваше приложение не очень большое и вам нужна только простая маршрутизация URL и шаблоны с несложным контекстом, вы можете использовать Flask с Jinja2 (или другим шаблонизатором) вместо Django.

2.2 Особенности и архитектура

Несмотря на то, что все фреймворки отличаются друг от друга и выбрать какой-нибудь из них может быть очень сложно, есть несколько вещей, общих для них всех. Речь идёт об архитектуре и особенностях, которые так же важны, как и функции.

Архитектура. Архитектура почти всех фреймворков основана на декомпозиции нескольких отдельных слоёв (приложения, модули и т.д.), что означает, что вы можете расширять функциональность исходя из своих потребностей и использовать изменённую версию вместе с кодом фреймворка или использовать сторонние приложения. Такая гибкость является ещё одним ключевым преимуществом фреймворков. Существует множество open-source сообществ и коммерческих организаций, которые создают приложения или расширения для популярных фреймворков, например, Django REST Framework, ng-bootstrap и т.д.

MVC — Модель, Представление и Контроллер (Model-View-Controller) — три составляющих каждого веб-фреймворка.



Модель содержит все данные и уровни бизнес-логики, её правила и функции.

Представление отвечает за визуальное отображение данных, например, диаграммы, графики и т.д.

Контроллер просто трансформирует данные для команд предыдущих двух составляющих.

Они неотделимы друг от друга, поэтому важно, как следует во всём разобраться, чтобы избежать ошибок во время работы приложения.

Особенности. Теперь давайте посмотрим на некоторые общие особенности, которые делают фреймворки многофункциональными и удобными на практике.

Веб-кэширование. Кэширование просто помогает хранить разные документы и позволяет избежать надоедливой перегрузки сервера. Пользователи могут использовать его в разных системах при определённых условиях. Также оно работает на серверной стороне. Например, вы могли заметить кэшированные веб-страницы на странице результатов поисковой выдачи Google.

Скаффолдинг. Это ещё одна технология, поддерживаемая некоторыми MVC-фреймворками, о которой следует знать. Фреймворк может автоматически сгенерировать типичные части приложения или даже всю структуру проекта (если речь идёт о инициализации). Это позволяет существенно увеличить скорость разработки и стандартизирует кодовую базу.

Система веб-шаблонов. Система веб-шаблонов представляет собой набор разных методологий и программного обеспечения, реализованных для создания и развёртывания веб-страниц. Для обработки веб-шаблонов используются шаблонизаторы. Они являются инструментом фреймворка, отвечающим за веб-публикацию.

Сопоставление URL. Если вы хотите упростить индексацию вашего сайта поисковыми движками, в то же время создавая привлекательное название для сайта, то эта функция фреймворков — то, что вам нужно. Также сопоставление URL может облегчить доступ к страницам вашего сайта.

Приложения. Множество типов веб-приложений поддерживаются веб-фреймворками. В основном они применяются для создания таких приложений, как блоги, форумы, CMS и т.д**.**

Вся эта функциональность свойственна всем фреймворкам. Однако, как ни парадоксально, с таким широким ассортиментом разработчик теряется и не может ничего выбрать. Поэтому вам нужно придумать критерии, согласно которым можно выбрать лучший инструмент, облегчающий разработку. Например, ваш выбор может зависеть от предпочитаемого языка программирования. Как было упомянуто ранее, фреймворки можно найти на всех языках. Кроме того, нужно обратить внимание на круг возможностей набора инструментов фреймворка. Если он покрывает ваши потребности, то вы на верном пути. Говоря о предпочтениях, стоит отметить, что они могут служить как во благо, так и во вред. Конечно, лучше использовать фреймворки, которые проще изучить, однако порой написанные по правилам старой школы и редко используемые, но подходящие фреймворки, могут привести вас к успеху.

2.3 Руководства

Как мы уже убедились, выбор и использование веб-фреймворка может стать тем ещё испытанием. Однако сам процесс не такой уж и сложный, как могло показаться. Есть достаточное количество документов, библиотек и руководств, призванных помочь изучить фреймворк и ответить на все возникающие вопросы. Существуют сайты, которые предоставляют быстрое введение в любой фреймворк.

Например, Tutorialspoint — кладезь разных руководств, покрывающих структуру каждого фрейморка и предоставляющих информацию по разным деталям. Есть руководства по Java-фреймворкам, PHP-фреймворкам и Zend.

Если ваш выбор пал на Ruby on Rails, можете заглянуть в это подробное руководство, которое описывает все «за» и «против» этого фреймворка и учит всему необходимому, начиная с установки.

Безусловно, это далеко не самый полный список. Интернет полон разных источников, из которых можно черпать новые знания. Вы можете посмотреть пошаговые уроки на YouTube и затем выбрать понравившийся фреймворк.

Если у вас появляются какие-то вопросы, то стоит заглянуть на StackOverflow.

Этим сайтом пользуются разработчики по всему миру. Здесь они делятся своим опытом и помогают другим решать их проблемы. Просто задайте вопрос, и вам предложат несколько возможных решений.

Фреймворк BOOTSTRAP

1. Основные понятия.
2. Работа с Bootstrap.

1. Основные понятия

Bootstrap (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков

навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

История.

Эта библиотека начала разрабатываться как внутренняя библиотека компании Twitter под названием Twitter Blueprint. После нескольких месяцев разработки он был открыт под названием Bootstrap 19 августа 2011 года.

Основными нововведениями второй версии, появившейся 31 января 2012 года, стали 12-колоночная сетка и поддержка адаптивности.

Третья версия выпущена 19 августа 2013 года. В ней адаптивность получила дальнейшее развитие, был осуществлён переход к концепции mobile first, оптимизации прежде всего под мобильные устройства. Дизайн по умолчанию стал плоским.

