Портфолио (структура удачных предметов)

1. Программирование (3-7 семестры)
1.1. Ссылки на Google Drive
1.2. Ссылки на Github (репозитории и просто отдельные папки)
1.3. Борды в repl.it
1.4. Борды в Google Colaboratory

2. Практики
2.1. Gitlab
2.2. Github

3. Веб-проектирование и веб-языки + Серверные веб-технологии 4 семестр
3.1. Ссылка на прямое скачивание с Github
3.2. Прямой просмотр XML/json/pdf
3.3. Ссылки на сторонние ресурсы

4. Компьютерный практикум
4.1. Ссылка на работу в кодакторе

5. Основы компьютерной алгебры
5.1. Прямой просмотр файлов

6. Дисциплина "Информационные средства и технологии инженерных и научных расчетов (7 семестр)"
6.1. Встраивание интерактивных элементов

Как в Moodle называются компоненты курса (в англоязычным варианте, единственном числе), которые могут вступать во взаимодействие с пользователем (чат, SCORM-пакет и т.п.)?
Ответ: activity
На каком языке серверного веб-программирования написана среда Moodle?
Ответ: PHP
Какая статья Закона об образовании РФ определяет понятие электронного обучения?
Ответ: 16
Как по-другому называется безымянная функция (см. пример ниже), что отсылает к функциональному исчислению, созданному Алонзо Чёрчем?
(x=>x*x)(5)
Ответ:λ
Какой тип структуры возвращает метод document.querySelectorAll?
Ответ: NodeList
Как в JavaScript называется функция, объявляемая со знаком звёздочки, вызов которой создаёт экземпляр итератора?
Ответ: генератор
Назовите имя файла в составе SCORM-пакета, который содержит перечисление использованных в пакете ресурсов и в том числе позволяет определить порядок их предъявления.
Ответ: imsmanifest.xml
Как называется папка в Moodle, которая хранит динамическую информацию, выходящую за пределы хранимого в базе данных, недоступная для доступа из веба?
Ответ: moodledata
Назовите расширение файла, содержащего ES2015-модуль (т.е. модуль нового типа, позволяющий использовать import/export)
Ответ: mjs
В браузерах каким должно быть значение атрибута type элемента script, чтобы в содержимом этого элемента можно было импортировать экспортированные другим сценарием сущности?
Ответ: module
Из списка выберите стандарты электронного обучения
Выберите один или несколько ответов:
SCORM
IMS
Откройте борд
http://kodaktor.ru/g/css_724cd
Ответ: 20
Назовите формат, в котором сохраняются СКОРМ-пакеты (расширение файлов)
Ответ:zip
Документ, выдаваемый от имени государства лицу, подавшему заявку в установленном законом порядке, в подтверждение его прав на изобретение - это ... (1 слово, именительный падеж, единственное число)
Ответ: Патент
Предварительная публикация текста научных исследований, статьи или доклада средствами оперативной полиграфии небольшим тиражом, до выхода в свет издания, в котором они могут быть помещены - это ... (1 слово, именительный падеж, единственное число)
Ответ:препринт
Что, согласно ГОСТ Р 53620-2009, определяется как образовательный ресурс, представленный в

электронно-цифровой форме и включающий в себя структуру, предметное содержание и метаданные о них? Впечатайте аббревиатуру.
Ответ: эор
Какой аббревиатурой обозначается база данных, которая принимает и отдает по запросу информацию об учебном опыте в соответствии со спецификацией xAPI?
Ответ: LRS
Какой код ответа по протоколу HTTP означает, что новый ресурс успешно создан?
Ответ: 201
Как называется в англоязычном варианте расширение CMS, дополнительный программный модуль, интегрируемый в основной движок системы для выполнения определённой задачи, не предусмотренной базовым функционалом ядра (например, Akismet)?
Ответ: plug-in
Как называется первый уровень стандарта кодинга на PHP?
Ответ: PSR-1

1. IP-адресация в Интернете и локальных сетях. Маска подсети. Статическое и динамическое назначение IP-адресов.

IP-адрес – это уникальный адрес, идентифицирующий устройство в интернете или локальной сети. IP означает «Интернет-протокол» – набор правил, регулирующих формат данных, отправляемых через интернет или локальную сеть. По сути, IP-адрес – это идентификатор, позволяющий передавать информацию между устройствами в сети: он содержит информацию о местоположении устройства и обеспечивает его доступность для связи.

IP-адрес работает по протоколу TCP/IP. TCP/IP — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю.

IP-адрес представляет собой 32-битовое (по версии IPv4) или 128-битовое (по версии IPv6) двоичное число. Удобной формой записи IP-адреса (IPv4) является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками. 4 разряда также называются октетами.

Маска подсети — битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу самого узла в этой сети.

У всех IP адресов есть две части сеть и узел.

Сеть – это та часть IP, которая не меняется во всей сети и все адреса устройств начинаются именно с номера сети.

Узел – это изменяющаяся часть IP. Каждое устройство имеет свой уникальный адрес в сети, он называется узлом.

IP-адрес называют статическим (постоянным, неизменяемым), если он назначается пользователем в настройках устройства, либо назначается автоматически при подключении устройства к сети и не может быть присвоен другому устройству.

IP-адрес называют динамическим (непостоянным, изменяемым), если он назначается автоматически при подключении устройства к сети и используется в течение ограниченного промежутка времени, указанного в сервисе назначавшего IP-адрес (DHCP).

Статический IP-адрес представляет собой адрес, который постоянно назначается сетевым устройством путем ISP (Интернет провайдером) и не меняется, даже если устройство перезагружается. Статические IP-адреса обычно имеют две версии: IPv4 и IPv6. Статический IP-адрес обычно назначается серверу хостинга сайтов и предоставляет услуги электронной почты, VPN и FTP.

Назначая статический IP-адрес, системе сообщается, что она должна использовать именно этот IP-адрес, а также указывается маска подсети для этого IP-адреса и при необходимости основной шлюз (шлюз по умолчанию), используемый для межсетевых соединений. Настроив эти параметры IP, нужно настроить и параметры разрешения имен через DNS (Domain Name System) и, возможно, через WINS (Windows Internet Name Service).

Динамический IP-адрес — это адрес, который постоянно меняется. Чтобы создать динамические IP-адреса, сеть должна иметь настроенный и работающий DHCP-сервер. DHCP-сервер назначает свободный IP-адрес всем устройствам, подключенным к сети. DHCP — это способ динамического и автоматического назначения IP-адресов сетевым устройствам в

физической сети. Он предлагает автоматизированный способ распространения и обновления IP-адресов и другой информации о конфигурации по сети.

2. XML и основанные на XML языки разметки: применение на примере RSS.

XML – это язык разметки, созданный для определения синтаксиса кодирования документов, которые могут быть прочитаны людьми и машинами. Он делает это с помощью тегов, которые определяют структуру документа, а также то, как документ должен храниться и транспортироваться. У XML нет предопределенного языка разметки, как у HTML. Вместо этого, XML позволяет пользователям создавать свои собственные символы разметки для описания контента, формируя неограниченный и самоопределяемый набор символов.

Словари, основанные на XML (например, RSS), сами по себе формально описаны на том же XML, что позволяет программно изменять и проверять документы на основе этих словарей, не зная их семантики, то есть не зная смыслового значения элементов.

XML-документ состоит из директив, элементов, атрибутов, комментариев и специальных символов.

Любой XML документ начинается с директивы, указывается версия и кодировка <?xml version="1.0" encoding="UTF-8 ?>

Элементами являются теги, пример: <author> </author>

Любой элемент может иметь атрибуты, содержащие дополнительную информацию об элементе. Атрибуты всегда включаются в начальный тег элемента и имеют вид: <author country="RUS"> </author>

Элементы должны либо следовать друг за другом, либо быть вложены один в другой

XML-документы могут содержать комментарии, которые игнорируются приложением, обрабатывающим документ.

RSS – это специальный файл в формате (rss или xml), который используется для описания новостей сайтов или их анонсов со ссылкой на полную версию текста.

Обычно с помощью RSS 2.0 даётся краткое описание новой информации, появившейся на сайте, и ссылка на её полную версию. Интернет-ресурс в формате RSS называется RSS-каналом, RSS-лентой или RSS-фидом.

Многие современные браузеры, почтовые клиенты и интернет-пейджеры умеют работать с RSS-лентами, среди них Safari, Maxthon, Miranda, Mozilla Firefox (до Firefox 63), Mozilla Thunderbird, Opera, Opera Mini, Microsoft Internet Explorer (начиная с 7-й версии), Yandex Browser. Кроме того, существуют специализированные приложения (RSS-агрегаторы), собирающие и обрабатывающие информацию RSS-каналов.

3. Веб-технологии на стороне клиента. ECMAScript и ES.Next. Транспиляция.

Web-технологиями является весь набор средств, позволяющих организовать WWW (World Wide Web), то есть всемирную паутину. Так как каждый сеанс является взаимодействием двух сторон, а именно, сервера и клиента, то и Web-технологии делятся на следующие группы:

Технологии серверной стороны (server-side)

Технологии клиентской стороны (client-side)

Технологии клиентской стороны включают в свой состав весь набор технологий по созданию веб-страниц (HTML, JavaScript, CSS и другие). То есть Клиент - это то, с чем взаимодействует пользователь. «клиентский» код отвечает за большую часть того, что на самом деле видит пользователь. Это включает в себя:

Определение структуры веб-страницы

Настройка внешнего вида веб-страницы

Реализация механизма пользовательского взаимодействия (нажатие кнопок, ввод текста и т.д.)

Web-страницы - это текстовые файлы, содержащие собственно текст

содержимого страницы и команды форматирования, называемые тегами

(tag) или дескрипторами разметки языка HTML. Язык разметки гипертекста HTML (HyperText Markup Language) является базовой технологией

разработки Web-страниц, которые также называют HTML-документами.

