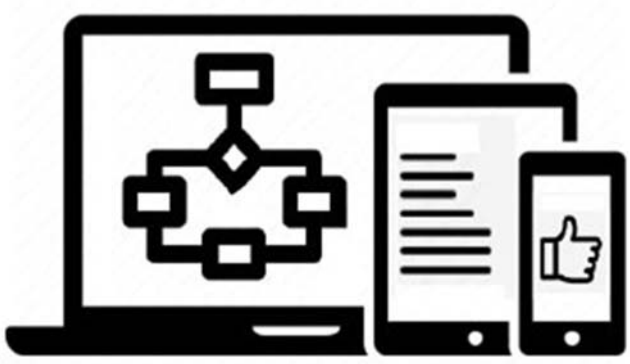
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

КОМПЬЮТЕРНЫЕ СЕТИ И ВЕБ-ТЕХНОЛОГИИ

Методические рекомендации к лабораторным работам для студентов специальности 1-28 01 02 «Электронный маркетинг» очной и заочной форм обучения



Могилев 2023

УДК 004.7
ББК 32.971.35
К63

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «20» октября 2023 г., протокол № 3

Составитель канд. техн. наук, доц. Э. И. Ясюкович

Рецензент В. М. Ковальчук

Методические рекомендации содержат основные базовые теоретические сведения, некоторые приемы реализации задач, а также практические задания для выполнения лабораторных работ по всем темам курса.

Учебное издание

КОМПЬЮТЕРНЫЕ СЕТИ И ВЕБ-ТЕХНОЛОГИИ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84 /16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Целью изучения дисциплины «Компьютерные сети и веб-технологии» является приобретение специальных знаний, умений и практических навыков, необходимых менеджеру по информационным технологиям для разработки интернет-приложений.

Язык ***html*** (***Hyper Text Markup Language***) используется для добавления разметки в обычный текст, позволяет создавать статические и динамические сайты и является языком, описывающим структуру и семантику веб-документа.

Стандарт языка ***html*** непрерывно обновляется и почти каждый год выходит его новая версия. Версия ***html*** 2.0 была опубликована в 1995 г., ***html*** 4.01 – основная версия ***html*** – в конце 1999 г. В настоящее время наиболее популярна версия ***html***-5, являющаяся расширением ***html*** 4.01.

Каскадные таблицы стилей (***Cascading Style Sheets*** – ***CSS***) влияют на отображение страниц в окнах браузеров (цвета, шрифты, фоновые изображения, интервалы между строками, отступы, границы, эффекты и даже анимация элементов). Благодаря ***CSS*** можно производить изменения, относящиеся ко всем страницам сайта, редактируя при этом лишь один единственный файл таблицы стилей.

Методические рекомендации предназначены для изучения основ веб-разработки и содержат краткий теоретический материал по вопросам соответствующей темы, предлагаются технологии решения некоторых типовых задач, приводятся задания для выполнения работ, а также контрольные вопросы.

В методических рекомендациях каждая лабораторная работа содержит краткие теоретические сведения для самостоятельной подготовки к ее выполнению, порядок выполнения, содержание отчета, варианты заданий и контрольные вопросы. Цель каждой работы соответствует названию, а ее выполнение производится в следующем порядке:

1) ознакомиться с основными теоретическими положениями;
2) выполнить задание, полученное у преподавателя;
3) по результатам выполнения каждой лабораторной работы следует оформить отчет, содержащий: титульный лист, постановку задачи, краткие теоретические сведения, описание хода выполнения работы, а также результаты ее выполнения и выводы.

Перед выполнением лабораторной работы рекомендуется изучить теоретический материал по теме, а также терминологию по теме работы.

1 Лабораторная работа № 1. Работа в Internet

Цель работы: изучение и повторение терминов и понятий, изучение интерфейса и настроек браузеров, а также изучение возможностей глобальной сети Internet.

Структура *Internet*. ***Internet*** – это глобальная компьютерная система, которая логически взаимосвязана пространством глобальных уникальных адресов, способна поддерживать коммуникации (обмен информацией) и обеспечивает работу высокоуровневых сервисов – служб, например, ***WWW***, электронная почта, телеконференции, разговоры в сети и др.

Структура интернет напоминает паутину, в узлах которой находятся компьютеры, связанные между собой линиями связи. Узлы интернет, связанные высокоскоростными линиями связи, составляют базис-интернет. Оцифрованные данные пересылаются через маршрутизаторы, которые соединяют сети с помощью сложных алгоритмов, выбирая маршруты для информационных потоков.

Сервер в сети Интернет – это компьютер, обеспечивающий обслуживание пользователей и предоставляющий услуги другим компьютерам, запрашивающим информацию. Таких пользователей называют клиентами, пользователями или абонентами. При этом работа в интернете предполагает наличие передатчика информации, приемника и канала связи. Когда мы входим в интернет, наш компьютер выступает в качестве клиента, он запрашивает необходимую нам информацию на выбранном сервере.

Состав служб, называемых также сервисами ***Internet***. В настоящее время в ***интернете*** существует достаточно большое количество сервисов, обеспечивающих работу со всем спектром ресурсов. Наиболее известными среди них являются:

– электронная почта – ***E-mail***, обеспечивающая возможность обмена сообщениями одного человека с одним или несколькими абонентами;

– телеконференции, или группы новостей ***Usenet***, обеспечивающие возможность коллективного обмена сообщениями;

– сервис ***FTP*** – система файловых архивов, обеспечивающая хранение и пересылку файлов различных типов;

– сервис ***Telnet***, предназначенный для управления удаленными компьютерами в терминальном режиме;

– ***World Wide Web*** (***WWW***, ***W3***) – гипертекстовая, гипермедиа-система, предназначенная для интеграции различных сетевых ресурсов в единое информационное пространство;

– сервис ***DNS***, или система доменных имен, обеспечивающий возможность использования для адресации узлов сети мнемонических имен вместо числовых адресов;

– сервис ***IRC***, предназначенный для поддержки текстовых общений в реальном времени, – ***chat***.

Перечисленные сервисы относятся к стандартным. Это означает, что принципы построения клиентского и серверного программного обеспечения, а также протоколы взаимодействия, сформулированы в виде международных стандартов. Следовательно, разработчики программного обеспечения при практической реализации обязаны выдерживать общие технические требования.

Наряду со стандартными сервисами, существуют и нестандартные, представляющие собой оригинальную разработку той или иной компании. В качестве примера можно привести различные системы типа ***Instant Messenger*** (своеобразные интернет-пейджеры – ***ICQ***, ***AOl***, ***Demos on-line*** и т. п.), системы интернет-телефонии, трансляции радио и видео и т. д. Важной особенностью таких систем является отсутствие международных стандартов, что может привести к возникновению технических конфликтов с другими подобными сервисами.

FQDN (***Fully Qualified Domain Name*** – полностью определённое имя домена) – это имя домена, не имеющее неоднозначностей в определении, которое включает в себя имена всех родительских доменов иерархии ***DNS***.