Работа над четвёртой версией начата 29 октября 2014 года. Альфа версия вышла 19 августа 2015 года. Первая бета версия выпущена 10 августа 2017. Вторая бета версия выпущена 19 октября 2017. 18 января 2018 года выпущена первая стабильная версия Bootstrap 4.

Основные инструменты Bootstrap:

Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например, ширина колонки 140 px относится к классу span2 (col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.

Шаблоны — фиксированный или резиновый шаблон документа.

Типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. П.

Медиа — предоставляет некоторое управление изображениями и видео.

Таблицы — средства оформления таблиц, вплоть до добавления функциональности сортировки.

Формы — классы для оформления форм и некоторых событий, происходящих с ними.

Навигация — классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов.

Алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Bootstrap 4.

29 октября 2014 года Марк Отто объявил, что Bootstrap 4 находится в разработке. 6 сентября 2016 года Марк приостановил работу над Bootstrap 3, чтобы высвободить время для работы над Bootstrap 4. На текущий момент было внесено более 4000 изменений к базовому коду Bootstrap 4. Первая стабильная версия вышла 18 января 2018 года.

Bootstrap 4 — это почти полностью переписанный Bootstrap 3.

Перечень самых значительных изменений:

– веб-шрифты по умолчанию (Helvetica Neue, Helvetica, Arial) интегрированы в Bootstrap 4 и заменены набором исходных шрифтов для оптимальной отрисовки текста на любом устройстве под любой ОС;

– переход от использования Less к Sass;

– не поддерживаются IE8, IE9 и iOS 6;

– добавлена поддержка Flexbox, а затем отключена поддержка non flexbox;

– смена основной единицы измерения с px на rem;

– увеличенный глобальный размер шрифта с 14px до 16px;

– новый компонент «карточка», обобщающий панели и другие компоненты;

– удалён шрифт значков Glyphicons;

– удалены компоненты пейджера;

– переписаны почти все компоненты, плагины jQuery и документация.

2. Работа с Bootstrap

Bootstrap является самым популярным фреймворком, у его ближайшего конкурента в 3-5 раз меньше сообщество. Кроме того, это не только css, но и js-фреймворк. То есть в Bootstrap написаны готовые стили и скрипты, для применения которых вам достаточно всего лишь прописать необходимые стилевые классы и атрибуты html-элементам.

Для чего нужен Bootstrap.

Вообще, чтобы лучше понять, для чего вам нужен Bootstrap, можно вернуться немного назад и ответить на вопрос: “А что такое вообще css-фреймворк?”

По сути, если говорить простым языком, это файл или несколько файлов с готовым написанным кодом, которые подключаются к сайту в секции head, после чего становится возможным использование возможностей этого фреймворка.

Фреймворки создают для того, чтобы другим веб-разработчикам было легче верстать сайты. Я уже говорил вначале о том, что сегодня практически любой разработчик после создания с нуля парочки сайтов задумывается, как ему ускорить процесс разработки.

Дело в том, что если мы с вами будем делать разработку сайта с нуля, то придется позаботиться об очень многих вещах. Все css-стили, все веб-сценарии придется писать с нуля, а ведь это могут быть сотни и тысячи строчек кода. Причем вы можете совершить массу ошибок в верстке. Например, попросту ваш шаблон будет по-разному выглядеть в основных браузерах или он будет не адаптивен.

В общем-то, как раз ради адаптивной верстки и стоит использовать Bootstrap, потому что если мы говорим о фиксированных макетах, то их легко сделать даже с нуля. Просто создаем блоки, задаем им фиксированную ширину и работаем по макету.

Но в случае с адаптивной верстkой все в разы сложнее. Вам нужно будет сделать так, чтобы на любых разрешениях экранов ваш сайт отображался хорошо. Для этого вам придется использовать медиа-запросы. Для крупных шаблонов таких вот запросов может понадобиться очень много, кроме того, вы же еще должны научиться их писать.

В общем, при разработке с нуля адаптивного шаблона вам придется потрудиться как следует, при этом ваша квалификация в верстке должна быть достаточно высокой.

А что же bootstrap? Если изучить этот фреймворк, то он сильно упростит для вас верстку. Во-первых, фреймворк берет на себя кроссбраузерность и адаптивность, а это основные вещи, о которых должен позаботиться разработчик. Но с bootstrap реализовать их очень просто. Это позволяет создать html-шаблон даже человеку, который ранее очень мало занимался верстkой и особо не знаком с css.

Во-вторых, фреймворк идеально подходит при работе в команде. Верстка на bootstrap при должном умении и понимании происходит в 3-5 раз быстрее, а единообразие кода позволит любому вашему коллеге внести правки. Если же мы говорим о верстке без фреймворка, то тут и каждого разработчика может быть свой стиль и другому человеку придется потратить время на изучение его кода.

В качестве преимуществ фреймворка я хотел бы также отметить очень большое русскоязычное сообщество и наличие хорошей документации на нашем языке. Благодаря такой распространенности для Bootstrap появилось много шаблонов, где уже переделан дизайн всех основных элементов. Вы можете подключать такие шаблоны и на их основе делать свои сайты, лишь незначительно что-то меняя.

Недостатки Bootstrap.

По сути, их всего два. Первый – кода обычно в библиотеке написано больше, чем если бы вы написали при разработке с нуля. Потому что, когда вы делаете самостоятельно, вы реализуете только необходимый функционал и все. В Bootstrap же есть все на все случаи жизни. Даже то, что вам может не пригодиться. Но опять же, эта проблема очень легко решается тем, что вы можете сами выбирать, какие компоненты фреймворка загрузить в css-файл. Например, вы вообще можете скачать только сетку, а все остальное делать самостоятельно.

Второй недостаток – шаблонный дизайн. Да, действительно, я часто захожу на разные сайты и вижу там одинаковые кнопки. И я знаю, что они сделаны в Bootstrap, потому что уж слишком это очевидно. Но и эта проблема легко решается, потому что она будет существовать только в том случае, если вы будете использовать только готовые компоненты фреймворка и ничего никогда не кастомизировать под себя.