Код HTML интерпретируется браузером (средством просмотра web-страниц), который выводит отформатированное содержимое Web-страницу

на экран. Чтобы определить внешний вид веб-страницы, веб-разработчики используют CSS , который расшифровывается как каскадные таблицы стилей (Cascading Style Sheets). CSS - это язык, который позволяет описать стиль элементов, определенных в HTML, позволяя изменять шрифт, цвет, макет, простые анимации и другие поверхностные элементы. Для реализации механизма взаимодействия с пользователем обычно используется язык программирования JavaScript. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

JavaScript - это скриптовый язык общего назначения, соответствующий спецификации ECMAScript. ECMAScript — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков, он был стандартизирован международной организацией ECMA в спецификации ECMA-262. Новые релизы EcmaScript выходят ежегодно и содержат возможности, которыми можно пользоваться уже сегодня при помощи транспилятора, например, Babel. ES.Next - это динамическое имя, которое относится к любой следующей версии на момент написания.. По сути это динамическое имя которое относится к любой следующей версии ECMAScript на момент написания.

Трансляция — это преобразование исходного кода программы из одного языка программирования в любой другой. Трансляцией языка занимается программа транслятор.

В общем понимании транспилятор или транспайлер (англ. transpiler), это программа (тип компилятора), которая переводит исходный код одного языка программирования высокого уровня в исходный код другого языка программирования высокого уровня. Получается любой транспилятор является, в то же время, транслятором, но не наоборот. А вот преобразованием программного кода в машинный занимается компилятор. Он осуществляет трансляцию программы с предметно-ориентированного языка на машинно-ориентированный язык. То есть компилятор — это транслятор, который осуществляет перевод исходной програм-мы в эквивалентную ей объектную программу на языке машинных команд или на языке ассемблера.

4. Веб-технологии на стороне клиента. Распространенные браузеры и их особенности. Язык HTML5.

Web-технологиями является весь набор средств, позволяющих организовать WWW (World Wide Web), то есть всемирную паутину. Так как каждый сеанс является взаимодействием двух сторон, а именно, сервера и клиента, то и Web-технологии делятся на следующие группы:

Технологии серверной стороны (server-side)

Технологии клиентской стороны (client-side)

Технологии клиентской стороны включают в свой состав весь набор технологий по созданию веб-страниц (HTML, JavaScript, CSS и другие). То есть Клиент — это то, с чем взаимодействует пользователь. «клиентский» код отвечает за большую часть того, что на самом деле видит пользователь. Это включает в себя:

- Определение структуры веб-страницы
- Настройка внешнего вида веб-страницы
- Реализация механизма пользовательского взаимодействия (нажатие кнопок, ввод текста и т.д.)

Web-страницы — это текстовые файлы, содержащие собственно текст

содержимого страницы и команды форматирования, называемые тегами

(tag) или дескрипторами разметки языка HTML. Язык разметки гипертекста HTML (HyperText Markup Language) является базовой технологией

разработки Web-страниц, которые также называют HTML-документами.

Код HTML интерпретируется браузером (средством просмотра webстраниц), который выводит отформатированное содержимое Web-страницу

на экран. Чтобы определить внешний вид веб-страницы, веб-разработчики используют CSS , который расшифровывается как каскадные таблицы стилей (Cascading Style Sheets). CSS - это язык, который позволяет описать стиль элементов, определенных в HTML, позволяя изменять шрифт, цвет, макет, простые анимации и другие поверхностные элементы. Для реализации механизма взаимодействия с пользователем обычно используется JavaScript как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Браузеры созданы для пользователей интернета, чтобы в онлайн режиме они могли общаться между собой, смотреть видеоролики, слушать аудиозаписи, искать нужную информацию, играть в игры, пользоваться приложениями и решать многие другие задачи. Сегодня в интернете доступно большое количество браузеров.

Есть несколько общих критериев, на которые пользователю стоит обратить внимание при выборе браузера:

- Безопасность.
- Скорость загрузки страниц.
- Использование оперативной памяти.
- Удобство настроек.
- Юзабилити. Важно, чтобы навигация и функционал программы были интуитивно понятными даже для непродвинутых пользователей интернета.
- Наличие плагинов и расширений.
- Кроссплатформенность.

ТОП Лучших Браузеров 2021 года

Google Chrome – высокоскоростной популярный браузер

Яндекс.Браузер – современный быстрый браузер от российских разработчиков

Mozilla Firefox – браузер с возможностью детальной настройки с учетом потребностей пользователя

Microsoft Edge – системный браузер Windows с режимом чтения

Opera – быстрый браузер со встроенным VPN и блокировщиком рекламы

Safari – браузер от Apple с возможностью установки на Windows.

Цель разработки HTML5 — улучшение уровня поддержки мультимедиатехнологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека. В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул.

5. Веб-технологии на стороне сервера. Суть и назначение AJAX Fetch API.

Web-технологиями является весь набор средств, позволяющих организовать WWW (World Wide Web), то есть всемирную паутину. Так как каждый сеанс является взаимодействием двух сторон, а именно, сервера и клиента, то и Web-технологии делятся на следующие группы:

Технологии серверной стороны (server-side)

Технологии клиентской стороны (client-side)

Серверные программы обеспечивают предоставление тех или иных ресурсов клиентским программам. Клиенты, когда им требуется какой-либо файл или просто какая-то информация от сервера, вырабатывают специальный запрос клиента и отправляют его серверу. Серверная программа выполняет обработку запроса и отправляет ответ сервера, который содержит запрошенную информацию или же извещение об ошибке, в случае недоступности требуемых данных.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль. Примерами таких технологий являются:

Nodejs — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), она превращает JavaScript из узкоспециализированного языка в язык общего назначения благодаря тому, что добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Java - строго типизированный объектно-ориентированный язык программирования общего назначения.

PHP - скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Python - Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным - все является объектами. Для разработки сайтов на Python используются различные фреймворки: Django, Flask, Tornado и другие.

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый Ajax. При использовании Ajax страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

AJAX — это аббревиатура, которая означает Asynchronous Javascript and XML. На самом деле, AJAX не является новой технологией, так как и Javascript, и XML существуют уже довольно продолжительное время, а AJAX — это синтез обозначенных технологий.

Fetch API — это современный подход для создания асинхронных запросов. Fetch API предоставляет интерфейс JavaScript для работы с запросами и ответами HTTP. Он также предоставляет глобальный метод fetch(), который позволяет легко и логично получать ресурсы по сети асинхронно.

При использовании AJAX Fetch API нет необходимости обновлять каждый раз всю страницу, так как обновляется только ее конкретная часть. Это намного удобнее, так как не приходится долго ждать, и экономичнее, так как не все обладают безлимитным интернетом. Правда в этом случае, разработчику необходимо следить, чтобы пользователь был в курсе того, что происходит на странице.

достоинства AJAX Fetch API :

Возможность создания удобного Web-интерфейса

Активное взаимодействие с пользователем

Частичная перезагрузка страницы, вместо полной

Удобство использования

AJAX Fetch API использует два метода работы с веб-страницей: изменение Web-страницы не перезагружая её, и динамическое обращение к серверу.

Для обращения к серверу Fetch API использует метод fetch().

Браузер сразу же начинает запрос и возвращает промис, который внешний код использует для получения результата.

Промис завершается с ошибкой, если fetch не смог выполнить HTTP-запрос, например при ошибке сети или если нет такого сайта. HTTP-статусы 404 и 500 не являются ошибкой.

Мы можем увидеть HTTP-статус в свойствах ответа:

status – код статуса HTTP-запроса, например 200.

ok – логическое значение: будет true, если код HTTP-статуса в диапазоне 200-299.

6. Веб-технологии на стороне сервера. Node.js и PHP, Python. Методы GET и POST. Формы.

Web-технологиями является весь набор средств, позволяющих организовать WWW (World Wide Web), то есть всемирную паутину. Так как каждый сеанс является взаимодействием двух сторон, а именно, сервера и клиента, то и Web-технологии делятся на следующие группы:

Технологии серверной стороны (server-side)

Технологии клиентской стороны (client-side)

Серверные программы обеспечивают предоставление тех или иных ресурсов клиентским программам. Клиенты, когда им требуется какой-либо файл или просто какая-то информация от сервера, вырабатывают специальный запрос клиента и отправляют его серверу. Серверная программа выполняет обработку запроса и отправляет ответ сервера, который содержит запрошенную информацию или же извещение об ошибке, в случае недоступности требуемых данных.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль. Примерами таких технологий являются:

Node.js (или просто Node) — это серверная платформа для работы с JavaScript через движок V8. JavaScript выполняет действие на стороне клиента, а Node — на сервере. С помощью Node можно писать полноценные приложения. Node умеет работать с внешними библиотеками, вызывать команды из кода на JavaScript и выполнять роль веб-сервера. С помощью Node код

обращается к жесткому диску, базам данных и Сети. Это делает возможным написание абсолютно любых приложений: от утилит командной строки и видеоигр до полноценных веб-серверов.

PHP - скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. PHP — это язык серверной стороны, то PHP-файлы (например, index.php) преобразуются на сервере в HTML, перед тем, как быть отправленными в браузер. По этой причине вы должны размещать PHP-файлы на сервере во время работы с ними. Это может быть либо удаленный сервер, либо локальный сервер на вашем компьютере.

Python - Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Для разработки сайтов на Python на стороне сервера используются различные фреймворки: Django, Flask, Tornado и другие.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Два часто используемых метода для запроса-ответа между клиентом и сервером: GET - запрашивает данные из указанного ресурса; POST - отправляет данных, подлежащие обработке, на указанный ресурс. Метод запроса POST запрашивает веб-сервер на прием и хранение данных, заключенные в тело сообщения запроса. Часто используется при загрузке файла или при отправке заполненной веб-формы. Метод запроса HTTP GET извлекает информацию с сервера. В рамках запроса GET некоторые данные могут передаваться в строке запроса URL-адреса, указывая условия поиска, диапазоны дат или другую информацию, которая определяет запрос.

При отправки конфиденциальных данных - платежных данных, паролей, на сервере как правило используются специальные алгоритмы шифрования, которые помещают эти данные в базу данных в зашифрованном виде. Шифрование — обратимое преобразование информации в целях сокрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней. Главным образом, шифрование служит задачей соблюдения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.

Для того чтобы создать между сервером и браузером постоянное соединение используется протокол WebSocket («веб-сокет»), описанный в спецификации RFC 6455. Данные передаются по нему в обоих направлениях в виде «пакетов», без разрыва соединения и дополнительных HTTP-запросов.

