Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 10.02.2023 09:21:16
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«_25_» _02_____2022 г.

Электронный бизнес

Методические указания к лабораторным работам
для студентов направления подготовки 09.03.02

Курск 2022

УДК 004

Составители: А.В. Киселев

Рецензент
Кандидат технических наук, доцент *Ю.А. Халин*

Электронный бизнес: методические указания к лабораторным работам для студентов направлений подготовки 09.03.02 / Юго-Зап. гос. ун-т; сост.; А.В. Киселев. – Курск, 2022. - 59 с.: - ил. 13, табл. 2. – Библиогр.: с. 59

Содержат основные сведения об основах электронной коммерции, ее безопасности и электронного маркетинга.

Предназначены для студентов направления подготовки 09.03.02 очной формы обучения.

Методические указания соответствуют рабочей программе дисциплины «Электронный бизнес».

Текст печатается в авторской редакции

Подписано в печать . Формат 60*84 1/16.
Усл. печ. л.___. Уч.-изд. л. 3,1. Тираж 50 экз. Заказ 895. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1 Использование SHA-хэширования для защиты конфиденциальных данных от несанкционированных действий штатными средствами 1С Предприятие 8.0.

Цель работы научиться работать с контрольной (хеш) суммой. В лабораторной работе рассмотрен пример реализации шифрования и расшифровки произвольных текстовых полей справочников с проверкой зашифрованных данных с помощью контрольной (хеш) суммы.

Порядок работы

1. Должен быть реализован функционал шифрования и расшифровки по паролю, заданному пользователем.
2. Зашифрованные данные должны храниться в полях исходных данных, т.е. длина зашифрованного текста должна быть равной длине исходного текста, дополнительных таблиц быть не должно.
3. Список полей для шифрования и расшифровки должен быть настраиваемым и не зависеть от конфигурации 1С.
4. Шифрование должно выполняться средствами платформы 1С без внешних компонент, com-объектов и т.д. (для исключения зависимости от операционной системы).
5. При шифровании должна вычисляться контрольная хеш-сумма на основании пользовательского пароля одним из четырех алгоритмов (на выбор пользователя). Данная хеш-сумма должна сохраняться в хранилище общих настроек конфигурации.
6. При расшифровке указанный пользователем паролем должен проверяться на равенство с ранее сохранённой хеш-суммой и, в случае несовпадения, не позволять пользователю расшифровать данные (чтобы не испортить их).

Хэш-функции – это функции, предназначенные для «сжатия» произвольного сообщения или набора данных, записанных, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной

длины, называемую сверткой. Хэш-функции имеют разнообразные применения при проведении статистических экспериментов, при тестировании логических устройств, при построении алгоритмов быстрого поиска и проверки целостности записей в базах данных. Основным требованием к хэш-функциям является равномерность распределения их значений при случайном выборе значений аргумента.

Криптографической хеш-функцией называется всякая хеш-функция, являющаяся криптостойкой, то есть удовлетворяющая ряду требований специфичных для криптографических приложений. В криптографии хэш-функции применяются для решения следующих задач:

— построения систем контроля целостности данных при их передаче или хранении,

— аутентификация источника данных.

Хэш-функцией называется всякая функция h:X -> Y, легко вычислимая и такая, что для любого сообщения M значение h(M) = H (свертка) имеет фиксированную битовую длину. X — множество всех сообщений, Y — множество двоичных векторов фиксированной длины.

Как правило хэш-функции строят на основе так называемых одношаговых сжимающих функций y = f(x1, x2) двух переменных, где x1, x2 и y — двоичные векторы длины m, n и n соответственно, причем n — длина свертки, а m — длина блока сообщения.

Для получения значения h(M) сообщение сначала разбивается на блоки длины m (при этом, если длина сообщения не кратна m то последний блок неким специальным образом дополняется до полного), а затем к полученным блокам M1, M2,.., MN применяют следующую последовательную процедуру вычисления свертки:

Ho = v,

Hi = f(Mi,Hi-1), i = 1,.., N,

h(M) = HN

Здесь v — некоторая константа, часто ее называют инициализирующим вектором. Она выбирается из различных соображений и может представлять собой секретную константу или набор случайных данных (выборку даты и времени, например).

При таком подходе свойства хэш-функции полностью определяются свойствами одношаговой сжимающей функции.

Выделяют два важных вида криптографических хэш-функций — ключевые и бесключевые. Ключевые хэш-функции называют кодами аутентификации сообщений. Они дают возможность без дополнительных средств гарантировать как правильность источника данных, так и целостность данных в системах с доверяющими друг другу пользователями.