А если вы, например, подключите только сетку Bootstrap, то сможете воссоздать любой дизайн, при этом пользуясь очень удобной гибкой сеткой фреймворка.

Компоненты фреймворка.

Bootstrap является всеобъемлющим фреймворком. Это означает, что в него заложено много компонентов. По сути, все, что может понадобиться при разработке типовых сайтов.

Это, например, выпадающее меню, кнопки, алерты, табы, индикаторы состояния, хлебные крошки, списки, заголовки и т.д. Если вы писали код для каких-то из этих компонентов, то наверняка знаете, что это делается не за 1 минуту. В Bootstrap же достаточно изучить немного сам фреймворк и все эти вещи вы сможете использовать очень быстро.

Давайте рассмотрим пример с кнопками. Вот такие кнопки очень легко вывести с помощью фреймворка:

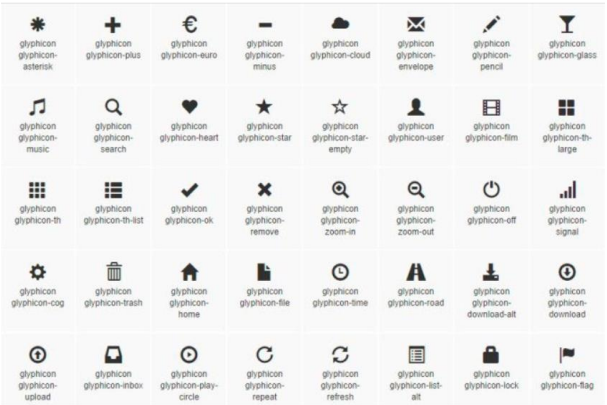
И для этого всего лишь нужен такой код:

```
<button type="button" class="btn btn-default">По умолчанию</button>
<button type="button" class="btn btn-primary">Основной</button>
<button type="button" class="btn btn-success">Успех</button>
<button type="button" class="btn btn-info">Инормирование</button>
<button type="button" class="btn btn-warning">Провал</button>
<button type="button" class="btn btn-danger">Предупреждение</button>
```

Естественно, чтобы все сработало, bootstrap уже должен быть подключен к вашим html-документам. Заметьте, как все работает. Во-первых, у нас есть универсальный класс btn, который определяет общий стили для всех кнопок. Во-вторых, уже непосредственно для дополнительной стилизации используется другой стилевой класс.

Таким образом в Bootstrap выполняется работа и со всеми остальными компонентами.

Отдельно хотел бы выделить такой компонент, как иконочный шрифт. Он очень сильно облегчает вам работу с иконками. А именно, не потребуется загружать эти иконки в виде изображений. По умолчанию в Bootstrap доступно около 200 иконок, вставлять их на веб-страницы очень просто, с примером можно ознакомиться в официальной документации.



Bootstrap: с чего начать работу с фреймворком.

Начать нужно, конечно же, с посещения официального сайта getbootstrap.com. Там все на английском, но на этом сайте мы задержимся ненадолго, а только для того, чтобы перейти на русскоязычный. Для этого в главном

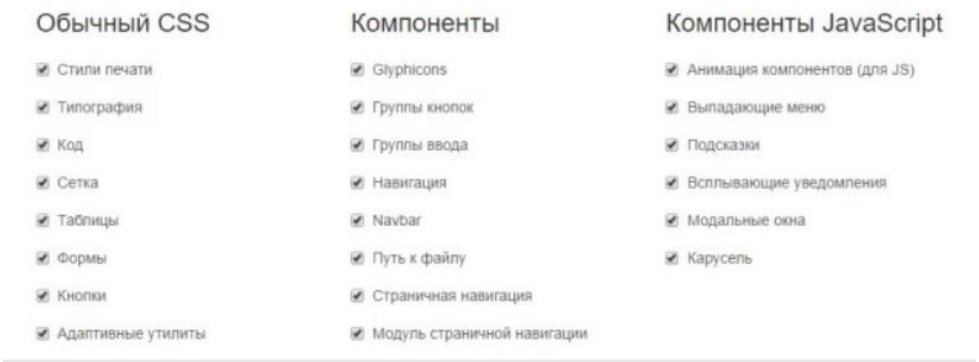
меню перейдите на страницу Getting Started и прокрутите страницу в самый низ. Там вы увидите список переводов на другие языки:

Находите русский язык и кликаете. Все, теперь вы на копии официального сайта bootstrap, переведенной на наш язык. Можете сохранить себе где-нибудь этот адрес, так как вам к нему придется обращаться сотни раз, если планируете активно работать с фреймворком.

Далее вам нужно перейти на страницу “С чего начать” или Getting Started. На ней вам будет предоставлена возможность скачать фреймворк одним из способов, вам подойдет для начала самый первый, то есть простая загрузка полного фреймворка.

Кроме этого, вы можете подключить нужные файлы через CDN. Это означает, что вам их не придется скачивать себе на компьютер, копировать в папку с проектом, а всего лишь нужно прописать в секции head сайта подключение тих файлов из cdn-хранилища.

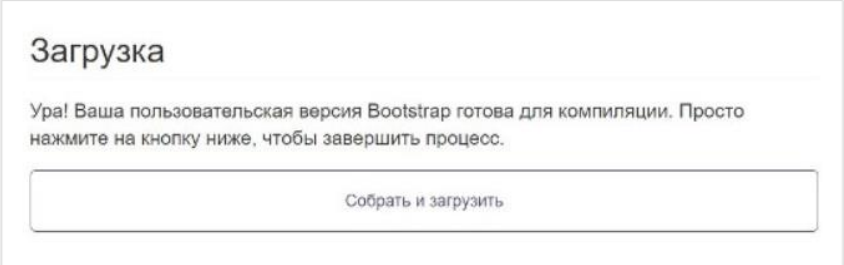
Если же вы хотите кастомизировать фреймворк, то есть использовать только определенные его компоненты, то перейдите на эту страницу — http://getbootstrap.com/customize. На ней вам нужно будет выбрать, какие компоненты включить в состав.