Различие между ***FQDN*** и доменным именем появляется при именовании доменов второго, третьего уровня и т. д. Для получения ***FQDN*** требуется обязательно указать в имени домены более высокого уровня. Например, ***sample*** является доменным именем, однако ***FQDN*** имя выглядит как

sample.gtw-02.office4.example.com.

Структура ***DNS*** (***Domain Name System***). ***DNS*** – это распределённая система, которая помогает браузерам находить адреса сайтов по их именам. Работает она благодаря разветвлённой сети серверов, на которых хранится информация обо всех сайтах интернета. То есть сам принцип работы остался прежним – база данных (табличка) с перечнем всех сайтов мира, просто заносятся в неё данные уже не вручную одним оператором, а автоматически.

Например, когда мы заходим на сайт ***Skillbox***, мы вводим в адресной строке его имя: ***skillbox.ru***. Но браузер не сможет подключить нас к нему просто по имени, потому что ему нужен ***IP***-адрес.

На самом деле весь интернет работает именно с системой IP-адресов, а привычные нам названия нужны только для удобства: их проще запоминать и вводить. Поэтому все, что вам нужно – это написать в адресной строке браузера ***skillbox.ru*** и нажать ***Enter***. Дальше браузер всё сделает сам: отправит нужные запросы, разыщет ***IP***-адрес и откроет страницу.

Назначение ***html***. Язык гипертекстовой разметки сайта, или ***html*** (***HyperText Markup Language***) – это код, помогающий структурировать содержание каждой веб-страницы интернета. С помощью HTML разработчики собирают скелет сайта, после чего работают с CSS-кодом для стилизации страницы. В финале запускается система, программируя на языке JavaScript или на других языках.

Порядок выполнения работы

1 Ознакомиться со списком литературных источников по дисциплине.

2 Изучить и выписать состав меню двух браузеров, например ***MS Internet Explorer*** и ***Mozilla Firefox***, либо других.

3 Найти в интернете или в литературе различные варианты определений, содержащие не менее двух источников, следующих понятий:

– ***Internet***;
– адрес ***IP v***.4;
– маска ***IP***-адреса;
– адрес ***IP*** v.6;
– ***DNS***;
– ***WWW***;
– сайт;
– ***html***;
– ***CSS***.

4 Оформить отчет в виде документа ***MS Word*** с гиперссылками на источники найденных определений. При оформлении отчета использовать возможности ***MS Word*** по форматированию текстовых документов.

Контрольные вопросы

1 Какова структура ***Internet***?
2 За что отвечают протоколы ***TCP/IP***?
3 Каков состав служб (сервисов) ***Internet***?
4 Что такое ***FQDN***?
5 Назовите основные службы ***Internet***.
6 Что такое гипертекст?
7 Какова структура ***dns***?
8 Каков состав ***FQDN***?
9 Назначение ***html***.
10 Как в браузере:
– настроить параметры соединения с Internet;
– просмотреть ***html***-код страницы;
– изменить кодировку для интерпретации страницы;
– просмотреть список загрузок;
– задать стили пользователя (форматирование пользователя)?

2 Лабораторная работа № 2. Изучение сетевого уровня модели OSI на примере протокола IP

Цель работы: изучение правил адресации сетевого уровня, практических навыков распределения адресов между участниками сетевого взаимодействия, а также правил организации маршрутизации между сегментами сети.

Модель ***OSI*** (***Open System Interconnected***) – это концептуальная модель сети, описывающая ее уровни и функции.

На практике модель ***OSI*** не используется, однако она служит стандартом для организации других моделей. Эта модель замечательно подходит для описания всех процессов, происходящих в компьютерных сетях.

Модель ***OSI*** является открытой системой, т. е. она основана на открытых спецификациях (стандартах).

В общем случае модель подразумевает разделение на уровни. Уровни в модели ***OSI*** расположены иерархически, один за другим. Каждый уровень определяет задачи более общего вопроса.

Каждый уровень модели ***OSI*** предоставляет вышестоящим уровням так называемый сервис – набор услуг. Услуга в данном случае описывает какую-либо функцию, предназначенную для вышестоящего уровня.

Наиболее популярной является открытая сетевая модель ***OSI*** (***Open Systems Interconnection model***), которая состоит из семи уровней и является эталонной.

Все уровни ***OSI*** можно условно разделить на две группы: Media layers – уровни среды (***L*1**, ***L*2**, ***L*3**) и Host layers – уровни хоста (***L*4**, ***L*5**, ***L*6**, ***L*7**).

Уровни группы ***Media Layers*** занимаются передачей информации – по кабелю или беспроводной сети, и используются сетевыми устройствами, такими как коммутаторы, маршрутизаторы и т. п. Уровни группы ***Host Layers*** используются непосредственно на устройствах: стационарных компьютерах или портативных мобильных устройствах.

Уровни открытой сетевой модели имеют следующее назначение.

1 Физический уровень ***L*1** отвечает за физический обмен данными между компьютерами.

2 Канальный уровень ***L*2**. На втором уровне ***OSI*** работают коммутаторы, которые передают сформированные кадры от одного устройства к другому, используя только физические ***MAC***-адреса.

3 В сетевом уровне ***L*3** добавляется новое понятие – маршрутизация, выполняющаяся маршрутизаторами, являющимися роутерами. Маршрутизаторы получают ***MAC***-адреса от коммутаторов с предыдущего уровня и строят маршрут от одного устройства к другому. Используется протокол ARP, с помощью него 64-битные ***MAC***-адреса преобразуются в 32-битные ***IP***-адреса и наоборот. Тем самым происходит инкапсуляция и декапсуляция данных.

4 Транспортный уровень ***L*4** является неким посредником между Media- и Host-уровнями. Его главной задачей является транспортировка пакетов.

Используется протокол ***TCP***, он контролирует целостность доставленной информации. При передаче по протоколу ***TCP*** данные делятся на сегменты. Когда приходит пакет, который превышает пропускную способность сети, протокол делит его на сегменты допустимого размера. Также существует сегментация пакетов и используется протокол ***UDP*** (***User Datagram Protocol***). Главным преимуществом протокола ***UDP*** является отсутствие траты времени на установку соединения с удаленным узлом.

5 Сеансовый уровень ***L5***. Пятый уровень, как и шестой и седьмой, оперирует чистыми данными. Он управляет взаимодействием между приложениями и может запрашивать выполнение действий на удаленных компьютерах.

6 Уровень представления данных ***L6*** представляет данные, которые все еще являются ***PDU***, в понятном и машине и человеку виде. Допустим, что одно устройство может отображать текст только в ***ASCII***, а другое только в ***UTF-8***, а перевод текста из одной кодировки в другую происходит именно на шестом уровне. Кроме этого, на данном уровне производится работа с картинками и аудиоформатом, а также шифрование данных при передаче.

7 Прикладной уровень ***L7*** является своего рода графическим интерфейсом ***OSI***, задачей которого является использование всех своих протоколы для понятной передачи информации пользователю.