Бесключевые хэш-функции называются кодами обнаружения ошибок. Они дают возможность с помощью дополнительных средств (шифрования, например) гарантировать целостность данных. Эти хэш-функции могут применяться в системах как с доверяющими, так и не доверяющими друг другу пользователями.

О статистических свойствах и требованиях

Основным требованием к хэш-функциям является равномерность распределения их значений при случайном выборе значений аргумента. Для криптографических хеш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось. Это называется лавинным эффектом.

К ключевым функциям хэширования предъявляются следующие требования:

— невозможность фабрикации,

— невозможность модификации.

Первое требование означает высокую сложность подбора сообщения с правильным значением свертки. Второе — высокую сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки.

К бесключевым функциям предъявляют требования:

— однонаправленность,

— устойчивость к коллизиям,

— устойчивость к нахождению второго прообраза.

Под однонаправленностью понимают высокую сложность нахождения сообщения по заданному значению свертки. Следует заметить что на данный момент нет используемых хэш-функций с доказанной однонаправленностью.

Под устойчивостью к коллизиям понимают сложность нахождения пары сообщений с одинаковыми значениями свертки. Обычно именно нахождение способа построения коллизий криптоаналитиками служит первым сигналом устаревания алгоритма и необходимости его скорой замены.

Под устойчивостью к нахождению второго прообраза понимают сложность нахождения второго сообщения с тем же значением свертки для заданного сообщения с известным значением свертки.

Алгоритмы CRC16/32 — контрольная сумма (не криптографическое преобразование).

Алгоритмы MD2/4/5/6. Являются творением Рона Райвеста, одного из авторов алгоритма RSA.

Алгоритм MD5 имел некогда большую популярность, но первые предпосылки взлома появились еще в конце девяностых, и сейчас его популярность стремительно падает.

Алгоритм MD6 — очень интересный с конструктивной точки зрения алгоритм. Он выдвигался на конкурс SHA-3, но, к сожалению, авторы не

успели довести его до кондиции, и в списке кандидатов, прошедших во второй раунд этот алгоритм отсутствует.

Алгоритмы линейки SHA Широко распространенные сейчас алгоритмы. Идет активный переход от SHA-1 к стандартам версии SHA-2. SHA-2 — собирательное название алгоритмов SHA224, SHA256, SHA384 и SHA512. SHA224 и SHA384 являются по сути аналогами SHA256 и SHA512 соответственно, только после расчета свертки часть информации в ней отбрасывается. Использовать их стоит лишь для обеспечения совместимости с оборудованием старых моделей.

Российский стандарт — ГОСТ 34.11-94.

Обзор алгоритмов формирования хеш-функций

В настоящее время предложены и практически используются различные специальные алгоритмы для вычисления хеш-функции. Наиболее известными алгоритмами являются MD5, SHA-1, SHA-2 и другие версии SHA, а также отечественный алгоритм, изложенный в ГОСТ Р 34.11-94.

Алгоритм MD5 появился в начале 90-х годов XX века в результате усовершенствования алгоритма формирования хеш-функции MD4. Символы в названии "MD" означают Message Digest – краткое изложение сообщения. Автор алгоритмов MD4 и MD5 – Р. Ривест (R.Rivest). В результате использования MD5 для произвольного сообщения формируется 128-битное хеш-значение. Входные данные обрабатываются блоками по 512 бит. В алгоритме используются элементарные логические операции (инверсия, конъюнкция, сложение по модулю 2, циклические сдвиги и др.), а также обыкновенное арифметическое сложение. Комплексное повторение этих элементарных функций алгоритма обеспечивает то, что результат после обработки хорошо перемешан. Поэтому маловероятно, чтобы два сообщения, выбранные случайно, имели

одинаковый хеш-код. Алгоритм MD5 имеет следующее свойство: каждый бит полученного хеш-значения является функцией от каждого бита входа. Считается, что MD5 является наиболее сильной хеш-функцией для 128-битного хеш-значения.

Алгоритм SHA (Secure Hash Algorithm – Безопасный хеш-алгоритм) был разработан национальным институтом стандартов и технологии (NIST) США и опубликован в качестве американского федерального информационного стандарта в 1993 году. SHA-1, как и MD5, основан на алгоритме MD4. SHA-1 формирует 160-битное хеш-значение на основе обработки исходного сообщения блоками по 512 бит. В алгоритме SHA-1 также используются простые логические и арифметические операции. Наиболее важным отличием SHA-1 от MD5 является то, что хеш-код SHA-1 на 32 бита длиннее, чем хеш-код MD5. Если предположить, что оба алгоритма одинаковы по сложности для криптоанализа, то SHA-1 является более стойким алгоритмом. Используя атаку методом грубой силы (лобовую атаку), труднее создать произвольное сообщение, имеющее данный хеш-код, а также труднее создать два сообщения, имеющие одинаковый хеш-код.