С помощью такого подхода можно избавиться от лишнего кода и оставить только то, что нужно вам. К примеру, вы на своем сайте не используете таблицы, модальные окна и выпадающие меню. Это означает, что совершенно спокойно вы можете отключить эти компоненты и сократить код.

Страница кастомизации очень длинная, на ней вы можете настроить очень много параметров, например, цвета по умолчанию для разных компонентов и т.д. На этой странице можно провести очень много времени, но зато в итоге вы получите уникальную версию Bootstrap, заточенную под ваш проект.

Когда все будет готово, прокрутите страницу в самый низ, где вы увидите кнопку:



Жмем и загружаем свою версию фреймворка.

Фреймворк ANGULAR JS

1. Основные понятия.
2. Работа с Angular JS.

1. Основные понятия

AngularJS — JavaScript-фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC-шаблона, а также упрощение тестирования и разработки.

Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.

История разработки.

AngularJS разработан в 2009 году Мишко Хевери и Адамом Абронсом в Brat Tech LLC как программное обеспечение позади сервиса хранения JSON-данных, измеряющихся мегабайтами, для облегчения разработки корпоративных приложений. Сервис располагался на домене «GetAngular.com» и имел нескольких зарегистрированных пользователей, прежде чем они решили отказаться от идеи бизнеса и выпустить Angular как библиотеку с открытым исходным кодом.

Абронс покинул проект, но Хевери, работающий в Google, продолжает развивать и поддерживать библиотеку с другими сотрудниками Google Игорем Минаром и Войта Джином.

В марте 2014 было объявлено о начале разработки AngularJS 2.0. Новая версия писалась с нуля на TypeScript и очень сильно отличалась от предыдущей, поэтому позже было решено развивать её как отдельный фреймворк с

названием Angular. Angular 2 был выпущен 15 сентября 2016 года, тогда как первая версия продолжила развиваться отдельно как AngularJS.

AngularJS спроектирован с убеждением, что декларативное программирование лучше всего подходит для построения пользовательских интерфейсов и описания программных компонентов, в то время как императивное программирование отлично подходит для описания бизнес-логики. Фреймворк адаптирует и расширяет традиционный HTML, чтобы обеспечить двустороннюю привязку данных для динамического контента, что позволяет автоматически синхронизировать модель и представление. В результате AngularJS уменьшает роль DOM-манипуляций и улучшает тестируемость.

Цели разработки.

1. Отделение DOM-манипуляции от логики приложения, что улучшает тестируемость кода.
2. Отношение к тестированию как к важной части разработки. Сложность тестирования напрямую зависит от структурированности кода.[10][11]
3. Разделение клиентской и серверной стороны, что позволяет вести разработку параллельно.
4. Проведение разработчика через весь путь создания приложения: от проектирования пользовательского интерфейса, через написание бизнес-логики, к тестированию.

Angular придерживается MVC-шаблона проектирования и поощряет слабую связь между представлением, данными и логикой компонентов. Используя внедрение зависимости, Angular переносит на клиентскую сторону такие классические серверные службы, как видозависимые контроллеры. Следовательно, уменьшается нагрузка на сервер и веб-приложение становится легче.

Популярные встроенные Angular-директивы.

С помощью директив AngularJS можно создавать пользовательские HTML-теги и атрибуты, чтобы добавить поведение некоторым элементам.

ng-app

Объявляет элемент корневым для приложения.

ng-bind

Автоматически заменяет текст HTML-элемента на значение переданного выражения.

ng-model

То же, что и ng-bind, только обеспечивает двустороннее связывание данных. Изменится содержимое элемента — ангуляр изменит и значение модели. Изменится значение модели — ангуляр изменит текст внутри элемента.

ng-class

Определяет классы для динамической загрузки.

ng-controller

Определяет JavaScript-контроллер для вычисления HTML-выражений в соответствии с MVC.

ng-repeat

Создает экземпляр DOM для каждого элемента из коллекции.

ng-show и ng-hide

Показывает или скрывает элемент, в зависимости от значения логического выражения.

ng-switch

Создаёт экземпляр шаблона из множества вариантов, в зависимости от значения выражения.

ng-view

Базовая директива, отвечает за обработку маршрутов[17], которые принимают JSON перед отображением шаблонов, управляемых указанными контроллерами.

ng-if

Удаляет или создаёт часть DOM-дерева в зависимости от значения выражения. Если значение выражения, назначенного ngIf, равно false, элемент удаляется из DOM, иначе — вновь клонированный элемент вставляется в DOM.

Также существует возможность создавать настраиваемые директивы, используя в том числе шаблоны в теге script.

Двустороннее связывание данных.

Двустороннее связывание данных в AngularJS является наиболее примечательной особенностью: оно уменьшает количество кода, освобождая сервер от работы с шаблонами. Вместо этого шаблоны отображаются как обычный HTML, наполненный данными, содержащимися в области видимости, определённой в модели. Сервис $scope в Angular следит за изменениями в модели и изменяет раздел HTML-выражения в представлении через контроллер. Кроме того, любые изменения в представлении отражаются в модели. Это позволяет обойти необходимость манипулирования DOM и облегчает инициализацию и прототипирование веб-приложений.

Плагин для Chrome.

В июле 2012 года команда Angular выпустила плагин для браузера Google Chrome под названием Batarang [22], который облегчает отладку веб-приложений, построенных на Angular. Расширение позволяет легко обнаруживать узкие места и предлагает графический интерфейс для отладки приложений.

Версии. Последняя стабильная на данный момент версия AngularJS — 1.7.9, выпущенная в ноябре 2019 года.