Сетевой уровень отвечает за возможность доставки пакетов по сети передачи данных – совокупности сегментов сети, объединенных в единую сеть любой сложности посредством узлов связи.

В связи с необходимостью перенаправлять пакеты из одного сегмента сети в другой, сетевые адреса должны удовлетворять следующим требованиям:

– адреса должны быть уникальны. В сети не может быть нескольких участников с одинаковыми адресами во избежание неоднозначности;

– сетевой адрес должен содержать информацию о том, как достичь получателя по сети.

Это приводит к структурированию (разбивке) сетевого адреса на части, позволяющие определить местоположение участника внутри сети.

Структура сетевого адреса может быть многоуровневой, например адрес содержит информацию о стране, области, населенном пункте, предприятии, здании, отделе и т. д., или простой, содержащей номер сети и номер компьютера в сети.

По сложной структуре легче построить маршрут прохождения пакета, но адрес оказывается сложным и перегруженным часто ненужной информацией. Примером такой адресации может служить доменная адресация в интернете, по адресу ***asu.bru.mogilev.by*** нетрудно понять, где находится данный участник сети и как до него добраться.

Простая структура позволяет значительно сократить размер адреса и сохраняет возможность работы в сети любой структуры, но для этого могут потребоваться сложные и, часто не столь очевидные алгоритмы, как в предыдущем случае.

Протокол ***IP***. Архитектуру сетевого уровня удобно рассматривать на примере сетевого протокола ***IP*** – самого распространенного в настоящее время,

основного протокола сети Интернет. Термин «Стек протоколов ***TCP/IP***» означает «Набор протоколов, связанных с **IP** и ***TCP***» – протоколом транспортного уровня.

Архитектура протоколов ***TCP/IP*** предназначена для объединенной сети, состоящей из соединенных друг с другом шлюзами отдельных разнородных пакетных подсетей, к которым подключаются разнородные машины.

Каждая из подсетей работает в соответствии со своими специфическими требованиями и имеет свои средства связи. Однако предполагается, что каждая подсеть может принять пакет информации – данные с соответствующим сетевым заголовком, и доставить его по указанному адресу в указанной подсети.

Таким образом, два компьютера, подключенные к одной подсети, могут обмениваться пакетами.

Если необходимо передать пакет между компьютерами, подключенными к разным подсетям, то компьютер-отправитель посылает пакет в соответствующий шлюз, который подключен к подсети как обычный узел. Далее пакет направляется по определенному маршруту через систему шлюзов и подсетей, пока не достигнет шлюза, подключенного к той же подсети, что и машина-получатель, направляющая пакет к получателю.

Таким образом, адрес получателя должен содержать:

– номер (адрес) подсети;

– номер (адрес) участника (хоста) внутри подсети.

IP-адреса представляют собой 32-разрядные двоичные числа. Для удобства их записывают в виде четырех десятичных чисел, разделенных точками или с помощью метода Горнера:

(((192*256) + 168)*256 + 200)*256 + 47 = 3232286767

Количество разрядов адреса подсети может быть различным и определяется маской сети.

Маска сети также является 32-разрядным двоичным числом. Разряды маски имеют следующий смысл: если разряд маски равен 1, то соответствующий разряд адреса является разрядом адреса подсети, а если 0 – то разрядом хоста внутри подсети. Все единичные разряды маски, если они есть, находятся в старшей – левой части маски, а нулевые, если они есть, – в ее правой, младшей части.

В таблице 2.1 приведены основные маски и размеры подсетей.

Например, «/24» – это префикс маски подсети (краткая запись маски). Полная запись маски подсети 255.255.255.0. В двоичном отображении маска подсети выглядит так: 11111111.11111111.11111111.00000000 – это значит, что нам доступно 8 бит для деления сети.

Таблица 2.1 – Маски и размеры подсетей

Маска	Десятеричная запись	# подсетей	# адресов	Класс
/17	255.255.128.0	2	32 K	128 C
/18	255.255.192.0	4	16 K	64 C
/19	255.255.224.0	8	8 K	32 C
/20	255.255.240.0	16	4 K	16 C
/21	255.255.248.0	32	2 K	8 C
/22	255.255.252.0	64	1 K	4 C
/23	255.255.254.0	128	512	2 C
/24	255.255.255.0	256	256	1 C
/25	255.255.255.128	2	128	1/2 C
/26	255.255.255.192	4	64	1/4 C
/27	255.255.255.224	8	32	1/8 C
/28	255.255.255.240	16	16	1/16 C
/29	255.255.255.248	32	8	1/32 C
/30	255.255.255.252	64	4	1/64 C

Определение диапазона адресов подсети.

IP-адрес – это массив битов. Принцип ***IP***-адресации – выделение диапазона ***IP***-адресов, в котором некоторые битовые разряды имеют фиксированные значения, а остальные разряды пробегают все возможные значения. Блок адресов задается указанием начального адреса и маски подсети. Бесклассовая адресация основывается на переменной длине маски подсети (***Variable Length Subnet Mask*** – ***VLSM***), в то время, как в традиционной адресации длина маски строго фиксирована 0, 1, 2 или 3 установленными октетами.

Определить диапазон адресов подсети можно по ее маске:

– разряды, которые относятся к адресу подсети, у всех хостов подсети должны быть одинаковы;

– адреса хостов в подсети могут быть любыми.

То есть, если наш адрес 192.168.200.47 и маска равна /20, то диапазон можно посчитать:

11000000.10101000.11001000.00101111 – адрес → 192.168.200.47
11111111.11111111.11110000.00000000 – маска →(/20 → 255.255.240.0)
11000000.10101000.1100***XXXX.XXXXXXXX*** – диапазон адресов,

где 0,1 – определенные значения разрядов;

X – любое значение,

что приводит к диапазону адресов:

от 11000000.10101000.11000000.00000000 (192.168.192.0)
до 11000000.10101000.11001111.11111111 (192.168.207.255)

Следует учитывать, что некоторые адреса являются запрещенными или служебными и их нельзя использовать для адресов хостов или подсетей. Это адреса, содержащие:

– 0 в первом или последнем байте;

– 255 в любом байте (это широковещательные адреса);

– 127 в первом байте (внутренняя петля – этот адрес имеется в каждом хосте и служит для связывания компонентов сетевого уровня).

Поэтому доступный диапазон адресов будет несколько меньше.

Порядок выполнения работы

1 Определить, какие адреса из приведенного ниже списка являются допустимыми адресами хостов:

0.10.10.10
10.0.10.10
10.10.0.10
10.10.10.10
127.0.127.127
127.0.127.0
255.0.200.1
1.255.0.0

2 Перечислить все допустимые маски.