В 2001 году национальный институт стандартов и технологии США принял в качестве стандарта три хеш-функции с большей длиной хеш-кода, чем у SHA-1. Часто эти хеш-функции называют SHA-2 или SHA-256, SHA-384 и SHA-512 (в названии указывается длина создаваемого алгоритмами хеш-кода). Эти алгоритмы отличаются не только длиной создаваемого хеш-кода, но и используемыми внутренними функциями и длиной обрабатываемого блока (у SHA-256 длина блока – 512, а у SHA-384 и SHA-512 длина блока – 1024 бита). Постепенные усовершенствования алгоритма SHA ведут к увеличению его криптостойкости. Несмотря на отличия рассматриваемых алгоритмов друг от друга, все они являются дальнейшим развитием SHA-1 и MD4 и имеют похожую структуру.

В России принят ГОСТ Р34.11-94, который является отечественным стандартом для хеш-функций. Его структура довольно сильно отличается от структуры алгоритмов SHA-1,2 или MD5, в основе которых лежит алгоритм MD4. Длина хеш-кода, создаваемого алгоритмом ГОСТ Р 34.11-94, равна 256 битам. Алгоритм последовательно обрабатывает исходное сообщение блоками по 256 бит справа налево. Параметром алгоритма является стартовый вектор хеширования – произвольное фиксированное значение длиной также 256 бит. В алгоритме ГОСТ Р 34.11-94 используются операции перестановки, сдвига, арифметического сложения, сложения по модулю 2. В качестве вспомогательной функции в ГОСТ 34.11-94 используется алгоритм по ГОСТ 28147-89 в режиме простой замены.

Ключевые термины

Hash function – хеш-функция.

ГОСТ Р34.11-94 – российский стандарт на функцию хеширования.

Хеш-функция – математическая или иная функция, которая для строки произвольной длины вычисляет некоторое целое значение или некоторую другую строку фиксированной длины.

Хеш-код – результат работы хеш-функции, некоторый характерный "признак" входного массива данных.

Хеш-функция – математическая или иная функция, которая для строки произвольной длины (прообраза) вычисляет некоторое целое значение или некоторую другую строку фиксированной длины. Смысл хеш-функции состоит в определении характерного признака прообраза – значения хеш-функции. Это значение обычно имеет определенный фиксированный размер. Хеш-код может быть в дальнейшем проанализирован для решения какой-либо задачи.

Хеш-функции широко применяются в современной криптографии. В криптографии хеш-функция считается хорошей, если трудно создать два прообраза с одинаковым значением хеш-функции, а также, если у выхода функции нет явной зависимости от входа.

В качестве хеш-функции можно использовать блочный алгоритм симметричного шифрования в определенных режимах. Кроме того, в настоящее время предложены и практически используются различные специальные алгоритмы для вычисления хеш-функции. Наиболее известными алгоритмами являются MD5, SHA-1, SHA-2 и другие версии SHA, а также отечественный алгоритм, изложенный в ГОСТ Р 34.11-94.

```
#Область Лабораторная_работа_2

&НаСервере
Функция ВычислитьХешСумму(ТекПароль)

  Если ЗначениеЗаполнено(ТипХешФункции) Тогда
    АлгоритмХешФункции = ХешФункция[ТипХешФункции];
  Иначе
        //по умолчанию такой
        АлгоритмХешФункции = ХешФункция.MD5;
  КонецЕсли;
```

```
        Хеш = Новый ХешированиеДанных(АлгоритмХешФункции);
        Хеш.Добавить(ТекПароль);

        Возврат Хеш.ХешСумма;

    КонецФункции

    &НаСервере
    Функция КонтрольнаяХешСуммаПравильная(ХешСуммаИхХранилища
,ТекХешСумма)
        Возврат (ХешСуммаИхХранилища = ТекХешСумма);
    КонецФункции

    &НаСервере
    Процедура СохранитьХешСуммуВХранилищеЗначенияНаСервере()

        //сохранить хеш сумму
        КлючНастроек = "ХранилищеКонтрольнойХешСуммы";
        Настройки = Новый Соответствие;
        Настройки.Вставить("КонтрольнаяХешСумма", Объект.Контрольная
ХешСумма);

        ХранилищеОбщихНастроек.Сохранить("ХранилищеКонтрольнойХе
шСуммы", КлючНастроек, Настройки);
```

```
КонецПроцедуры

&НаСервере
Функция ВосстановитьХешСуммуИзХранилищаЗначенияНаСервере()

  //восстановить хеш сумму
  КлючНастроек = "ХранилищеКонтрольнойХешСуммы";
  ЗначениеНастроек = ХранилищеОбщихНастроек.Загрузить("ХранилищеКонтрольнойХешСуммы", КлючНастроек);
  Если ТипЗнч(ЗначениеНастроек) = Тип("Соответствие") Тогда
      Возврат ЗначениеНастроек.Получить("КонтрольнаяХешСумма");
  КонецЕсли;

КонецФункции

&НаКлиенте
Процедура ВосстановитьХешСуммуИзХранилищаЗначения(Команда)
  ХешСуммаИзХранилища = ВосстановитьХешСуммуИзХранилищаЗначенияНаСервере();
КонецПроцедуры

&НаКлиенте
```

```
Процедура СформироватьХешФункцию(Команда)

  ХешСуммаИзХранилища = ВычислитьХешСумму(Объект.Пароль);

КонецПроцедуры


#КонецОбласти
```