WebSocket особенно хорош для сервисов, которые нуждаются в постоянном обмене данными, например онлайн игры, торговые площадки, работающие в реальном времени, и т.д.

При обмене данными между клиентом и сервером используются два основных формата данных - XML и JSON.

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования.

7. ГОСТ на создание жизненного цикла программного изделия.

ДЕЙСТВУЮЩИЙ ГОСТ: https://docs.cntd.ru/document/1200030164

ГОСТ - региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств. На территории Евразийского экономического союза межгосударственные стандарты применяются добровольно.

Стандарт как нормативно-технический документ устанавливает комплекс норм, правил, требований к объекту стандартизации. Стандарт может быть разработан как на материальные предметы (продукцию, эталоны, образцы веществ), так и на нормы, правила, требования в различных областях.

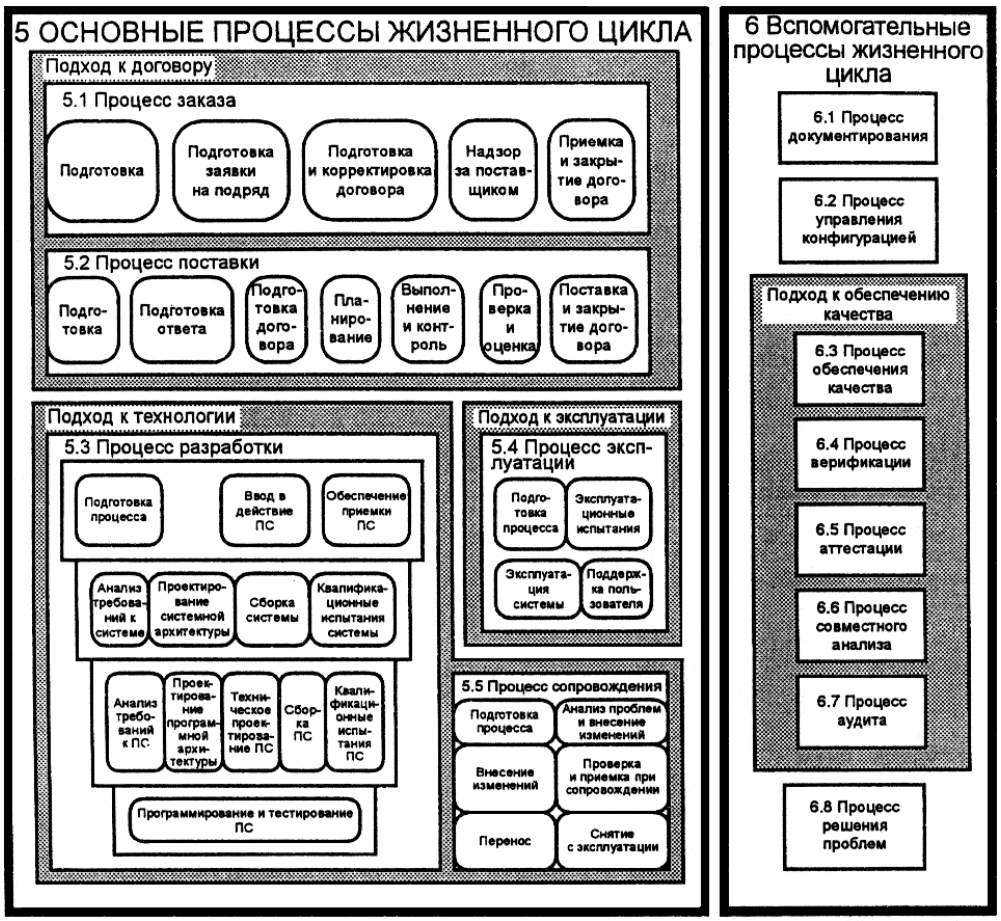
Программные средства являются неотъемлемыми частями информационных технологий и традиционных систем, таких как транспортные, военные, здравоохранения и финансовые. При этом подразумевается усиление роли стандартов, процедур, методов, средств (инструментария) и внешних условий для разработки и сопровождения программных средств (программного обеспечения). Подобная многоплановость подходов создает значительные трудности при управлении программными средствами и в технологиях программирования, особенно при интеграции продуктов и услуг. Требуется определенное упорядочение вопросов создания программных средств при переходе от подобной многоплановости к общей структуре, которая может быть использована профессионалами для "разговора на одном языке" при создании и управлении программными средствами. Настоящий стандарт устанавливает такую общую структуру.

Данная структура охватывает жизненный цикл программных средств от концепции замыслов через определение и объединение процессов для заказа и поставки программных продуктов и услуг. Кроме того, данная структура предназначена для контроля и модернизации данных процессов.

Процессы, определенные в настоящем стандарте, образуют множество общего назначения. Конкретная организация, в зависимости от своих целей, может выбрать соответствующее подмножество процессов для выполнения своих конкретных задач. Поэтому настоящий стандарт следует адаптировать для конкретной организации, проекта или приложения. Настоящий стандарт предназначен для использования как в случае отдельно поставляемых программных средств, так и для программных средств, встраиваемых или интегрируемых в общую систему.

На рисунке изображен ЖЦПИ.


Начало
Определение потребностей
Используют процесс заказа для принятия решения о выполнимости технических, эксплуатационных и экономических требований
Описание концепции
Используют процессы заказа, поставки и разработки для определения требований к проектированию системы, к прототипам разработки и к анализу обратной связи с пользователем в целях реализации предложенных решений
Демонстрация
Используют процессы заказа, поставки и разработки для определения требований к системе, системной архитектуры и проектных программных требований к элементам (компонентам) программного средства
Разработка
Используют процессы заказа и поставки для запуска процесса разработки.
Используют процесс разработки (полностью) для создания, тестирования и сборки продукта
Изготовление
Используют вспомогательные процессы для контроля и верификации тиражируемых продуктов
Распространение и продажа
Используют процессы заказа, поставки и разработки для ввода в действие и настройки продуктов
Эксплуатация
Используют процессы заказа, поставки и эксплуатации для обеспечения эксплуатационных услуг
Сопровождение и поддержка
Используют процессы заказа, поставки и сопровождения для сопровождения/поддержки
Снятие с эксплуатации
Используют процессы заказа и сопровождения для снятия с эксплуатации
Конец


5 ОСНОВНЫЕ ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА
Подход к договору
5.1 Процесс заказа
Подготовка
Подготовка заявки на подряд
Подготовка и корректировка договора
Надзор за поставщиком
Приемка и закрытие договора
5.2 Процесс поставки
Подготовка
Подготовка ответа
Подготовка договора
Планирование
Выполнение и контроль
Проверка и оценка
Поставка и закрытие договора
Подход к технологии
5.3 Процесс разработки
Подготовка процесса
Ввод в действие ПС
Обеспечение приемки ПС
Анализ требований к системе
Проектирование системной архитектуры
Сборка системы
Квалификационные испытания системы
Анализ требований к ПС
Проектирование программной архитектуры
Техническое проектирование ПС
Сборка ПС
Квалификационные испытания ПС
Программирование и тестирование ПС
Подход к эксплуатации
5.4 Процесс эксплуатации
Подготовка процесса
Эксплуатационные испытания
Эксплуатация системы
Поддержка пользователя
5.5 Процесс сопровождения
Подготовка процесса
Анализ проблем и внесение изменений
Внесение изменений
Проверка и приемка при сопровождении
Перенос
Снятие с эксплуатации
6 Вспомогательные процессы жизненного цикла
6.1 Процесс документирования
6.2 Процесс управления конфигурацией
Подход к обеспечению качества
6.3 Процесс обеспечения качества
6.4 Процесс верификации
6.5 Процесс аттестации
6.6 Процесс совместного анализа
6.7 Процесс аудита
6.8 Процесс решения проблем


7 ОРГАНИЗАЦИОННЫЕ ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА
Подход к управлению
7.1 Процесс управления
Подготовка и определение области управления
Планирование
Выполнение и контроль
Проверка и оценка
Завершение
7.2 Процесс создания инфраструктуры
7.4 Процесс обучения
7.3 Процесс усовершенствования
Создание процесса
Оценка процесса
Усовершенствование процесса

8. Доменная система имен. DNS-сервера. Регистрация доменов.

DNS «система доменных имён») — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене (SRV-запись).

Все компьютеры, подключенные к Интернету, включая смартфоны, настольные компьютеры и серверы, предоставляющие контент для огромных торговых веб-сайтов, находят друг друга и обмениваются информацией с помощью цифр. Эти цифры называются **IP-адресами**. Чтобы открыть веб-сайт в браузере, не требуется запоминать длинные наборы цифр. Достаточно ввести **доменное имя**, например example.com, и браузер откроет нужную страницу.

DNS-сервер, **Domain name server** — приложение, предназначенное для ответов на DNS-запросы по соответствующему протоколу. Также DNS-сервером могут называть хост, на котором запущено соответствующее приложение.

DNS-сервер — это специализированный компьютер (или группа), который хранит IP-адреса сайтов. Последние, в свою очередь, привязаны к именам сайтов и обрабатывает запросы пользователя. В интернете много DNS-серверов, они есть у каждого провайдера и обслуживают их пользователей.

Основное предназначение DNS-серверов — хранение информации о доменах и ее предоставление по запросу пользователей, а также кэширование DNS-записей других серверов. Это как раз «книга контактов», о которой мы писали выше.

IP-адрес сайта может измениться — например, при переезде на другой хостинг или сервер в рамках прежнего хостинга. Что происходит в этом случае? В этом случае обращения пользователей к сайту, чей IP-адрес поменялся, некоторое время обрабатываются по-старому, то есть перенаправление идет на прежний «айпишник». И лишь через определенное время (например, сутки) кэш локальных серверов обновляется, после чего обращение к сайту идет уже по новому IP-адресу.

Регистрация домена у регистраторов – это всегда платная услуга. Стоимость определяется популярностью и востребованностью доменной зоне и длинной всего домена.

Регистрируется домен на физическое лицо, либо организацию. Процесс регистрации, следующий:

Перейти на сайт регистратора.

Придумать доменное имя.

Проверить его на незанятость на сайте.

Зарегистрироваться на сайте.