3 Определить диапазоны адресов подсетей (даны адрес хоста и маска подсети):

10.212.157.12/24
27.31.12.254/31
192.168.0.217/28
10.7.14.14/16

4 Определить, какие из адресов будут достигнуты напрямую с хоста 242.254.169.212/21:

241.253.169.212
243.253.169.212
242.252.169.212
242.254.169.212
242.253.168.212
242.253.170.212
242.253.169.211
242.253.169.213

5 Посмотрите параметры IP на своем компьютере с помощью команды ipconfig.

Определить диапазон адресов и размер подсети, в которой Вы находитесь. Объяснить, почему выбраны такие сетевые параметры и какие сетевые параметры выбрали бы Вы.

Контрольные вопросы

1 Для чего используется сетевой уровень?
2 Что такое сеть передачи данных?
3 Какие требования предъявляются к сетевой адресации?
4 Можно ли использовать в качестве сетевого адреса ***MAC***-адрес?
5 Что такое маска подсети?
6 Какова структура ***IP***-адреса?
7 Чем определяется размер подсети?
8 Как определить диапазон адресов в подсети?
9 Как определить размер подсети?
10 Для чего используется ***URL***?

3 Лабораторная работа № 3. Изучение маршрутизации IP

Цель работы: изучить правила адресации сетевого уровня, научиться распределять адреса между участниками сети передачи данных и организовывать маршрутизацию между сегментами сети.

Основное назначение протокола ***IP*** – это объединение отдельных разнородных пакетных подсетей канального уровня: ***Ethernet***, ***Token Ring***, ***FDDI***, ***Frame Relay***) в составную сеть.

В любой из разнородных подсетей пакет информации (данные с соответствующим сетевым заголовком) может быть доставлен по указанному адресу в этой конкретной подсети с использованием механизмов и свойств технологий канального уровня. Таким образом, две машины, подключенные к одной подсети, могут обмениваться пакетами.

Когда необходимо передать пакет между машинами, подключенными к разным подсетям, то машина-отправитель посылает пакет в соответствующий шлюз (шлюз подключен к подсети так же, как обычный узел).

Шлюз (***Gateway***) – это любое сетевое оборудование с несколькими сетевыми интерфейсами, осуществляющее продвижение пакетов между сетями на уровне протоколов сетевого уровня.

Из шлюза пакет направляется по определенному маршруту через систему шлюзов и подсетей, пока не достигнет шлюза, подключенного к той же подсети, что и компьютер-получатель.

Таким образом, шлюз выполняет маршрутизацию – процедуру нахождения в структуре сети пути достижения получателя, т. е. построение пути доставки пакетов.

Для продвижения пакетов по тому или иному маршруту шлюз использует таблицу маршрутизации, состоящую из отдельных записей – правил маршрутизации.

Правила маршрутизации определяют, куда и как должны посылаться пакеты для разных сетей.

Каждое правило состоит из следующих компонентов.

1 Начальный адрес подсети, порядок достижения которого описывает правило.

2 Маска подсети, которую описывает правило.

3 Шлюз показывает, на какой адрес будут посланы пакеты, идущие в сеть назначения. Если пакеты будут идти напрямую, то указывается собственный адрес (точнее адрес того канала, через который будут передаваться пакеты).

4 Интерфейс показывает, через какой сетевой адаптер (его номер или ***IP***-адрес) должен посылаться пакет в заданную сеть.

5 Метрика показывает время, за которое пакет может достигнуть сети получателя (величина условная и может быть изменена при маршрутизации).

Если имеется несколько правил достижения одной сети, пакеты посылаются по правилу с наименьшей метрикой.

Применение правила заключается в определении, какой подсети (сети) принадлежит хост назначения, указанный в принимаемом пакете. Далее, согласно правилу в таблице маршрутизации, направить пакет на адрес шлюза через соответствующий интерфейс.

Правила маршрутизации сведены в таблицу маршрутизации (где расположены по степени уменьшения маски), которую можно посмотреть с помощью команды route print.

Порядок выполнения работ

1 С помощью программы ***route print*** посмотрите таблицу маршрутизации Вашего компьютера. Объясните все правила.

2 Посмотрите таблицу маршрутизации хоста, имеющего несколько каналов. Объясните все правила.

3 Посмотрите таблицу маршрутизации маршрутизатора. Объясните все правила.

4 Добавьте новое правило в таблицу маршрутизации для сети 192.168.0.0/24 через шлюз в Вашей сети с последним байтом в адресе 125 и метрикой 12.

5 Удалите это правило.

6 В соответствии с таблицей и схемами выполните задание на распределение адресов по подсетям согласно варианту. Постройте таблицы маршрутизации для всех шлюзов и для одного хоста каждого сегмента.

Контрольные вопросы

1 Сколько адресов может иметь хост?

2 Может ли у хоста быть прописано несколько шлюзов и почему?

3 Может ли у хоста быть прописано несколько шлюзов по умолчанию и почему?

4 Чем отличаются таблицы у разных классов сетевых устройств и почему?

5 Почему начальный адрес подсети должен быть кратен ее размеру?

6 Чем Вы руководствовались при выборе шлюзов по умолчанию?

7 Может ли физический сегмент сети содержать несколько сетевых подсетей?

8 Из чего состоит ***IP***-адрес?

9 Какие Вы знаете виды подключения к ***Internet***?

10 Можно ли по ***IP***-адресу открыть известный Вам сайт?

4 Лабораторная работа № 4. html. Оформление web-страницы средствами html

Цель работы: освоение синтаксиса ***html*** и правил оформления **web**-страниц, содержащих таблицы и изображения.

Web-страницы используют язык ***html***. ***html***-документ – это файл, содержащий обыкновенный структурно размеченный текст. Такой файл может быть подготовлен в произвольном текстовом редакторе.

Каждая управляющая конструкция ***html***-документа должна заключаться в угловые скобки: <тег>. Чаще всего в документе встречаются парные теги: открывающий и соответствующий ему закрывающий, т. к. браузеру необходимо знать область действия тега.

Открывающий и закрывающий теги называются одинаково и отличаются друг от друга только слеш символом «/», который ставится сразу после открывающей угловой скобки закрывающего тега. Закрытие парных тегов выполняется так, чтобы соблюдались правила вложения:

На этот текст воздействуют два тега

Кроме того, тег может включать атрибуты, предоставляющие браузеру дополнительную информацию.

Атрибуты представляют собой дополнительные ключевые слова, отделяемые пробелами от ключевого слова, определяющего тег, от других атрибутов и размещаемые до завершающего тег символа «>». Значение атрибута отделяется от ключевого слова атрибута символом «=» и заключается в кавычки, например

<h1 align="left">

Списки. В языке ***html*** используются следующие виды списков:

– упорядоченные (нумерованные);

– неупорядоченные (ненумерованные);

– списки определений.

Перед пунктами неупорядоченных списков обычно используются символы-маркеры (***bullets***), например, точки, ромбики и т. п., в то время как пунктам упорядоченных списков предшествуют их номера.