Сравнение с Backbone.js

Похожими возможностями обладает Backbone.js — JavaScript-библиотека, основанная на шаблоне проектирования Model-View-Presenter (MVP), предназначена для разработки веб-приложений с поддержкой RESTful JSON интерфейса. Backbone — очень лёгкая библиотека (упакованная и gzip-сжатая по величине ~6.3 Кб), но для работы необходима библиотека Underscore.js, а для поддержки REST API и работы с DOM элементами рекомендуется подключить jQuery-подобную библиотеку: jQuery или Zepto. Backbone.js создан Джереми Ашкенасом, который известен также как создатель CoffeeScript.

Однако, есть и существенные различия:

Связывание данных

Наиболее характерной особенностью, которая разделяет библиотеки, является способ синхронизации модели и представления. В то время как AngularJS поддерживает двустороннее связывание данных, Backbone.js, чтобы связать модель и представление, в значительной мере опирается на шаблонный код.

REST

Backbone.js хорошо поддерживает RESTful-бэкэнд. В AngularJS также очень легко работать с RESTful API при помощи сервиса $resource. В то же время в AngularJS есть более гибкий сервис $http, который подключается к удаленным серверам с помощью браузерного объекта XMLHttpReques или через JSONP.

Шаблоны

В качестве шаблона AngularJS использует комбинацию настраиваемых HTML-тегов и выражений. Backbone.js использует различные шаблонизаторы, такие как Underscore.js.

2. Работа с Angular JS

По большому счету AngularJS является основой, которая связывает HTML-код (который генерируется для просмотра страницы в окне браузера) с JavaScript объектов/моделей. Когда изменяется один из объектов, автоматически обновляется и генерируемая страница. Верно и обратное — модели связаны с текстовым полем (контентом страницы). Когда изменяется контент, это вызывает изменения и в коде сайта.

AngularJS связывает коды в единую систему, и вам не нужно больше обновлять HTML вручную или инспектировать элементы, как в случае, если вы используете JQuery. В самом деле, ни в одном из примеров, приведенных здесь, даже не упоминается JQuery!

Для использования AngularJS вы должны добавить его в код своей страницы. Для этого в тэг нужно прописать соответствующие параметры. Google’s CDN для более быстрой загрузки рекомендую еще такой код:

<script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.0.7/angular.min.js"></script>

AngularJS включает в себя большое количество директив, которые позволяют связать HTML-элементы моделей. Они представляют из себя атрибуты, начинающиеся с префикса ng-, их можно добавлять к любому элементу. Самым важным атрибутом, который, если вы хотите использовать Angular, нужно включить во все страницы сайта является ng- приложений:

<body ng-app>

Атрибут следует добавлять к элементу, который охватывает нужную нам часть страницы: либо все тело страницы, либо ее часть ограниченную тэгами div. Angular ищет его при загрузке страницы и автоматически оценивает все директивы, которые видит, как дочерние элементы.

Меню навигации.

В качестве примера создадим меню навигации, в котором выделяется активный элемент. В примере задействованы только директивы AngularJS, это самое простейшее приложение, использующее фреймворк.

Код выглядит следующим образом:

HTML:

```
<!-- Добавляем атрибут ng-app, который инициализирует запуск AngularJS
-->
<div id="main" ng-app>
<!-- Всем переменным классов в меню навигации будет присвоено значение
"active". Функция $event.preventDefault() выводит страницу, которая была от-
крыта по ссылке. -->
<nav class="{{active}}" ng-click="$event.preventDefault()">
<!-- Когда пункт меню открыт по ссылке, мы устанавливаем активные пе-
ременные -->
<a href="#" class="home" ng-click="active='home'">Home</a>
<a href="#" class="projects" ng-click="active='projects'">Projects</a>
<a href="#" class="services" ng-click="active='services'">Services</a>
<a href="#" class="contact" ng-click="active='contact'">Contact</a>
</nav>
<!-- ng-show выводит элемент, если значение переменной в кавычках со-
ответствует истине. ng- hide – скрывает элемент, если наоборот. Так как из-
начально активная переменная не установлена, то сперва на экране будет виден
следующий текст -->
<p ng-hide="active">Please click a menu item</p>
<p ng-show="active">You chose <b>{{active}}</b></p>
</div>
```

CSS:

```
*{
margin:0;
padding:0;
}
body{
font:15px/1.3 'Open Sans', sans-serif;
color: #5e5b64;
text-align:center;
}
a, a:visited {
outline:none;
color:#389dc1;
}
a:hover{
text-decoration:none;
}
section, footer, header, aside, nav{
display: block;
}
/*--------------------------
Меню
--------------------------*/
nav{
display:inline-block;
margin:60px auto 45px;
background-color:#5597b4;
box-shadow:0 1px 1px #ccc;
border-radius:2px;
}
nav a{
```

```
display:inline-block;
padding: 18px 30px;
color:#fff !important;
font-weight:bold;
font-size:16px;
text-decoration:none !important;
line-height:1;
text-transform: uppercase;
background-color:transparent;
-webkit-transition:background-color 0.25s;
-moz-transition:background-color 0.25s;
transition:background-color 0.25s;
}
nav a:first-child{
border-radius:2px 0 0 2px;
}
nav a:last-child{
border-radius:0 2px 2px 0;
}
nav.home .home,
nav.projects .projects,
nav.services .services,
nav.contact .contact{
background-color:#e35885;
}
p{
font-size:22px;
font-weight:bold;
color:#7d9098;
}
p b{
color:#ffffff;
display:inline-block;
padding:5px 10px;
background-color:#c4d7e0;
border-radius:2px;
text-transform:uppercase;
font-size:18px;
}
```

В приведенном выше примере, используются директивы Angular для задания и считывания активной переменной. Когда она изменяется, это инициирует изменения HTML-кода элемента. В терминологии Angular эта переменная называется моделью. Она доступна для всех директив текущей области. Ее также могут обрабатывать контроллеры (подробнее об этом в следующем примере).