Оплатить услугу доменного имени в подходящей доменной зоне

Любое доменное имя, если человек не продлевает на него подписку, может быть выставлено на продажу администратором.

После регистрации доменного имени нужно «рассказать» о нем DNS-серверам. Для этого нужно прописать ресурсные записи, что обычно делается в админке хостинг-провайдера или доменного провайдера. Примерно через сутки DNS-записи пропишутся в локальном сервере, также они попадут и в реестры всех прочих DNS-серверов. Как только это произойдет, новый домен станет нормально открываться браузером. «DNS сайта», как иногда ошибочно называют доменное имя, активируется.

9. Модель жизненного цикла программного изделия.

Жизненный цикл программного обеспечения (Software Life Cycle Model) — это период времени, который начинается с момента принятия решения о создании программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл — процесс построения и развития ПО. Жизненный цикл следует рассматривать как основу деятельности менеджера программного проекта: с ним связываются и цели проекта — окончательные и промежуточные, распределение и контроль расходования ресурсов, а также все другие аспекты управления развитием проекта. Прежде всего эта привязка обусловлена разбиением производства любой программы на этапы, которые ассоциируются с определенными видами работ или функций, выполняемых разработчиками в тот или иной момент развития проекта. Этапы характеризуются направленностью выполняемых функций на достижение локальных (для этапа) целей проекта. Необходимость отслеживания целей приводит к понятию контрольных точек — моментов разработки, когда осуществляется подведение промежуточных итогов, осмысление достигнутого и ревизия сделанных ранее предположений.

Жизненный цикл программного обеспечения можно представить в виде моделей. Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует.

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Модель кодирования и устранения ошибок
Данная модель имеет следующий алгоритм:
Постановка задачи
Выполнение
Проверка результата
При необходимости переход к первому пункту
Каскадная модель жизненного цикла программного обеспечения (водопад)
Алгоритм каскадной модели

V модель (разработка через тестирование)

Данная модель имеет более приближенный к современным методам алгоритм, однако все еще имеет ряд недостатков. Является одной из основных практик экстремального программирования.

Модель на основе разработки прототипа

Данная модель основывается на разработки прототипов и прототипирования продукта. Прототипирование используется на ранних стадиях жизненного цикла программного обеспечения:

Горизонтальные прототипы — моделирует исключительно UI не затрагивая логику обработки и базу данных.

Вертикальные прототипы — проверка архитектурных решений.

Одноразовые прототипы — для быстрой разработки.

Эволюционные прототипы — первое приближение эволюционной системы.

Спиральная модель жизненного цикла программного обеспечения

Спиральная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции.

10. Модель жизненного цикла программного изделия. Кодирование и отладка.

Кодирование (программирование)

На данной стадии строятся прототипы как целой программной системы, так и её частей, осуществляется физическая реализация структур данных, разрабатывается программные коды, выполняется отладочное тестирование, создается техническая документация. В результате этапа кодирования появляется рабочая версия продукта.

Тестирование и отладка

Тестирование ПО тесно связано с этапами проектирования и реализации. В систему встраиваются специальные механизмы, которые дают возможность производить тестирование программного обеспечения на соответствие требований к нему, проверку оформления и наличие необходимого пакета документации.

Результатом тестирования является устранение всех недостатков программного продукта и заключение о ее качестве.

11. Модель жизненного цикла программного изделия. Планирование и составление требований к программному изделию.

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Модель ЖЦ ПО включает в себя:

- Стадии;
- Результаты выполнения работ на каждой стадии;
- Ключевые события — точки завершения работ и принятия решений.

Разработка требований – один из основных процессов на проекте по разработке программного обеспечения.

Требования являются «валютой» проекта – реальное понимание целей и проблем заказчика, четкое формулирование задач и решений является основой успешного взаимодействия с заказчиком. Полнота и согласованность требований является условием построения простой и гибкой архитектуры, продуктивного и экономичного процесса разработки программного решения. Требования к программному обеспечению – это описание того, что, как и для чего должна сделать программная система.

// Заказчик начинает процесс заказа, описывая концепцию или потребность в заказе, разработке или модернизации системы, программного продукта или программной услуги. Заказчик должен определить и проанализировать требования к системе. Требования к системе должны охватывать функциональные, коммерческие, организационные и потребительские аспекты системы, а также требования безопасности, защиты и другие критические требования наряду с требованиями к проектированию, тестированию и соответствующим стандартам и процедурам. Если заказчик поручает поставщику выполнение анализа требований к системе, то заказчик должен согласовать требования, сформулированные в результате анализа. //

Схема разработки требований

1) Разработка требований - это первая из основных фаз процесса создания программных систем. Этот фаза состоит из следующих основных работ

2) Анализ предметной области. Позволяет выделить сущности предметной области, определить первоначальные требования к функциональности и определить границы проекта.

3) Анализ осуществимости. Должен выполняться для новых программных систем. На основании анализа предметной области, общего описания системы и ее назначения принимается решение о продолжении или завершении проекта.

4) Формирование и анализ требований. Взаимодействуя с пользователями, обсуждая и анализируя с ними задачи, возлагаемые на систему, разрабатывая модели и прототипы, разработчики формулируют пользовательские требования.

5) Документирование требований. Сформированные на предыдущем этапе пользовательские требования должны быть документированы. При этом нужно учесть, что основными читателями этого документа будут пользователи, поэтому основными требованиями к нему будут ясность и понятность.

6) Детализация требований. Разработчики детализируют требования пользователей, формируя более точные подробные системные требования.

7) Согласование и утверждение требований. На этом этапе пользовательские и системные требования должны быть оформлены в виде единого документа, содержащего все функциональные и нефункциональные требования. Такой документ, обычно, называется спецификацией требований. Спецификация требований должна удовлетворять следующим характеристикам качества: корректность, однозначность, завершенность и согласованность.

12. Модель жизненного цикла программного изделия. Проектирование программного изделия (высокоуровневое проектирование и низкоуровневое проектирование).

Проектирование программного обеспечения — процесс создания проекта программного обеспечения (ПО), а также дисциплина, изучающая методы проектирования. Проектирование ПО является частным случаем проектирования продуктов и процессов.

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к ПО (исходные условия задачи). Эти требования подвергаются анализу.

Проектирование ПО включает следующие основные виды деятельности:

выбор метода и стратегии решения;

выбор представления внутренних данных;

разработка основного алгоритма;

документирование ПО;

тестирование и подбор тестов;

выбор представления входных данных.

В российской практике проектирование ведется поэтапно в соответствии со стадиями, регламентированными ГОСТ 2.103-68[2] :

Техническое задание(по ГОСТ 2.103-68 к стадиям разработки не относится),

Техническое предложение,

Эскизный проект,

Технический проект,

Рабочий проект.

Высокоуровневое проектирование — определяются структура программного продукта, взаимосвязи между основными его компонентами и реализуемые ими функции;

Детальное проектирование — определяется алгоритм работы каждого компонента. заключается в детальной проработке поставленных задач и проверке качества разработанных решений.

13. Модель жизненного цикла программного изделия. Системное тестирование.

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Модель ЖЦ ПО включает в себя:

- Стадии;
- Результаты выполнения работ на каждой стадии;
- Ключевые события — точки завершения работ и принятия решений.

Тестирование ПО представляет из себя процесс проверки того, насколько поведение реального ПО соответствует требованиям и ожиданиям. Проводимые для этого тесты основаны на задокументированных требованиях заказчика.

Системное тестирование — процесс тестирования системы, на котором проводится не только функциональное тестирование, но и оценка характеристик качества системы — ее устойчивости, надежности, безопасности и производительности.

Системное тестирование происходит на стадии разработки. Эта стадия жизненного цикла разработки ПО подразумевает общий тест системы на предмет интеграции ее компонентов. Это значит, что в случае, если система состоит из различных модулей, мы должны проверить, насколько хорошо или насколько плохо каждый из них работает внутри системы. Более того, на этом этапе важно произвести тестирование пользовательского интерфейса. Системное тестирование проводится в несколько фаз, на каждой из которых проверяется один из аспектов поведения системы, т.е. проводится один из типов системного тестирования.

Принято выделять следующие виды системного тестирования:

- функциональное тестирование (В ходе данного вида тестирования проверяются все функции системы с точки зрения ее пользователей (как пользователей-людей, так и "пользователей" - других программных систем));

- тестирование производительности (Тестирование производительности выполняется при различных уровнях нагрузки на систему, на различных конфигурациях оборудования. Выделяют три основных фактора, влияющие на производительность системы: количество поддерживаемых системой потоков (например, пользовательских сессий), количество свободных системных ресурсов, количество свободных аппаратных ресурсов);

- нагрузочное или стрессовое тестирование (Основная цель стрессового тестирования - вывести систему из строя, определить те условия, при которых она не сможет далее нормально функционировать. Для проведения стрессового тестирования используются те же самые инструменты, что и для тестирования производительности. Однако, например, генератор нагрузки при стрессовом тестировании должен генерировать запросы пользователей с максимально возможной скоростью либо генерировать данные запросов таким образом, чтобы они были максимально возможными по объему обработки.);

- тестирование конфигурации (В ходе тестирования конфигурации проверяется, что программная система корректно работает на всем поддерживаемом аппаратном обеспечении и совместно с другими программными системами. Необходимо также проверять, что система продолжает стабильно работать при горячей замене любого поддерживаемого устройства на аналогичное. При этом система не должна давать сбоев ни в момент замены устройства, ни после начала работы с новым устройством.);

- тестирование безопасности (Если программная система предназначена для хранения или обработки данных, содержимое которых представляет собой тайну определенного рода (личную, коммерческую, государственную и т.п.), то к свойствам системы, обеспечивающим сохранение этой тайны, будут предъявляться повышенные требования. Эти требования должны быть проверены при тестировании безопасности системы. В ходе этого тестирования проверяется, что информация не теряется, не повреждается, ее невозможно подменить, а также к ней невозможно получить несанкционированный доступ, в том числе при помощи использования уязвимостей в самой программной системе);

- тестирование надежности и восстановления после сбоев (При тестировании восстановления после сбоев имитируются сбои оборудования или окружающего программного обеспечения либо сбои программной системы, вызванные внешними факторами. При анализе поведения системы в этом случае необходимо обращать внимание на два фактора - минимизацию потерь данных в результате сбоя и минимизацию времени между сбоем и продолжением нормального функционирования системы);

- тестирование удобства использования (Требования к пользовательскому интерфейсу могут быть разбиты на две группы:

1) требования к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем;

2) требования по доступу к внутренней функциональности системы при помощи пользовательского интерфейса.).