Таблица 4.1 – Теги списков

Тег	Назначение
*** – ***	Создает неупорядоченный список
*** – ***	Создает упорядоченный список
*** – ***	Создает пункт списка внутри тегов ***ol*** или ***ul***
<dl> – </dl>	Открывает и закрывает список определений
<dt> – </dt>	Создает термин в списке определений внутри элемента ***dl***
<dd> – </dd>	Создает определение термина внутри элемента ***dl***

Таблицы. Таблица состоит из строк, содержащих ячейки, которые могут содержать текст и рисунки. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на ***web***-страницу используется тег-контейнер <***table***>, который служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью элементов <***tr***> и <***td***>. Внутри тега <***table***> допустимо использовать следующие элементы: <***caption***>, <***col***>, <***colgroup***>, <***tbody***>, <***td***>, <***tfoot***>, <***th***>, <***thead***> и <***tr***>.

Для добавления строк используются теги <***tr***> и ***</tr>***. Чтобы разделить строки на колонки применяются теги <***td***> и ***</td>***.

Для изменения вида и свойств таблицы использовались атрибуты, которые добавлялись в тег <***table***>: <***table*** атрибут1 =... атрибут 2=...> (таблица 4.2).

Таблица 4.2 – Атрибуты таблицы и их значения

Атрибут	Описание	Пример
align = left right center	Выравнивание таблицы	align="center"
background = URL	Фоновый рисунок	background="pic.gif"
bgcolor = #rrggbb	Цвет фона таблицы	bgcolor="#FF9900"
border = n	Толщина рамки в пикселах	border="2"
cellpadding = n	Расстояние между ячейкой и ее содержимым	cellpadding="7"
cellspacing = n	Дистанция между ячейками	cellspacing="3"
colspan = n	Число ячеек, объединяемых по горизонтали	colspan="2"
rowspan = n	Число ячеек, объединяемых по вертикали	rowspan="3"
nowrap	Запрещает переносы строк в тексте	<table nowrap>
valign = top bottom	Выравнивание по высоте	valign="top"

Окончание таблицы 4.2

Атрибут	Описание	Пример
width = n, n%	Минимальная ширина таблицы, в пикселах или процентах	width="90%"
height = n, n%	Минимальная высота таблицы, в пикселах или процентах	height="18

Порядок выполнения работы

Создать ***html***-документ – ***rasp.html*** с расписанием занятий (таблица 4.3), который должен начинаться заголовком «Расписание занятий гр. ***NNN*** на _____ семестр 20___ г.». Первая строка таблицы должна быть оформлена как заголовки полей с использованием тегов <***th***>, который используется в таблицах внутри тегов <***tr***> для создания заголовочной ячейки. Текст, введенный в заголовочную ячейку, по умолчанию отображается жирным шрифтом и выравнивается по центру ячейки.

Таблица 4.3 – Расписание занятий гр. _____ на _____ семестр 20___ г.

День недели	Время	Предмет	Преподаватель	Аудитория
Понедельник	08:30 – 10:05	Математика – лекция	доц. Иванов А. А.	517
	10:25 – 12:00	Математика – практика	преп. Петрова И. А.	518
	12:30 – 14:05	Физика – лабораторная	доц. Сидоров О. И.	518
Вторник	10:25 – 12:00	История – лекция	проф. Громова О. А.	519
	12:30 – 14:05	История – семинар	преп. Попов М. А.	517
	14:20 – 15:55	Физика – лабораторная	доц. Сидоров О. И.	517
...	...	…	…	..
Пятница	12:30 – 14:05	Физика – лекция	проф. Терехов М. В.	518
	14:20 – 14:55	География – семинар	преп. Фролов И. С.	519
	16:05 – 17:40	Химия – лабораторные	доц. Симонов В. М.	519

Таблица должна занимать полный размер по ширине окна. Ширину отдельных столбцов задать в относительных единицах – в процентах, для того, чтобы при изменении ширины окна сохранялись пропорции таблицы.

После создания ***html***-документа ***rasp.html*** с расписанием занятий:

1) просмотреть созданный документ в браузере при различных размерах окна и различных настройках шрифта;

2) построить таблицу 4.4 «Расписание», содержащую пять столбцов: «День недели», «Время», «Предмет», «Аудитория» и «Тема занятия»;

3) построить гиперссылку для считывания из выбираемой строки «День недели» значений параметров «Время», «Предмет», «Аудитория», соответствующих выбранному кликом мыши по дню недели в таблице 4.3, и ввести с клавиатуры значение элемента «Тема занятия»;

4) вставить в таблицу 4.4 значения элемента, введенного в п. 3;

5) построить ***html***-документ, содержащий выбранные значения, и установить заданные преподавателем параметры выбранных элементов;

6) выполнить цветовое оформление каждого элемента меню;

7) сохранить построенный ***html***-документ с именем ***rasp02.html***;

8) проверить правильность выполнения переходов по гиперссылкам.

Таблица 4.4 – Расписание

День недели	Время	Предмет	Аудитория	Тема занятия

Контрольные вопросы

1 Что понимают под ***html***-разметкой?
2 Приведите примеры парных и непарных тегов.
3 В каких единицах измерения задается значение атрибута ***size***?
4 При использовании какого тега появляются вертикальные отступы?
5 Чем отличаются абсолютные и относительные ссылки?
6 Какие существуют виды списков?
7 Какие Вы знаете виды подключения к ***Internet?***
8 Для чего нужна маршрутизация в ***Internet***?
9 Опишите известные Вам команды тестирования ***web***-ресурсов.
10 Опишите принцип функционирования сервера ***DNS***.

5 Лабораторная работа № 5. CSS

Цель работы: изучение порядка создания и использования ***CSS*** при разработке ***web***-страниц.

CSS – это набор правил форматирования ***web***-страницы, который может быть применен к различным элементам страницы.

В ***html*** для присвоения какому-либо элементу определенных свойств (таких как цвет, размер, положение на странице и т. п.) приходится каждый раз описывать эти свойства.

Применяя ***CSS***, можно один раз описать свойства и их значения и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, оформляемый соответствующим образом, должен принять необходимые свойства стиля (таблица 5.1).

Описание стиля можно сохранить не в тексте страницы, а в отдельном файле – это позволит использовать описание стиля на любом количестве ***web***-страниц.

Описания стилей в ***web***-странице должны находиться в тегах <***style***> <***/style***>, которые размещаются между тегами <***head***> <***/head***>.