Если вы использовали шаблоны JavaScript раньше, то вы знакомы с синтаксисом команды {{VAR}}. Когда фреймворк видит такую строку, он заменяет содержимое переменной. Эта операция повторяется каждый раз, когда изменяется переменная.

Встроенный редактор.

Для второго примера, создадим простой встроенный редактор – при нажатии пункта меню всплывает небольшое текстовое поле с подсказкой. Используем контроллер, который будет инициализировать модели и задавать два разных метода отображения подсказки. Контроллеры являются стандартными функциями JavaScript, которые автоматически выполняются фреймворком

Angular. Они связаны с кодом отображения страницы вашего сайта через директивы ng-controller.

HTML:

```
<!-- Когда элемент выбран, всплывающая подсказка скрывается-->
<div id="main" ng-app ng-controller="InlineEditorController" ng-click="hideTooltip()">
    <!-- Это всплывающая подсказка. Она показывается только, когда значение переменной "showtooltip" – «истина» -->
    <div class="tooltip" ng-click="$event.stopPropagation()" ng-show="showtooltip">
        <!-- ng-модель связывает содержание текстового поля с моделью "value".
         Любые изменения текстового поля будут автоматически задаваться, как значение этой модели, а также вызывать изменения других элементов страницы, связанных с ней. -->
        <input type="text" ng-model="value" />
    </div>
    <!-- Выбор метода отображения подсказки из вариантов заданных в InlineEditorController (контроллере встроенного редактора), он зависит от значения переменной "showtooltip". -->
    <p ng-click="toggleTooltip($event)">{{value}}</p>
</div>
```

JS:

```
// Контроллер – стандартная функция. Она инициируется, когда
AngularJS при обработке кода находит атрибут ng-controller.
function InlineEditorController($scope){
    // $scope – специальный объект, который задает параметры отображения
    // переменной. Здесь вы можете задать некоторые значения по умолчанию
    $scope.showtooltip = false;
    $scope.value = 'Edit me.';
    // Некоторые вспомогательные функции, которые доступны после инициации    // Angular.
    $scope.hideTooltip = function(){
        // Когда значение модели меняется, AngularJS автоматически вносит        // изменения в формат вывода. И всплывающее меню скрывается с экрана.
        $scope.showtooltip = false;
    }
    $scope.toggleTooltip = function(e){
        e.stopPropagation();
        $scope.showtooltip = !$scope.showtooltip;
    }
}
```

CSS:

```
*{
    margin:0;
    padding:0;
}
body{
    font:15px/1.3 'Open Sans', sans-serif;
    color: #5e5b64;
    text-align:center;
}
a, a:visited {
    outline:none;
    color:#389dc1;
}
```

```
a:hover{
	text-decoration:none;
}
section, footer, header, aside, nav{
	display: block;
}
/*-------------------------
	Всплывающее меню редактора.
-------------------------*/
.tooltip{
	background-color:#5c9bb7;
	background-image:-webkit-linear-gradient(top, #5c9bb7, #5392ad);
	background-image:-moz-linear-gradient(top, #5c9bb7, #5392ad);
	background-image:linear-gradient(top, #5c9bb7, #5392ad);
	box-shadow: 0 1px 1px #ccc;
	border-radius:3px;
	width: 290px;
	padding: 10px;
	position: absolute;
	left:50%;
	margin-left:-150px;
	top: 80px;
}
.tooltip:after{
	/* The tip of the tooltip */
	content:'';
	position:absolute;
	border:6px solid #5190ac;
	border-color:#5190ac transparent transparent;
	width:0;
	height:0;
	bottom:-12px;
	left:50%;
	margin-left:-6px;
}
.tooltip input{
	border: none;
	width: 100%;
	line-height: 34px;
	border-radius: 3px;
	box-shadow: 0 2px 6px #bbb inset;
	text-align: center;
	font-size: 16px;
	font-family: inherit;
	color: #8d9395;
	font-weight: bold;
	outline: none;
}
p{
	font-size:22px;
	font-weight:bold;
	color:#6d8088;
	height: 30px;
	cursor:default;
}
p b{
	color:#ffffff;
	display:inline-block;
	padding:5px 10px;
	background-color:#c4d7e0;
	border-radius:2px;
```

```
      text-transform:uppercase;
      font-size:18px;
}

p:before{
      content:'✎';
      display:inline-block;
      margin-right:5px;
      font-weight:normal;
      vertical-align: text-bottom;
}
#main{
      height:300px;
      position:relative;
      padding-top: 150px;
}
```

Когда функция контроллера запускается на исполнение, для нее в качестве параметра задается специальный объект $scope. Он отвечает за ввод текста в текстовый редактор. Для того, чтобы вывести его на экран, нужно прописать дополнительные свойства и функции, которые описывают отображение его элементов. С помощью NG-моделей осуществляется связь кода сайта с текстом, который вводится в рабочее поле редактора. При вводе текста Angular задает соответствующие изменения переменных.

Форма заказа.

В этом примере рассмотрим код формы заказы, в которой автоматически выводится общая сумма. Здесь использована еще одна полезная функция AngularJS — фильтры. Фильтры позволяют вносить изменения в модели, а также объединять их с помощью символа «|». В приведенном ниже примере, используется фильтр валюты, чтобы перевести числовое значение в корректный формат цены — с разделением долларов и центов. Когда вы рассмотрите пример № 4, вы сможете с легкостью задавать свои собственные фильтры.