После завершения системного тестирования разработка переходит в фазу приемо-сдаточных испытаний (для программных систем, разрабатываемых на заказ) или в фазу альфа- и бета-тестирования (для программных систем общего применения).

14. Модель жизненного цикла программного изделия. Сопровождение программного изделия.

Сопровождение (поддержка) программного обеспечения — процесс улучшения, оптимизации и устранения дефектов программного обеспечения (ПО) после передачи в эксплуатацию. Сопровождение ПО — это одна из фаз жизненного цикла программного обеспечения, следующая за фазой передачи ПО в эксплуатацию. В ходе сопровождения в программу вносятся изменения с тем, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности, с целью повысить удобство использования (юзабилити) и применимость ПО.

Сопровождение программного обеспечения связано с внесением изменений в течение всего времени использования программного изделия. К причинам, определяющим необходимость внесения изменений в изделии, относятся:

- наличие ошибок в используемом программном продукте;
- изменение требований пользователя (расширение или модификация);
- появление более совершенных общесистемных программных средств или технических устройств;
- изменение организационной структуры, условий и методов работы пользователя.

Первая причина связана с качеством программного изделия; остальные обусловлены, как правило, длительным процессом эксплуатации. Конечной целью любых изменений является совершенствование программного изделия: повышение его корректности, надежности и функциональной полезности. Однако внесение изменений в программное изделие может породить новые ошибки, поэтому требуется жесткая регламентация всех процессов внесения изменений.

В зависимости от сложности программного изделия и числа пользователей, сопровождение может осуществляться в тесной связке с группой разработки изделия, т.е. сопровождение поручается программистам-разработчикам. В последнее время используется другая схема. После гарантийного периода сопровождение может быть передано от разработчика к организации (или специальному подразделению), которая специально занимается сопровождением, т.е. для каждого программного изделия, находящегося в практичес-ком использовании, имеется организация, ответственная за его сопровождение.

Принято выделять несколько линий сопровождения (структура приведена на примере внешнего сопровождения ПО):

- 0 линия (call-center, информационный центр, горячая линия) — обработка телефонных обращений от клиентов, передача обращений техническим специалистам (1-я линия сопровождения)

· 1 линия (инженер по сопровождению, инженер технической поддержки, support engineer) — консультация/настройка/устранение ошибок в работе ПО/наполнение базы знаний, составление мануалов

· 2 линия (инженер по сопровождению, инженер технической поддержки, support engineer) — функциональное сопровождение/проектная деятельность на этапе запуска ПО на машинах заказчика

· 3 линия (инженер по сопровождению, инженер технической поддержки, support engineer) — системное сопровождение/проектная деятельность на этапе запуска ПО на оборудовании заказчика

В процессе эксплуатации программного изделия пользователи взаимодействуют с организацией (группой), ответственной за сопровождение. Задачами службы сопровождения являются:

1. Сбор и анализ поступающих от пользователей сведений об обнаруженных ошибках, замечаний и предложений по совершенствованию и изменению программного изделия.

2. Исправление ошибок в программах, выдающих результаты, не отвечающие установленным требованиям, и внесение соответст-вующих изменений в документацию.

3. Модернизация программного изделия путем расширения функциональных возможностей или улучшения эксплуатационных характеристик программного изделия.

4. Внесение изменений в программы с целью их приспособления к условиям работы конкретного пользователя.

5. Контроль правильности всех корректировок, вносимых в изделие, и проверка качества измененных программ.

6. Доведение до пользователя информации о внесенных изменениях.

Обучение и постоянные консультации пользователя с целью повышения эффективности использования программного изделия.

15. Мультимедийные компоненты в составе веб-ресурсов: встраивание звука и видео.

Мультимедиа — это собирательное понятие для различных компьютерных технологий, при которых используется несколько информационных сред, таких, как графика, текст, видео, фотография, анимация, звуковые эффекты, высококачественное звуковое сопровождение.

Для встраивания видео на веб-ресурс, написанном на html, используют элемент <video>.

Простой пример:

<video src="rabbit320.webm" controls></video>

Описание параметров:

src – атрибут src содержит путь к видео, которое вы хотите внедрить.

controls – атрибут для отображения элементов управления, чтобы позволить пользователю управлять воспроизведением видео, регулировать громкость, осуществлять перемотку, а также ставить на паузу и возобновление воспроизведение.

Присутствует одна проблема с приведённым выше примером, если попытался получить доступ к прямой ссылке выше с помощью браузера, такого как Safari или Internet Explorer, видео не будет воспроизводится. Это происходит из-за разности кодеков в разных браузерах. Чтобы максимизировать вероятность того, что веб-сайт или приложение будет работать в браузере

пользователя, может потребоваться предоставить каждый медиафайл, который используется, в нескольких форматах. Пример:

```
<video controls>
  <source src="rabbit320.mp4" type="video/mp4">
  <source src="rabbit320.webm" type="video/webm">
</video>
```

Здесь изымается атрибут src из тега <video>, и вместо этого включаются отдельные элементы <source>, каждый из которых ссылается на собственный источник. В этом случае браузер пройдётся по элементам <source> и начнёт воспроизводить первый из них, который имеет поддерживаемый кодек. Включение источников WebM и MP4 должно быть достаточно для воспроизведения видео на большинстве платформ и браузеров в наши дни.

Разумеется можно использовать и другие параметры для настройки видео (width и height, autoplay, muted, loop(зацикливание), poster(атрибут принимает в качестве значения URL-адрес изображения, который будет отображаться до воспроизведения видео), preload(буферизации больших файлов))

Для встраивания аудио-файлов используется элемент <audio>. Элемент <audio> работает точно так же, как элемент <video>, с несколькими небольшими отличиями.