Таблица 5.1 – Свойства ***CSS***

Свойство	Назначение	Пример
font-family	Используется для указания шрифта или шрифтового семейства, которым будет отображаться элемент	***p {font-family: Verdana, sans-serif;}***
font-weight	Определяет степень насыщенности шрифта: ***bold|bolder|lighter|normal***|100|200|300|400|500|600|700|800|900	***b {font-weight: bolder;}***
font-size	Устанавливает размер шрифта. Параметр может указываться в процентах, пикселях или сантиметрах	***h1 {font-size: 200%;}***
text-decoration	Устанавливает эффекты оформления шрифта, такие как подчеркивание или зачеркивание текста	***h4 {text-decorati-on: underline;}*** (подчеркивание)
text-align	Определяет выравнивание элемента	***p {text-align: left}*** (выравнивание по левому краю) ***p {text-align: right}*** (выравнивание по правому краю)
text-indent	Устанавливает отступ первой строки текста. Чаще всего используется для создания параграфов с табулированной первой строкой	***h1 {text-indent: 60 pt;}***
line-height	Управляет интервалами между строками текста	***p {line-height: 50 %}***
color	Определяет цвет элемента	h3 {color: #0000FF;}
background-color	Устанавливает цвет фона для элемента	***<h3 style = "background-color:gold; color:brown;"> Пример </h3>***
margin-left margin-right; margin-top margin-bottom	Устанавливают значения отступов вокруг элемента	***img { margin-left: 20pt} img { margin-right: 20pt}***

Example.css – это ***CSS***-файл, содержащий описание применяемых стилей. Создается CSS-файл в любом текстовом редакторе, например, в ***Notepad***++, нужно только изменить расширение текстового файла на ***CSS***. В ***CSS***-файле не должны использоваться теги <***style***> <***/style***>.

Можно определить стиль для любого тега отдельно. Для этого нужно в тег добавить атрибут и описать его стиль в кавычках: ***style = ""***.

Следующий пример отображает слово «Пример» шрифтом ***Verdana***, размером 150 % и красным цветом:

<h3 style=" font-size:150%; color:red">Пример</h3>

Свойства и назначение основных селекторов ***CSS*** и приведены в таблице 5.1.

Порядок выполнения работы

1 Ознакомиться с основами применения ***CSS***.

2 Выполнить задание из лабораторной работы № 5 с использованием ***CSS***.

3 Использовать все способы подключения ***CSS*** и основные типы селекторов.

4 Сделать вывод о размере кода ***html*** с применением ***CSS***, а также о целесообразности применения ***CSS*** в конкретном случае.

5 Оформить отчет в виде ***html***-файла и двух файлов ***CSS***.

Контрольные вопросы

1 Какова структура правила CSS?
2 Как расшифровывается CSS?
3 Из чего состоит правило таблицы стилей?
4 Как выглядит свойство селектора?
5 Что такое стиль и таблица стилей?
6 Что такое селектор?
7 Способы подключения таблиц стилей CSS.
8 Какие существуют типы селекторов?
9 Для чего используется группирование в CSS?
10 У какого селектора комбинатор #?

6 Лабораторная работа № 6. CSS. Каскадирование

Цель работы: изучить механизм каскадирования в ***CSS***.

В соответствии со спецификацией ***CSS*** имеется несколько способов подключения таблиц стилей к ***html***-документу. Может получиться, что возникнет конфликт – на одно свойство конкретного элемента претендуют несколько разных значений из разных правил – таблицы стилей.

Для разрешения конфликтов значений свойств в ***CSS*** определен механизм (порядок, алгоритм) каскадирования:

– по важности – явной приоритетности;
– по источнику правил;
– по специфичности;
– по расположению.

Рассмотрим приведенные элементы структуры механизма каскадирования.

1 Важность объявления – явная приоритетность, является настолько значительной, что перевешивает все остальные факторы. ***CSS*** называет эти факторы важными объявлениями (***important declarations***) и предоставляет возможность отмечать их путем введения в объявление ключевого слова !***important*** прямо перед завершающей точкой с запятой:

p.dark {color: #333 !important;
background: white;}

Необходимо правильно размещать ***!important***, иначе объявление будет признано недействительным. Ключевое слово ***!important*** всегда располагается в конце объявления, прямо перед точкой с запятой. Это особенно важно, когда дело доходит до свойств, например, ***font***, значения которых могут состоять из нескольких ключевых слов:

p.light (color: yellow;
font: smaller Times, serif !important;}

Если бы ***!important*** было расположено где-либо в другом месте объявления ***font***, то все объявление было бы признано недействительным и ни один из его стилей не был бы применен.

Объявления, отмеченные как ***!important***, не имеют особого значения специфичности, они рассматриваются отдельно от остальных. Фактически все объявления ***!important*** группируются вместе и тогда уже их конфликты специфичностей разрешаются относительно друг друга. Аналогично группируются все остальные объявления и конфликты свойств разрешаются с помощью специфичностей.

2 Источник правила. Существуют три возможных источника правил: автор, читатель и браузер пользователя. На приоритетность правил влияет инструкция ***!important***.

Таким образом, с точки зрения приоритетности объявлений выделяют пять их уровней, которые представим в порядке уменьшения приоритетности:

1) важные объявления пользователя;
2) важные объявления автора;
3) обычные объявления автора;
4) обычные объявления пользователя;
5) объявления браузера пользователя.

3 Специфичность. Для каждого правила браузер пользователя вычисляет специфичность (***specificity***) селектора и прикрепляет ее к каждому объявлению правила.

Специфичность селектора определяется компонентами селектора.

Значение специфичности состоит из четырех частей: 0,0,0,0 и разбивается на четыре уровня: ***a***, ***b***, ***c*** и ***d***:

– если стиль встроенный (***inline***), ***a*** = 1;

– значение ***b*** равно общему количеству селекторов идентификаторов;

– значение ***c*** равно количеству классов, псевдоклассов и селекторов атрибутов;

– значение ***d*** равно количеству селекторов типов и псевдоэлементов.

4 Порядок расположения. Если два правила имеют одинаковые приоритетность и специфичность, тогда побеждает то из них, которое расположено в таблице стилей позже.

С этих позиций принимается, что стили, определенные в атрибуте ***style*** элемента, находятся в самом конце таблицы стилей документа, т. е. размещаются после всех остальных правил и поэтому имеют более высокую специфичность, чем любой селектор таблицы стилей.

Порядок выполнения работы

1 Подготовить ***web***-документ, при форматировании которого применяется механизм каскадирования.

2 Применяемые ***CSS***-механизмы каскадирования пояснить комментариями.

3 Оформить отчет в виде файла *.***html** и двух файлов *.***css**.

Контрольные вопросы

1 Когда применяется механизм каскадирования?
2 Как задать явную приоритетность?
3 По какому компоненту правила вычисляется специфичность?
4 Какие существуют источники правил?
5 Как подключить пользовательский стиль в браузере?
6 Как задать значения свойствам селектора?
7 Как задать множество свойств для одного селектора?
8 Дайте определение понятия селектора класса.
9 Что такое селектор потомков и как он формируется?
10 Как задать вид, цвет рамок и фон объекта?

7 Лабораторная работа № 7. CSS. Модель визуального форматирования

Цель работы: изучить возможности модели визуального форматирования в ***CSS*** и основных типов позиционирования в ***CSS***.

В интернет-программировании используются следующих четыре свойства.

1 Типы отображения элементов. ***CSS***-свойство ***display*** задаёт тип отображения элемента. На русский язык его можно перевести фразой: «Веди себя как …». То есть берём любой элемент ***html*** и говорим ему: веди себя как … блок, строка, таблица или вообще, как будто тебя нет.