HTML:

```
<!-- Объявление нового приложения AngularJS и связанного с ним контроллера -->
<form ng-app ng-controller="OrderFormController">
    <h1>Services</h1>
    <ul>
        <!-- Запуск цикла обработки массива с перечнем услуг, назначается при нажатии элемента. При этом если нужно устанавливается активный css-класс. -->
        <li ng-repeat="service in services" ng-click="toggleActive(service)" ng-class="{active:service.active}">
            <!-- Обратите внимание на использование фильтра валюты, он задает формат вывода цены. -->
            {{service.name}} <span>{{service.price | currency}}</span>
        </li>
    </ul>
    <div class="total">
        <!-- Подсчет общей стоимости всех выбранных услуг. Выводится по заданному формату валюты. -->
        Total: <span>{{total() | currency}}</span>
    </div>
```

JS:

```
function OrderFormController($scope){
  // Определение параметров модели. Вид отображения будет зависеть от
```

```
  // обработки циклом массива перечня услуг, для которых автоматически
будет  // сгенерирован формат вывода списком через тэг li.
  $scope.services = [
          {
                  name: 'Web Development',
                  price: 300,
                  active:true
          },{
                  name: 'Design',
                  price: 400,
                  active:false
          },{
                  name: 'Integration',
                  price: 250,
                  active:false
          },{
                  name: 'Training',
                  price: 220,
                  active:false
          }
  ];
  $scope.toggleActive = function(s){
          s.active = !s.active;
  };
  // Вспомогательный метод подсчета общей суммы по всем выбранным
  // позициям.
  $scope.total = function(){
          var total = 0;
          // Use the angular forEach helper method to
          // loop through the services array:
          angular.forEach($scope.services, function(s){
                  if (s.active){
                          total+= s.price;
                  }
          });
          return total;
  };
}
```

CSS:

```
*{
  margin:0;
  padding:0;
}
body{
  font:15px/1.3 'Open Sans', sans-serif;
  color: #5e5b64;
  text-align:center;
}
a, a:visited {
  outline:none;
  color:#389dc1;
}
a:hover{
  text-decoration:none;
}
section, footer, header, aside, nav{
  display: block;
}
/*--------------------------
  Форма заказа
```

```
-------------------------*/
form{
  background-color: #61a1bc;
  border-radius: 2px;
  box-shadow: 0 1px 1px #ccc;
  width: 400px;
  padding: 35px 60px;
  margin: 50px auto;
}
form h1{
  color:#fff;
  font-size:64px;
  font-family:'Cookie', cursive;
  font-weight: normal;
  line-height:1;
  text-shadow:0 3px 0 rgba(0,0,0,0.1);
}
form ul{
  list-style:none;
  color:#fff;
  font-size:20px;
  font-weight:bold;
  text-align: left;
  margin:20px 0 15px;
}
form ul li{
  padding:20px 30px;
  background-color:#e35885;
  margin-bottom:8px;
  box-shadow:0 1px 1px rgba(0,0,0,0.1);
  cursor:pointer;
}
form ul li span{
  float:right;
}
form ul li.active{
  background-color:#8ec16d;
}
div.total{
  border-top:1px solid rgba(255,255,255,0.5);
  padding:15px 30px;
  font-size:20px;
  font-weight:bold;
  text-align: left;
  color:#fff;
}
div.total span{
  float:right;
}
```

Связка ng-repeat (описание) – это еще один полезный элемент Angular. Она позволяет запустить цикл обработки массива элементов, а также задать разметку для каждого из них. Она автоматически обновляет код, если один из элементов был изменен или удален.

Фреймворк JQUERY

1. Основные понятия.
2. Работа с jQuery.

1. Основные понятия

jQuery — набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Разработка jQuery ведётся командой добровольцев на пожертвования.

История создания.

HTML был одной из первых вещей, которую Джон Резиг освоил, когда он только начал заниматься программированием. Резиг программировал на QBasic, когда один его знакомый показал ему, как создать веб-страницу (используя Angelfire), а также основы HTML. Отец подарил ему на Рождество две книги по HTML. Именно тогда, когда он только начал программировать на Visual Basic, HTML и веб-дизайн очень заинтересовали его.

Но страсть к JavaScript пришла значительно позже, примерно в 2004 году. Тогда Резиг получал степень в области компьютерных наук и работал на полставки в местной фирме Brand Logic. Он занимался дизайном сайта, в котором создавался пользовательский скроллинг. Джон был разочарован и расстроен, особенно потому, что использовал код других разработчиков, после чего решил серьёзно изучить JavaScript. Изучив, пришёл к выводам, что JavaScript — это простой, но изящный язык, который является невероятно мощным для решения многих задач. В течение следующей пары лет Джон создал множество различных JavaScript-приложений, прежде чем закончить создание jQuery. Основной целью создания jQuery Резиг видел возможность закодировать многоразовые куски кода, которые позволят упростить JavaScript и использовать их так, чтобы не беспокоиться о кросс-браузерных вопросах. Библиотека была представлена общественности на компьютерной конференции «BarCamp» в Нью-Йорке в 2006 году.

Возможности.

- движок кросс-браузерных CSS-селекторов Sizzle[4], выделившийся в отдельный проект;
- переход по дереву DOM, включая поддержку XPath как плагина;
- события;
- визуальные эффекты;
- AJAX-дополнения;
- JavaScript-плагины.

Философия.

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки управление передаётся

JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функциональность, полезную для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, бóльшая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно ту JavaScript-функциональность, которая на нём была бы востребована.

Использование.

jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл:

```
<head>
      <script src="jquery-2.2.2.min.js">
      </script>
   </head>
```

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким, можно писать его следующим образом:

```
jQuery(function($) {
  // здесь код скрипта, где в $ будет находиться объект, предоставляющий доступ к функциям jQuery
})
```

Работу с jQuery можно разделить на 2 типа:

1. Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.
2. Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора CSS, что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery. Например,

```
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
```

находит все элементы div с классом test, а также все элементы p с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз.

Здесь методы add, addClass и slideDown возвращают ссылку на исходный объект $("div.test"), поэтому возможно вести такую цепочку.

Методы, начинающиеся с $., удобно применять для обработки глобальных объектов. Например:

```
$.each([1,2,3], function() {
  document.write(this + 1);
});
```

добавит на страницу `234`.