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3">
  <source src="viper.ogg" type="audio/ogg">
</audio>
```

Он занимает меньше места, чем видеоплеер, поскольку нет визуального компонента - вам просто нужно отображать элементы управления для воспроизведения звука. В этом и заключается отличие, в нем нет некоторых атрибутов, таких как width и height, poster.

Для управления <video> и <audio> используются различные API, такие как HTMLMediaElement, Web Audio API, и WebRTC. Они позволяют создать собственный пользовательский интерфейс (User Interface, UI) для проигрывания аудио/видео, вывод на экран субтитров, записывать видео с веб-камеры для обработки или для передачи на другой компьютер в видеоконференции, применять звуковые эффекты к аудио-файлам (такие как gain, distortion, panning и т.д.).

16. Основные технологии создания анимации на веб-платформе, их сходство и различие.

Анимация в основном возникает, когда неподвижный элемент «оживает». Затем он начинает демонстрировать некоторую форму движения.

По сути технология анимации — это оптическая иллюзия. В классической анимации, она же покадровая или мультипликация, автор сначала от руки рисует серию картинок с минимальным отличием друг от друга.

После чего эту последовательность картинок демонстрируют зрителю с достаточно большой скоростью, чтобы в возникла иллюзия движения. Ввиду особенностей человеческого глаза и мозга, ощущение, что картинка оживает, появляется у зрителя уже при скорости воспроизведения 16 кадров в секунду. Общепризнанным же стандартом для кино и анимации стала скорость 24 кадра в секунду.

В основном для создания анимированных веб-страниц разработчики пользуются CSS-анимациями и переходами или JavaScript.

CSS-анимации делают возможными переходы между различными состояниями и используют при этом наборы ключевых кадров.

С CSS-анимациями вам не нужны внешние библиотеки. У них отличная производительность. Также их можно с легкостью использовать в адаптивной разработке, поскольку вы можете их модифицировать с помощью медиа-запросов. Вместо того чтобы создавать объекты для ключевых кадров и временные характеристики, вы можете просто передавать их значения напрямую.

Но с CSS-анимациями вы не можете создавать сложные физические эффекты и имитировать реалистичное движение.Они также не сработают, если вам нужно сделать больше трех анимаций подряд. Сложные и последовательные анимации лучше писать на JavaScript.

JavaScript дает вам больше опций и больше контроля. В базовом Javascript есть собственный функционал анимаций, но чаще всего их создают с помощью дополнительных библиотек. Производительность при этом зависит от выбранной библиотеки. Но следует учитывать, что сложные анимации, написанные на Javascript, могут увеличить время загрузки страницы.

Javascript-анимации предлагают больше возможностей и гибкости, чем переходы и анимации, написанные на CSS. Именно с помощью Javascript зачастую создаются продвинутые анимации, такие как подпрыгивание, пауза, остановка и замедление.

SVG означает Scalable Vector Graphics (масштабируемая векторная графика). Это формат векторной графики, который может использоваться в интернете. SVG-анимации выглядят очень четкими благодаря тому, что векторы не имеют никаких пиксельных ограничений. Не важно, как вы измените размеры страницы, – SVG будет сохранять свой вид и не потеряет качество, в отличие от растровых изображений.

SVG-элементы можно анимировать с помощью CSS. Но SVG также имеет и собственный синтаксис для анимации, который называется SMIL. В отношении SVG-анимаций SMIL подходит лучше, поскольку позволяет анимировать некоторые свойства, недоступные CSS.

Canvas предоставляет визуальное пространство, где вы можете создавать сложные анимации с высокопроизводительным рендерингом. При этом canvas-анимации работают с пикселями: это не векторные анимации вроде SVG.

Canvas это отличный способ делать комплексные и красивые анимации. Взаимодействуя с пикселями, вы можете создавать сложные вещи и при этом не утяжелять производительность. Так что это отличный выбор для сложных рисунков и взаимодействий.

WebGL означает Web Graphics Library (библиотека веб-графики). Эта библиотека, в основном, используется для сложных эффектов и 3D. Также ее можно использовать при создании анимаций для виртуальной реальности. WebGL позволяет рендеринг графики при 60 кадрах в секунду. Для создания анимаций, аналогичных WebGL, вы также можете использовать Canvas, но это будет сложнее. Большинство красивых и креативных визуальных эффектов созданы с помощью WebGL.

Форматы анимации:

GIF-формат

Тип: растровая графика

Это очень старый формат. Его показывают все браузеры. У него есть прозрачный слой, т.е. под картинкой будет виден фон, на котором она лежит. Для анимации сохраняет серию полноценных картинок.

GIF показывает только 256 цветов. Это можно исправить при помощи анимации. Мы создаем две картинки с разным набором цветов. Демонстрируем их с задержкой 0. В итоге получаем 256 + 256 = 512 цветов.

Формат хорошо сжимает изображение без потерь. Для маленьких картинок это важно.

APNG-формат

Тип: растровая графика

Этот формат (Animated PNG) является расширением формата PNG. Однако разработчики последнего не включили это расширение в спецификацию. Получилось, что не все программы могут его правильно отображать. Некоторые браузеры покажут первый кадр статичной картинкой, а про анимацию забудут.

Все кадры, кроме первого, хранятся в дополнительных блоках APNG. Дополнительный блок хранит информацию о количестве кадров и повторений анимации.

Чтобы уменьшить размер, APNG использует промежуточный. Каждый кадр имеет свой режим работы с кадровым буфером:

1. None — сохранять кадр в кадровый буфер.
2. Background — очищать кадровый буфер.
3. Previous — не сохранять кадр в кадровый буфер.

Формат прост и содержит только данные, то есть нет никаких инструкций. Также прост в реализации – объем спецификации 30 кб.

WebP-формат

Тип: растровая графика

WebP images — это формат изображений от Google, предназначенный для размещения картинок в интернете в качестве. Размер файла получается меньше, чем у других форматов, но качество при этом не ухудшается.

Преимущества WebP по сравнению с GIF

- WebP поддерживает 24-битный цвет RGB по сравнению с 8-битным цветом GIF.
- WebP поддерживает сжатие как с потерями, так и без потерь; Фактически, одна анимация может комбинировать кадры с потерями и без потерь. GIF поддерживает только сжатие без потерь. Методы сжатия WebP с потерями хорошо подходят для анимированных изображений, созданных из реальных видео.
- WebP требует меньше байтов, чем GIF. Анимированные GIF-файлы, преобразованные в WebP с потерями, на 64% меньше, а файлы WebP без потерь на 19% меньше. Это особенно важно в мобильных сетях.

Недостатки анимированного WebP по сравнению с анимированным GIF

- Прямолинейное декодирование WebP потребляет больше ресурсов ЦП, чем GIF.
- Поддержка WebP не так широко распространена, как поддержка GIF.
- Добавление поддержки WebP в браузеры увеличивает объем кода и поверхность для атак.

В будущем эту проблему можно будеть решить, если WebP и WebM будут совместно использовать более общий код декодирования или если возможности WebP будут включены в WebM.

Мини вывод по первым трем форматам:

gif плохое сжатие, 256 цветов

Apng сжатие без потерь, много цветов

webp хорошее сжатие, много цветов, есть режим без потерь и анимация.

Разница составляет в том, что gif работает в любом браузере, 2 другие не везде.

17. Методологии управления проектами: классификация, особенности, достоинства и недостатки. Описание принципов

семейства гибких (Agile) методологий. Описание методологии SCRUM.

Методологии управления проектами принято разделять на каскадную (waterfall) и гибкие (agile) методологии. Каскадная методология обладает очень простым подходом, который ценит надежное планирование, однократное и правильное выполнение. Состоит из шести этапов: планирование, проектирование, разработка, тестирование и отладка, внедрение и эксплуатация. **Особенностями** данной методологии являются:

+ Четкая, последовательная структура. Только после завершения одного из шести этапов команда переходит к следующему.

+ Большое внимание к документации и этапам планирования/ проектирования в целом

- Низкая вовлеченность клиентов. Требования должны быть согласованы на раннем этапе жизненного цикла проекта.

- Конечный продукт доходит до владельца уже полностью готовым, внести правки на таком этапе может быть весьма затратно по времени и бюджету.

В целом, подход waterfall заключается в тщательном планировании и исполнении. Waterfall может быть полезным подходом, если требования фиксированы и хорошо задокументированы.

Гибкая методология разработки (Agile software development) – это не конкретный метод, а обобщающий термин для целого ряда методов и практик. Эти методы основаны на ценностях Манифеста гибкой разработки программного обеспечения (Agile Manifesto), разработанного в феврале 2001 года, содержащего описание ценностей и принципов, которыми руководствуется команда.

Основные идеи гибкой методологии:

Люди и взаимодействие важнее процессов и инструментов.

Работающий продукт важнее исчерпывающей документации.

Сотрудничество с заказчиком важнее согласования условий контракта.

Готовность к изменениям важнее следования первоначальному плану.

То есть, не отрицая важности первых, более ценны вторые.

К гибким методологиям относят: экстремальное программирование, Crystal Clear, Dynamic Systems Development Method (DSDM), Agile Unified Process (AUP), Feature-driven development, ICONIX, Канбан, SCRUM.

Методика SCRUM – это каркас, который можно использовать для организации команды и достижения результата более продуктивно и с более высоким качеством за счет анализа сделанной работы и корректировки направления развития между итерациями. Методология позволяет команде выбрать задачи, которые должны быть выполнены, учитывая бизнес-приоритеты и технические возможности, а также решить, как их эффективно реализовать. Это позволяет создать условия, при которых команда работает с удовольствием и максимально продуктивно. Рабочий процесс поделен на равные этапы (от двух до четырех недель) – спринты, что позволяет лучше контролировать ход проекта, быстрее получать обратную связь от заказчика и моментально решать проблемы, если они возникнут.

В SCRUM принято выделять три основных роли:

Владелец продукта (Product owner) – это человек, ответственный за приоритезацию требований и часто за их создание.

Скрам-мастер (Scrum Master) – член команды, который дополнительно отвечает за процессы, координацию работы команды и поддержание социальной атмосферы в команде.

Команда – 5-9 человек, которые реализуют требования владельца продукта.

Процессы:

Скрам-митинг (Scrum meeting, скрам, ежедневный скрам, планерка) – собрание членов команды (с возможностью приглашения владельца продукта) для синхронизации деятельности команды и обозначения проблем.

18. Развертывание веб-ресурса в контейнере. Технология Docker и методология DevOps.

Контейнер — это стандартная единица программного обеспечения, в которую упаковано приложение со всеми необходимыми для его работы зависимостями — кодом приложения, средой запуска, системными инструментами, библиотеками и настройками.

Контейнеризация – это метод визуализации. Все процессы в ней протекают на уровне операционной системы, что позволяет существенно экономить ресурсы и увеличивать эффективность работы с приложениями.

Одним из наиболее популярных инструментов для программной виртуализации является Docker — автоматизированное средство управления виртуальными контейнерами. Он решает множество задач, связанных с созданием контейнеров, размещением в них приложений, управлением процессами, а также тестированием ПО и его отдельных компонентов.

Основной принцип работы Docker — контейнеризация приложений. Этот тип виртуализации позволяет упаковывать программное обеспечение по изолированным средам — контейнерам. Каждый из этих виртуальных блоков содержит все нужные элементы для работы приложения. Это дает возможность одновременного запуска большого количества контейнеров на одном хосте.

Каждый контейнер строится на основе Docker-образов. Контейнеры запускаются напрямую из ядра операционной системы Linux. Благодаря этому, они потребляют гораздо меньше ресурсов, чем при аппаратной виртуализации.

Что происходит при запуске контейнера:

Происходит запуск образа (Docker-image). Docker Engine проверяет существование образа. Если образ уже существует локально, Docker использует его для нового контейнера. При его отсутствии выполняется скачивание с Docker Hub.

Создание контейнера из образа.

Разметка файловой системы и добавление слоя для записи.

Создание сетевого интерфейса.

Поиск и присвоение IP-адреса.

Запуск указанного процесса.

Захват ввода/вывода приложения.

DevOps — процесс непрерывного совершенствования программных продуктов посредством ускорения циклов релизов, глобальной автоматизации интеграционных и развёрточных конвейеров-пайплайнов и тесного взаимодействия между командами. Целью DevOps является сокращение времени и стоимости воплощения идеи в продукте, которым пользуются клиенты.

DevOps – это и новая профессиональная область. Инженер DevOps должен обладать рядом системных компетенций, состав которых формируется прямо сейчас.

Организации внедряют DevOps для того, чтобы уменьшить стоимость времени при задержке циклов разработки и удовлетворить потребности клиентов.

С использованием DevOps разработчики могут выпускать новые версии своих программ, тестировать их и развёртывать для клиентов за несколько часов. Это не значит, что версии всегда

выпускаются так быстро, — для должной проверки качества (QA) тоже требуется время, но DevOps при необходимости предоставляет возможность быстрее двигаться дальше.

DevOps и его предшественники — Agile-манифест — имеют одну общую черту — стремление к ускоренной поставке заказчику улучшенных продуктов. Каждая успешная стратегия начинается с сосредоточенности на заказчике.

В рамках DevOps все, кто задействован в конвейере продукта, преследуют цели заказчика.

Менеджеры продукта оценивают показатели взаимодействия и задержек.

Разработчики оценивают эргономику и практичность.

Специалисты по эксплуатации оценивают продолжительность работоспособного состояния и время ответа.

DevOps учит нас: для успешной стратегии необходимы сближение стороны эксплуатации со стороной разработки, а также разрушение коммуникационного барьера между различными разработчиками и специалистами по эксплуатации.

Составляющей DevOps становится безопасность – непрерывная безопасность.

19. Разработка технического задания. ГОСТ на разработку технического задания.

ДЕЙСТВУЮЩИЙ ГОСТ: https://docs.cntd.ru/document/1200144624

ГОСТ 34.602-89. Межгосударственный стандарт. Информационная.rtf

Техни́ческое зада́ние — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями.

ТЗ на АС содержит следующие разделы, которые могут быть разделены на подразделы:

1) общие сведения;

2) назначение и цели создания (развития) системы;

3) характеристика объектов автоматизации;

4) требования к системе;

5) состав и содержание работ по созданию системы;

6) порядок контроля и приемки системы;

7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;

9) источники разработки.

В зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ.

В разделе "Общие сведения" указывают:

1) полное наименование системы и ее условное обозначение;

2) шифр темы или шифр (номер) договора;

3) наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты;

4) перечень документов, на основании которых создается система, кем и когда утверждены эти документы;

5) плановые сроки начала и окончания работы по созданию системы;

6) сведения об источниках и порядке финансирования работ;

7) порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы.

В подразделе "Назначение системы" указывают вид автоматизируемой деятельности (управление, проектирование и т.п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать.

Для АСУ дополнительно указывают перечень автоматизируемых органов (пунктов) управления и управляемых объектов.

В подразделе "Цели создания системы" приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания АС, и указывают критерии оценки достижения целей создания системы.

В разделе "Характеристики объекта автоматизации" приводят:

1) краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию;

2) сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

Раздел "Требования к системе" состоит из следующих подразделов:

1) требования к системе в целом;

2) требования к функциям (задачам), выполняемым системой;

3) требования к видам обеспечения.

В подразделе "Требования к системе в целом" указывают:

- требования к структуре и функционированию системы;
- требования к численности и квалификации персонала системы и режиму его работы;
- показатели назначения;
- требования к надежности;
- требования безопасности;
- требования к эргономике и технической эстетике;
- требования к транспортабельности для подвижных АС;
- требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
- требования к защите информации от несанкционированного доступа;
- требования по сохранности информации при авариях;
- требования к защите от влияния внешних воздействий;
- требования к патентной чистоте;
- требования по стандартизации и унификации;
- дополнительные требования.

В требования к надежности включают:

1) состав и количественные значения показателей надежности для системы в целом или ее подсистем;

2) перечень аварийных ситуаций, по которым должны быть регламентированы требования к надежности, и значения соответствующих показателей;

3) требования к надежности технических средств и программного обеспечения;

4) требования к методам оценки и контроля показателей надежности на разных стадиях создания системы в соответствии с действующими нормативно-техническими документами.

20. Растровая и векторная графика в составе веб-ресурсов. SVG и внедрение шрифтов.

Для веб-дизайнера важно знать, что графика в интернете бывает двух типов: растровая и векторная.

Растровая графика

Растровое изображение, как мозаика, складывается из множества маленьких ячеек — пикселей, где каждый пиксель содержит информацию о цвете. Определить растровое изображение можно увеличив его масштаб: на определённом этапе станет заметно множество маленьких квадратов — это и есть пиксели.

К растровой графике, которой гораздо больше, чем векторной, относятся все картинки форматов JPEG, GIF, PNG, ICO, BMP.

Преимуществом растровой графики является то, что с ее помощью можно создать любой по сложности рисунок. При этом применять огромное количество фильтров и плагинов. Основным недостатком – растровую графику нельзя растягивать без потери качества. Большие рисунки нужно изначально рисовать большими, а фотографии делать с большим разрешением.

Растровая графика удобна для создания качественных фотореалистичных изображений, цифровых рисунков и фотографий. Самый популярный редактор растровой графики — Adobe Photoshop.

Растровый формат GIF, позволяет создавать анимацию. Правда, она проще и дольше загружается, чем flash, зато сделать gif-анимацию, как правило, намного легче.

Преимущества:

Возможность создать изображение любой сложности — с огромным количеством деталей и широкой цветовой гаммой.

Растровые изображения наиболее распространённые.

Работать с растровой графикой проще, так как механизмы её создания и редактирования более привычны и распространены.

Недостатки:

Большой занимаемый объём памяти: чем больше «размер» изображения, тем больше в нём пикселей и, соответственно, тем больше места нужно для хранения/передачи такого изображения.

Невозможность масштабирования: растровое изображение невозможно масштабировать без потерь. При изменении размера оригинального изображения неизбежно (в результате процесса интерполяции) произойдёт потеря качества.

Векторная графика

В отличие от растровых, векторные изображения состоят уже не из пикселей, а из множества опорных точек и соединяющих их кривых. Векторное изображение описывается математическими формулами и, соответственно, не требует наличия информации о каждом пикселе. Сколько ни увеличивай масштаб векторного изображения, вы никогда не увидите пикселей.

Самые популярные векторные форматы: SVG, AI.

Векторная графика используется для иллюстраций, иконок, логотипов и технических чертежей, но сложна для воспроизведения фотореалистичных изображений. Самый популярный редактор векторной графики — Adobe Illustrator.

Преимущества:

Малый объём занимаемой памяти — векторные изображения имеют меньший размер, так как содержат в себе малое количество информации.

Векторные изображения отлично масштабируются — можно бесконечно изменять размер изображения без потерь качества.

Недостатки:

Чтобы отобразить векторное изображение требуется произвести ряд вычислений, соответственно, сложные изображения могут требовать повышенных вычислительных мощностей.

Не каждая графическая сцена может быть представлена в векторном виде: для сложного изображения с широкой цветовой гаммой может потребоваться огромное количество точек и кривых, что сведёт «на нет» все преимущества векторной графики.

Процесс создания и редактирования векторной графики отличается от привычной многим модели — для работы с вектором потребуются дополнительные знания.

Встраивание изображений

Графика может быть встроена напрямую с помощью тега <img>. Тег <img> предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG, PNG, SVG, BMP и т.д.. Адрес файла с картинкой задаётся через атрибут src. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег <img> в контейнер <a>. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут border="0" в тег <img>.

Синтаксис