Наиболее часто используемые значения свойства ***display***:

– ***none***;
– ***block***;
– ***inline***;
– ***inline-block***;
– ***list-item***.

Свойство ***display*** изменяет поведение элемента, но не классовую принадлежность.

2 Замещаемые и незамещаемые элементы. ***CSS*** определяется элементами, но не все элементы создаются одинаково. Например, изображения и абзацы – это элементы разных типов, так же как <***span***> и <***div***>. В CSS элементы разделяются на замещаемые и незамещаемые.

Замещаемыми (***replaced***) называются те элементы, содержимое которых замещается чем-то, что не содержится непосредственно в документе. Наиболее очевидный пример – элемент img, замещаемый файлом изображения, который является внешним по отношению к документу. Кстати, ***img*** фактически не имеет содержимого, как видно из простого примера:

******

Основная масса элементов HTML – незамещаемые (***nonreplaceable***). Это означает, что их отображаемое агентом пользователя содержимое указано в коде. Например, элемент <***span***>…<***/span***> – незамещаемый элемент, и агент пользователя (***user agent***) будет отображать текст. Это выполняется для абзацев, заголовков, ячеек таблиц, списков и почти всех остальных элементов ***html***.

3 Сворачивание вертикальных полей (***margin***). Есть две ситуации в верстке, т. е. если вертикальные поля сворачиваются, то анализируется:

1) соседство двух элементов с вертикальными полями;

2) наличие вертикальных полей у родительского и дочернего элемента.

Внешне такой эффект выглядит так, будто поля накладываются друг на друга.

4 Строчные и внутристрочные ***inline***-элементы. Строчными называются элементы ***web***-страницы, которые являются непосредственно частью строки. К строчным элементам относятся элементы <***img***>, <***span***>, <***a***>, <***q***>, <***code***> и др. В основном они используются для изменения вида текста.

Характерные особенности строчных элементов:

– внутрь строчных элементов допустимо помещать текст или другие строчные элементы. Вставлять блочные элементы внутрь строчных запрещено;

– эффект схлопывания полей не действует. Если рядом стоят два строчных элемента с определенными полями, то эти поля суммируются;

– свойства, связанные с размерами ***width***, ***height***, не применимы, их просто нет;

– ширина элемента равна содержимому, плюс значения отступов, полей и границ;

– несколько строчных элементов, идущих подряд, располагаются на одной строке друг за другом и переносятся на следующую строку при необходимости;

– строчные элементы можно выравнивать по вертикали при помощи ***CSS***-свойства ***vertical-align***.

Порядок выполнения работы

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить отчет в виде ***html***-файла.

Контрольные вопросы

1 Какие существуют концепции визуального форматирования?
2 В чем отличие замещаемого элемента от незамещаемого?
3 Какие есть типы отображения элементов?
4 Что нужно сделать, чтобы не происходило схлопывание полей?
5 Как сделать элемент невидимым?
6 Можно ли в строчные элементы вкладывать блочные, и наоборот?
7 Как создается слой в html-коде?
8 Как задается абсолютное позиционирование?
9 Дайте определение понятия селектора класса.
10 Для чего предназначается встраиваемый стиль?

8 Лабораторная работа № 8. CSS. Позиционирование и свободное перемещение

Цель работы: изучить возможности основных типов позиционирования в ***CSS*** при создании ***web***-документов.

Сущность позиционирования состоит в том, чтобы любой элемент ***web***-страницы расположить в необходимом месте.

Для верстки страниц используются два основных инструмента – позиционирование (***positioning***) и свободное перемещение (***floating***). CSS-позиционирование позволяет указать, где появится блок элемента, а свободное перемещение перемещает элементы к левому или правому краю блока-контейнера, позволяя остальному содержимому «обтекать» его.

Известны следующие виды позиционирования: позиционирование в ***CSS***, абсолютное позиционирование, фиксированное позиционирование, относительное позиционирование, липкое позиционирование, позиционирование с плавающими элементами и позиционирование с использованием свойства *z*-индекс.

1 Позиционирование в ***CSS*** определяется свойством ***position***, которое может иметь следующие значения:

position: static;
position: absolute;
position: fixed;
position: relative.

Указание точного месторасположения элементов осуществляется свойствами смещения: ***top***, ***right***, ***bottom***, ***left***:

top – расстояние от верхнего края окна блок-контейнера или браузера;
bottom – расстояние от нижнего края окна блок-контейнера или браузера;
left – расстояние от левого края окна блок-контейнера или браузера;
right – расстояние от правого края окна блок-контейнера или браузера.

Свойства смещения работают со всеми позиционированными элементами, кроме имеющих значение ***static***.

2 Абсолютное позиционирование задается при установке свойства элемента ***position***:***absolute***.

При абсолютном позиционировании элемент может выводиться из общего потока, становиться независимым от других элементов и может накладываться на другие элементы.

При абсолютном позиционировании положение элемента задаётся только свойствами его смещения ***bottom***, ***left***, ***right***, ***top*** относительно блок-контейнера.

Блок-контейнером считается любой ближайший предок, значение свойства ***position*** которого отлично от ***static***. Если таких предков нет, то блок-контейнером считается начальный блок-контейнер.

Известны следующие важные моменты абсолютного позиционирования:

– абсолютно позиционированные элементы перемещаются вместе с документом при прокрутке;

– свойства ***left*** и ***top*** имеют приоритет выше, чем свойства ***right*** и ***bottom***;

– элементы можно спрятать. Для этого их свойство можно задать отрицательным, или значением ***left***, либо ***top***. При этом элемент выйдет за границы окна браузера, а полоса прокрутки при этом не возникнет;

– если задать свойство ***left*** больше, чем ширина окна браузера, либо отрицательное значение свойства ***right***, то возникнет горизонтальная полоса прокрутки. Аналогично происходит и с использованием свойства ***top***, но полоса прокрутки при этом будет вертикальной.

3 Фиксированное позиционирование задается свойством ***position***:***fixed*** и по своей сути очень похоже на абсолютное позиционирование. Элемент с фиксированным позиционированием также выводится из общего потока и не зависит от расположения других элементов. Положение элемента не изменяется при прокрутке. Элемент так же, как и при абсолютном позиционировании, может накладываться на другие. Еще одной особенностью фиксированного позиционирования является то, что при выходе содержимого за пределы видимой области не возникает полос прокрутки.

Фиксированное позиционирование используется для создания меню, вкладок, заголовков, т. е. таких элементов, которые всегда должны быть видны пользователю.

4 Относительное позиционирование задается свойством ***position***:***relative***. При таком способе позиционирования положение элемента определяется относительно краёв элемента родителя и сам элемент не выводится из основного потока. Расстояние до краёв родительского элемента задаётся свойствами: ***bottom***, ***left***, ***right***, ***top***.