`$.ajax` и соответствующие функции позволяют использовать методы AJAX. Например:

```
$.ajax({
  type: "POST",
  url: "some.php",
  data: {name: 'John', location: 'Boston'},
  success: function(msg){
    alert( "Data Saved: " + msg );
  }
});
```

В этом примере идет обращение к скрипту `some.php` с параметрами `name=John&location=Boston`, и полученный результат выдается в сообщении посредством `alert()`.

Пример добавления к элементу обработчика события click с помощью jQuery:

```
$("a").click(function() {
    alert("Hello world!");
  });
```

В данном случае при нажатии на элемент <a> происходит вызов `alert("Hello world!")`.

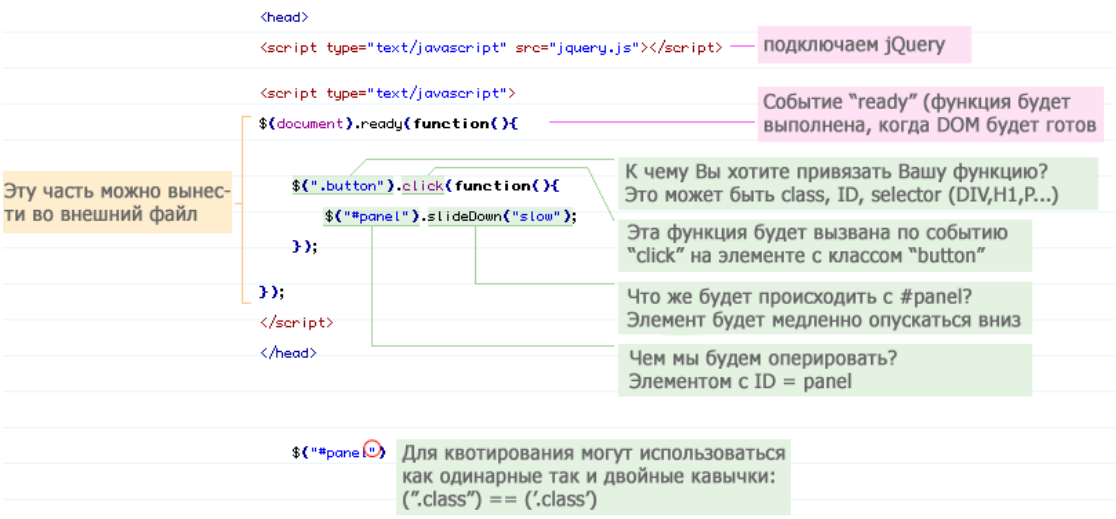
2. Работа с jQuery

jQuery — это замечательный JavaScript Framework, который подкупает своей простотой в понимании и удобством в использовании. Но изучение надо с чего-то начинать, и лично моё мнение — лучше всего начинать с наглядных примеров, и они далее.

Для начала Вам понадобиться сам фреймворк, его вы сможете скачать с домашней страницы проекта, затем проинициализировать:

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

Основные моменты поможет понять следующая диаграмма:



Как получить элемент с помощью jQuery.

Для того чтобы понимать, как работает селектор Вам все-же необходимы базовые знания CSS, т.к. именно от принципов CSS отталкивает селектор jQuery:

– $("#header") — получение элемента с id=«header»;

– $(«h3») — получить все <h3> элементы;

– $(«div#content.photo») — получить все элементы с классом =«photo» которые находятся в элементе div с id=«content»;

– $(«ul li») — получить все <li> элементы из списка <ul>;

– $(«ul li:first») — получить только первый элемент <li> из списка <ul>;

Литература

1. Богун В.В. Сетевые технологии. Организация интерактивности в рамках статических Интернет-сайтов: учеб. пособие. Саратов: Ай Пи Ар Медиа, 2020. 65 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/92640.html (дата обращения: 20.01.2022).
2. Бойкова М.В. Совершенствование теории администрирования: монография. М: Российская таможенная академия, 2020. 96 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/69790.html (дата обращения: 20.01.2022).
3. Власов Ю.В., Рицкова Т.И. Администрирование сетей на платформе MS Windows Server: учеб. пособие. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 622 с. Текст : электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/97536.html (дата обращения: 20.01.2022).
4. Горелов С.В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 т. Т. I: учебник. М.: Прометей, 2019. 362 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/94532.html (дата обращения: 20.01.2022).
5. Горелов С.В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 т. Т. II : учебник. М.: Прометей, 2019. 378 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/94533.html (дата обращения: 20.01.2022).
6. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем: учеб. пособие. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 299 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/97577.html (дата обращения: 20.01.2022).
7. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Управление внедрением информационных систем: учеб. пособие. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. 277 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/102073.html (дата обращения: 20.01.2022).
8. Гунько А.В. Программирование (в среде Windows): учеб. пособие. Новосибирск: Новосибирский государственный технический университет, 2019. 155 с. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/99209.html (дата обращения: 20.01.2022).

9. Джон Роббинс Отладка Windows-приложений. Саратов: Профобразование, 2017. 447 c. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. - Режим доступа: URL: https://www.iprbookshop.ru/63940.html (дата обращения: 20.01.2022).

10. Ершова Н.Ю., Соловьев А.В. Организация вычислительных систем: учеб. пособие. М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. 221 c. Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. Режим доступа: - URL: https://www.iprbookshop.ru/102024.html (дата обращения: 20.01.2022).

Учебное издание

Федькова Н.А.

Современные технологии разработки программного обеспечения

Методическое пособие

Редактор Адылина Е.С.

Подписано к печати 21.11.2022 г. Формат 60x84 $^{1}/_{16}$.
Бумага офсетная. Усл. п. л. 3,37. Тираж 25 экз. Изд. №7421

Издательство Брянского государственного аграрного университета
243365 Брянская обл., Выгоничский район, с. Кокино, Брянский ГАУ