```HTML
<img src="URL" alt="альтернативный текст">
```

Изображение может быть закодировано с помощью base64. Base64 - это специальный метод кодирования информации в 64-разрядный код (6 бит). Этот метод широко используется в приложениях электронной почты для кодирования бинарных данных. Закодированные данные записываются в виде спецсимволов, букв английского алфавита и цифры.

Плюсы:

- уменьшение числа запросов к вебсерверу
- простота обновления, по сравнению со спрайтами
- картинки будут показаны даже если в браузере пользователя включена настройка «Не показывать картинки» (что может быть полезно для навигационных кнопок, в виде картинок)

Минусы:

- изображения в base64 не кешируются
- увеличение результативного объема изображения (примерно на 22%)

Также изображение может быть встроено средствами для генерации графики серверной частью.

PHP позволяет работать с графическими изображениями без использования сторонних библиотек. Графические изображения, построенные с помощью PHP, можно использовать для визуализации данных, рисования графиков, отчётов, диаграмм.

Существует множество JavaScript библиотек для работы с изображениями, каждая из которых имеет свои плюсы и минусы. Самой популярной на сегодняшний день является библиотека D3.

Язык Python также имеет множество средств для работы с графикой. Самые популярные - scikit-image и NumPy.

SVG это язык на базе XML для описания векторных изображений. По сути это язык разметки, как и HTML, только содержащий множество различных элементов для определения фигур вашего

изображения, а также параметров их отображения. SVG предназначен для разметки графики, а не содержимого.

простые SVG можно создавать, используя текстовый редактор, но в случае сложного изображения это становится сложным. Для создания SVG изображений используются редакторы векторной графики, такие как Inkscape или Illustrator. Данные приложения позволяют создавать различные изображения, используя множество графических инструментов, и создавать приближения фотографий.

Дополнительные преимущества SVG:

Текст в векторном изображении остаётся машинописным (то есть доступным для поисковика).

SVG легко поддаются стилизации/программированию (scripting), потому что каждый компонент изображения может быть стилизован с помощью CSS или запрограммирован с помощью JavaScript.

Но также SVG имеет ряд недостатков:

· SVG может очень быстро стать сложным в том смысле, что размер файла увеличивается; сложные SVG-изображения также создают большую вычислительную нагрузку на браузер.

· SVG может быть сложнее создать, нежели растровое изображение, в зависимости от того, какое изображение необходимо создать.

· не поддерживается старыми версиями браузеров, то есть не подойдёт для сайтов, поддерживающих Internet Explorer 8 или старее.

Встраивание

Быстрый путь: <img>

Чтобы встроить SVG используя элемент <img>, вам просто нужно сослаться на него в атрибуте src, как и следовало ожидать. Вам понадобится атрибут height или width (или оба, если ваш SVG не имеет собственного соотношения сторон).

Пример кода:

```
<img
    src="equilateral.svg"
    alt="triangle with all three sides equal"
    height="87px"
    width="100px" />
```

SVG inline.

Вы можете открыть файл SVG в текстовом редакторе, скопировать этот код и вставить его в ваш HTML документ — такой приём иногда называют встраиванием SVG (SVG inline или inlining SVG). Убедитесь, что фрагмент вашего SVG кода начинается и заканчивается с тегов <svg></svg> (не включайте ничего, кроме них).

Шрифты

Текст — основная часть контента на большинстве сайтов, и важно грамотно подходить к его отрисовке.

Шрифты применённые к вашему HTML могут контролироваться при помощи свойства font-family. Оно принимает одно и более имён семейств шрифтов и браузер следует по списку пока не найдёт тот шрифт, который является доступным в системе, под управлением которой он работает:

Эта система работает хорошо, но традиционно выбор шрифтов веб-разработчиков была ограниченной. Существует только горсть шрифтов которые вы можете гарантировать, что они являются доступными во всех распространённых системах — так называемые Безопасные веб-шрифты. Вы можете использовать стек шрифта для указания предпочтительных шрифтов, за которыми следует веб-безопасные альтернативы, за которыми следует системный шрифт по умолчанию, но это добавляет дополнительной работы с точки зрения тестирования, чтобы убедиться, что ваш дизайн выглядит хорошо с каждым из шрифтов и т. д.

Но есть альтернатива, которая работает очень хорошо начиная с 6-ой версии IE. Веб-шрифты — это функция CSS позволяющая вам указывать файлы шрифтов, загружаемые вместе с вашим веб-сайтом по мере доступа к нему, это означает, что любой браузер, поддерживающий веб-шрифты, может иметь в своём распоряжении именно те шрифты, которые вы укажете. Замечательно!

Требуемый синтаксис выглядит примерно так:

Во-первых, у вас есть блок @font-face в начале CSS, который указывает файл(-ы) шрифтов для загрузки:

```
p {
  font-family: Helvetica, "Trebuchet MS", Verdana, sans-serif;
}
```

Ниже вы можете использовать имя семейства шрифтов, указанное внутри @font-face, чтобы применить свой собственный шрифт ко всему, что вам нравится, как обычно:

```
html {
  font-family: "myFont", "Bitstream Vera Serif", serif;
}
```

Есть две важные вещи, которые нужно иметь в виду о веб-шрифтах:

1. Браузеры поддерживают разные форматы шрифтов, поэтому вам будут нужны несколько форматов шрифтов для приличной кросс-браузерной поддержки. Например, большинство современных браузеров поддерживают WOFF/WOFF2 (Web Open Font Format versions 1 and 2), наиболее эффективный формат, но старые версии IE поддерживают только шрифты EOT (Embedded Open Type) и вам возможно понадобиться включать версию SVG шрифта для поддержки старых версий браузеров iPhone и Android. Ниже мы покажем вам как генерировать требуемый код.

2. В основном шрифты не бесплатны для использования. Вы должны платить за них и/или соблюдать другие условия лицензии такие как указание создателя шрифта в коде (или на вашем сайте). Вы не должны красть шрифты и использовать их без должного указания авторства.

Встраивание своего шрифта возможно несколькими способами.

1) Организовать загрузку файла шрифта на компьютер пользователя и отображение текста выбранным шрифтом. По сравнению с другими методами вроде отображения текста через рисунок этот способ самый простой и универсальный.

2) Использовать банки шрифтов. Например, Google Web Fonts. 977 свободных шрифтов, которые любой может использовать совершенно бесплатно. Коммерческие шрифты, зачастую, смехотворно дороги и обычно лицензируются, а не продаются, то есть плата взимается в зависимости от ожидаемого количества просмотров страницы. Поэтому иметь столько бесплатных и столь удобных в использовании шрифтов в одной коллекции — очень полезно. Google Fonts, как и многие провайдеры ресурсов для сайтов, хранит шрифты у себя давая возможность использовать их всем желающим непосредственно со своих серверов. Это означает, что вы можете начать использовать шрифты, просто добавив одну строку кода на вашем сайте, чтобы подгрузить таблицу стилей.

21. Создание текстов в издательских системах, основанных на TeX. Основные этапы работы.

TeX — это система для верстки текстов с формулами. TeX представляет собой специализированный язык программирования, на котором пишутся издательские системы. Каждая издательская система на базе TeX-а представляет собой пакет макроопределений (макропакет) этого языка. В частности, LaTeX - это издательская система на базе TeX-а. Наряду с LaTeX'ом распространены также макропакеты Plain-TeX и AMS-TeX.

Установка TeX

TeX и LaTeX доступны практически для всех платформ.

Наиболее популярными дистрибутивами LaTeX являются:

TeX Live — наиболее полный дистрибутив TeX/LaTeX для BSD, GNU/Linux, Mac OS X и Windows.

MiKTeX — дистрибутив для Microsoft Windows.

MacTeX — дистрибутив для Mac OS.

список онлайн сервисов LaTex:

ShareLaTeX; writeLaTeX; Papeeria; Verboses. LaTeX Base, overleaf.com.

LaTeX использует специальный язык разметки, преобразуя исходный текст вместе с его разметкой в документ высокого качества. Аналогичным образом формируются веб-страницы: исходный текст записывается с помощью языка HTML, а браузер открывает эту страницу уже во всей красе — с различными цветами, шрифтами, размерами и т.д.

TEX работает только с боксами (box) и клеем (glue). Элементарные боксы — это буквы, которые объединяются в боксы-слова, которые в свою очередь сливаются в боксы-строчки, боксы-абзацы и так далее. Между боксами «разлит» клей, который имеет ширину по умолчанию и степени увеличения/уменьшения этой ширины. Объединяясь в бокс более высокого порядка, элементарные боксы могут шевелиться, но после того как найдено оптимальное решение, это состояние замораживается и полученный бокс выступает как единое целое. Оптимальное решение находится с помощью системы штрафов за то, что клея больше или меньше, чем оптимальное значение, а также за разрывы абзаца в неподходящем месте. Чем меньше штрафа было получено, тем размещение «красивее». В зависимости от системы штрафов меняется форматирование.

LaTeX-документ

LATEX представляет из себя набор макросов на языке TEX. Основные правила создания текстового документа:

подготовить с помощью любого текстового редактора файл с текстом, расширение tex, оснащенным командами для LATEX’a.

обработать файл с помощью программы-транслятора; в результате получается файл с расширением dvi (device independent — не зависящий от устройства).

dvi-файл можно с помощью программ, называемых dvi-драйверами, распечатать на лазерном или струйном принтере, посмотреть на экране (текст будет в таком же виде, как он появится на печати) и т. д. Скомпилировать исходный файл (hi.tex) в файл документа, например в формате PDF (hi.pdf) с помощью PDFLaTeX или XeLaTeX.

Самый простой пример работающего исходного файла:

```
\documentclass{article}
\begin{document}
Привет, мир!
\end{document}
```

Для кириллических текстов после строки \documentclass потребуется также указать одну из следующих последовательностей команд:

- для XeLaTeX

\usepackage {fontspec}

\setmainfont [Ligatures=TeX] {DejaVu Serif}

- для обычного LaTeX

\usepackage [utf8x] {inputenc}

\usepackage [T2A] {fontenc}

Структура текста

Разбивка текста на разделы. Пример команд разделов:

\chapter{Введение}

Содержимое введения...

\section{Структура}

Содержимое раздела...

Некоторые разделы LaTeX генерирует автоматически. В частности, титульную страницу:

\maketitle

и содержание:

\tableofcontents

22. Создание текстов в издательских системах, основанных на TeX. Форматирование. Создание математических формул.

Появление ПК и издательского ПО произвело революцию в издательском деле и полиграфии, существенно изменив и упростив издательское дело. TeX является системой компьютерной верстки, в нее входят средства секционирования документов, для работы с перекрестными ссылками. Используя TeX пользователь задает текст и его структуру, а TeX самостоятельно на основе выбранного шаблона форматирует документ. Наиболее популярным макропакетом TeX является LaTeX(далее все на его примере).

Создание. Исходный текст не должен содержать переносов, слова разделяются пробелом. Чтобы LaTeX определил новый абзац необходимо отделить его пустой строкой. Любой файл LaTeX при создании начинается с команды \documentclass{в фигурных скобках указывается в виде чего будет оформлен документ(book / article / report(среднее между книгой и статьей) / proc(труды конференций) / letter)}

Чтобы получить печатное значение следующих символов: `{ } $ & # % _` необходимо поставить перед ними \ (без него выйдет сообщение об ошибке или символ засчитается как команда, например, % будет являться знаком комментария). Любые команды в LaTeX начинаются с \ , в том числе и для форматирования текста.

Форматирование.

После команды \documentclass{} следуют команды \usepackage{}, относящиеся ко всему документу и задающие параметры оформления текста.

Далее должны идти команды \begin{document} и \end{document}, между которыми располагается текст/формулы/таблицы и т.д.

Для выравнивания текста используются окружения center (для центрирования), а также flushleft и flushright (для выравнивания по левому и правому краю соответственно).

левый
марш

```
\begin{flushleft}
левый\\
марш
\end{flushleft}
```

наше дело
правое

```
\begin{flushright}
наше дело\\
правое
\end{flushright}
```

а вот мы позиционируемся
в центристской части
политического спектра

```
\begin{center}
а вот мы позиционируемся\\
в центристской части\\
политического спектра
\end{center}
```

Команда \textit{здесь необходимое слово или фраза} набирает свой аргумент курсивом
Команда \textbf{здесь необходимое слово или фраза} — полужирным шрифтом
Команда для создания абзацного отступа 2см \parindent=2cm
\section*{название заголовка} - создание заголовка первого уровня без нумерации
\subsection{название заголовка} - создание заголовка второго уровня с нумерацией
\subsubsection{название заголовка} - создание заголовка третьего уровня с нумерацией

Создание формул

Для набора линейных формул используют знак $, например, набираем $y'=f(x)$, получаем $y' = f(x)$. Для выключенных формул $$

Степени и индексы набираются с помощью знаков ^ и _ соответственно. Если индекс или показатель степени — выражение, состоящее более чем из одного символа, то его надо взять в фигурные скобки

Знак «больше или равно» генерируется коман- дой \ge, «меньше или равно» — командой \le

```
$$
  \frac{(a+b)^2}{4}-
  \frac{(a-b)^2}{4}=
  ab
$$
```

Дроби, в которых числитель расположен над знаменателем, набираются с помощью команды \frac{тут формула}. Эта команда имеет два обязательных аргумента: первый — числитель, второй — знаменатель.

корень набирается с помощью команды \sqrt[показатель корня]{тут подкоренное выражение}

Если формула очень высокая(например, дроби), то для нее нужны скобки подходящего размера, для их создания используются команды \left и \right

для создания такой формулы используются команды

$$e = \lim_{n\to\infty} \left(1+\frac{1}{n}\right)^n$$

```
$$
  e=\lim_{n\to\infty}
    \left(1+\frac{1}{n}\right)^n
$$
```

Команды для значков:

Греческие буквы. Совпадают с английскими названиями букв

α \alpha β \beta

логарифм \log \lg \ln

sin	\sin	arcsin	\arcsin	cos	\cos
arccos	\arccos	tan	\tan	arctan	\arctan
cot	\cot	sec	\sec	csc	\csc

$\sum$ \sum $\prod$ \prod

Пример формулы для интеграла:

```
$$
  \int_0^1x^2 dx=1/3
$$
```

будет напечатана как $$\int_0^1 x^2 dx = 1/3$$

Пример формулы с суммой:

$$\left(\sum_{k=1}^{n} x^k \right)^2$$

```
$$
\left(
\sum_{k=1}^n x^k
\right)^2
$$
```

Аббревиатуры:

(Cascading Style Sheets). CSS
HTML (HyperText Markup Language — «язык гипертекстовой разметки»)
JPEG (Joint Photographic Experts Group)
TCP Transmission Control Protocol
XML eXtensible Markup Language
RSS Rich Site Summary
JSON (англ. JavaScript Object Notation
WWW (World Wide Web)
AJAX Asynchronous Javascript and XML
DHCP Dynamic Host Configuration Protocol
API application programming interface