Важными моментами относительного позиционирования являются:

– данный тип позиционирования не применяется к элементам таблицы;

– если при смещении элемента образовалось пустое место, оно не занимается ниже- или вышележащими элементами;

– относительно позиционированный элемент образует блок-контейнер для других элементов.

5 Липкое позиционирование. Элемент остается в нормальном потоке документа до тех пор, пока не пересечет определенное пороговое положение, после чего он рассматривается как фиксированный позиционируемый элемент в своем блок-контейнере. Как только элемент пересекает пороговое положение в обратном направлении, он возвращается в исходное место в нормальном потоке документа.

6 Позиционирование с плавающими элементами. Плавающее поведение элемента выполняется заданием свойства ***float***. Оно применятся для создания красивого эффекта обтекания содержимым какого-либо элемента.

Возможные значения свойства ***float***:

– ***none*** – без обтекания;

– ***left*** – выравнивание по левому краю и обтекание по правому краю;

– ***right*** – выравнивание по правому краю и обтекание по левому краю.

7 Свойство ***z***-индекс предназначено для работы с элементами страницы в трехмерном пространстве. Для этого вводится третья ось – ось ***Z***. Так как страницу мы видим как двухмерную, то элементы накладываются друг на друга в порядке слой за слоем. Управлять подобным наложением можно с помощью свойства ***z-index***.

Возможные значения свойства ***z-index***:

– ***auto***-элементы накладываются друг на друга в том порядке, в каком они были указаны в коде ***html***;

– целое число – чем больше число, тем более высокую позицию элемент занимает по оси ***Z*** и перекрывает все элементы, расположенные ниже по этой оси.

Порядок выполнения работы

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить отчет в виде набора файлов.

Контрольные вопросы

1 В чем сущность позиционирования? Каким свойством оно определяется?

2 Какие свойства являются свойствами смещения?

3 Какие базовые схемы размещения элементов есть в ***CSS***?

4 Что такое нормальный поток?

5 В чем отличие абсолютного позиционирования от относительного?

6 Каким свойством задаются плавающие элементы?

7 Когда можно воспользоваться внутренними стилями?

8 В чем состоит суть правила наследования?

9 Дайте определение понятия селектора класса.

10 Для чего предназначается встраиваемый стиль?

9 Лабораторная работа № 9. Основы Flexbox

Цель работы: изучить основы модели ***flexbox*** для разработки ***web***-документов.

Flexbox – это общее название для модуля ***Flexible Box Layout***, который входит в ***CSS3***. Данный модуль определяет особый режим компоновки/верстки пользовательского интерфейса, который называется ***flex layout*** (гибкий макет). В этом плане ***Flexbox*** предоставляет иной подход к созданию пользовательского интерфейса, который отличается от табличной или блочной верстки.

Главной характеристикой ***flex***-контейнера является способность менять ширину или высоту дочерних элементов, чтобы наиболее оптимально заполнить доступное пространство при разных размерах экрана.

Основными составляющими компоновки ***flexbox*** являются: ***flex***-контейнер (***flex container***) и ***flex***-элементы (***flex items***).

Flex-контейнер – это родительский элемент, в котором содержатся ***flex***-элементы. ***Flex***-контейнер определяется установкой свойства ***display*** в значение ***flex*** или ***inline-flex***.

Под ***Flex***-элементом понимается дочерний элемент ***flex***-контейнера. Текст, который содержится в ***flex***-контейнере, оборачивается анонимным ***flex***-элементом.

Ось ***main axis*** – это условная центральная ось во ***flex***-контейнере, вдоль которой позиционируются ***flex***-элементы.

Main start и ***main end*** – описывают начало и конец центральной оси, а расстояние между ними обозначается как ***main size***.

Cross axis – это поперечная ось, перпендикулярная основной оси.

Cross start и ***cross end*** – это линии, которые определяют начало и конец поперечной оси, относительно которых выкладываются ***flex***-элементы. Расстояние между этими линиями описывается термином ***cross size***.

Порядок выполнения работы

1 Ознакомиться с основными теоретическими положениями.

2 Пройти учебную игру и результаты представить преподавателю при защите работы.

3 Выполнить задание, полученное у преподавателя.

4 Оформить отчет в виде набора файлов.

Контрольные вопросы

1 Какую задачу решает технология ***flex-box***?

2 Что такое ***flex***-контейнер?

3 Какие значения свойства flex-wrap Вы знаете?

4 Что делает свойство ***flex-flow***?

5 Назовите оси ***flex***-контейнера?

6 Как создать ***flex***-контейнер?

7 Как создать ***flex***-элемент?

8 Что делает свойство ***flex-basis***?

9 Какое свойство задаёт направление основных осей в контейнере?

10 Как разрешить перенос ***flex***-элементов внутри ***flex***-контейнера?

Список литературы

1 **Бройдо, В. Л.** Вычислительные системы, сети и телекоммуникации: учебник / В. Л. Бройдо. – 4-е изд. – Санкт-Петербург: Питер, 2013. – 567 с. : ил.

2 **Гуриков, С. Р.** Интернет-технологии: учебное пособие / С. Р. Гуриков. – Москва: ФОРУМ; ИНФРА-М, 2017. – 184 с.

3 **Фрейн, Б.** HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – Санкт-Петербург: Питер, 2014. – 304 с: ил.

4 **Гоше, Х. Д.** HTML5. Для профессионалов: пер. с англ. / Х. Д. Гоше. – 2-е изд. – Санкт-Петербург: Питер, 2015. – 560 с.: ил.

5 **Дронов, В. А.** HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2011. – 414 с.

6 **Дакетт, Д.** HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – 2-е изд. – Санкт-Петербург: Эксмо, 2017. – 923 с.

7 **Дунаев, В. В.** HTML, скрипты и стили / В. В. Дунаев. – Санкт-Петербург: БХВ-Петербург, 2011. – 810 с.

8 **Евсеев, Д. А.** Web-дизайн в примерах и задачах: учебное пособие / Д. А. Евсеев, В. В. Трофимов; под ред. В. В. Трофимова. – Москва: КНОРУС, 2010. – 272 с.

9 **Макфарланд, Д.** Большая книга CSS / Д. Макфарланд. – 2-е изд. – Санкт-Петербург: Питер, 2012. – 560 с.: ил.

10 **Мак-Дональд, М.** HTML5. Недостающее руководство пер. с англ. / М. Мак-Дональд. – Санкт-Петербург: БХВ-Петербург, 2012. – 480 с. : ил.

11 **Матросов, А. В.** HTML 4.0 / А. В. Матросов, А. О. Сергеев, М. П. Чаунин. – Санкт-Петербург: БХВ-Петербург, 2007. – 672 с.: ил.

12 **Мейер, Э.** CSS-каскадные таблицы стилей: пер. с англ. / Э. Мейер. – 3-е изд. – Санкт-Петербург: Символ-Плюс, 2010. – 576 с.: ил.

13 **Комолова, Н.** HTML, XHTML и CSS / Н. Комолова, Е. Яковлева. – Санкт-Петербург: Питер, 2012. – 304 с.: ил.