Контрольные вопросы

1. Что в криптографии называется хеш-функцией?
2. Для каких целей используются хеш-функции?
3. Перечислите основные требования, предъявляемые к хеш-функциям.
4. Назовите примеры криптографических хеш-функций.
5. Каков российский стандарт на алгоритм формирования криптографической хеш-функции?
6. Каким образом можно использовать блочный алгоритм шифрования для формирования хеш-функции?

Упражнения для самопроверки

Пусть хеш-функция $y=h(x_1x_2\ldots x_n)$ определяется как результат выполнения побитовой операции "сумма по модулю 2" для всех байтов сообщения, представленного в двоичном виде. Длина хеш-кода равна 8 битам. Для каждого из шести сообщений, записанных в левом столбце, найдите соответствующий результат вычисления хеш-функции из правого столбца. Все сообщения и значения хеш-функции представлены в шестнадцатеричном формате.

Сообщения	Значения хеш-функции
0A3 69 2C	38
82 0F B5	1B
0DA 14 90	0F9
32 01 BF	8C
9E A6 23	0E6
10 BE 57	5E

Лабораторная работа №2 Разработка простейшего умного контракта на платформе Ethereum.

Цель работы: изучить и закрепить на практике возможности основных инструментовразработчика смарт-контрактов.

Смарт-контракты: что это?

Смарт-контракты, или "умные контракты", позволяют передавать некоторые ценности, например, собственность или акции, прозрачным и одновременно безопасным способом, что делает весь процесс сверхэффективным, одновременно устраняя промежуточные звенья, зачастую долгие идорогие. Рассмотрим пример, который позволит понять, как блокчейн работает со смарт- контрактами.

Давайте представим, что есть два заинтересованных лица в сделке с недвижимостью. Один (*продавец*) желает продать жилье, а другой (*покупатель*) хочет купить это жилье. *Сделка* по продаже может быть реализована посредством блокчейна, и *покупатель* готов платить, например, биткоинами. Как только *покупатель* заплатит, то сразу получит подтверждение о транзакции, которое будет исполнено в виде виртуального смарт-контракта. *Продавец*, в свою *очередь*, передает покупателю цифровой *ключ* от *входной* двери, который будет доставлен в день, о котором заинтересованные стороны договорились. Если *продавец* вдруг передумает продавать дом, *покупатель* не получит *ключ*, блокчейн в этом случае автоматически вернет покупателю деньги в тотдень, когда должен был быть получен *ключ*. А если *покупатель* получит *ключ* заранее, то блокчейн его удержит до дня, в который была договоренность осуществления передачи. Поэтому каждая из сторон получит то, что хочет, в оговоренный в контракте день: *продавец* - деньги, а *покупатель* - *ключ*. А поскольку блокчейн - это технология, основанная на пиринговой сети, договор по этой сделке будет храниться на множестве узлов, что обеспечит выполнение взятых по контракту обязательств, и ни одна из сторон не сможет изменить условия

контракта после его заключения. Ну а если кто-то из сторон наберется смелости сделать это, все узлы в сети тут же об этом узнают, и проблема будет мгновенно решена.

Мы рассмотрели пример с куплей-продажей недвижимости. Но такие же соглашения могут заключаться при передаче акций, в страховании автомобилей или другого имущества и во многих других случаях. Позвольте привести несколько ключевых преимуществ смарт-контрактов.

Первое качество, за которое смарт-контракты так ценятся, это *автономность*. Смарт-контракты не могут быть изменены третьими лицами, так как только их стороны заключают соглашение. Нет необходимости обращаться к услугам юристов при заключении соглашений.

Второе преимущество, за которое люди любят - или еще полюбят - смарт-контракты, это *доверие* к ним. Смарт-контракт невозможно потерять. Они все зашифрованы и хранятся в общественном хранилище. Поэтому потеря любого из них исключена.

Это подводит к следующему плюсу - *резервированию*. Можно положиться на *надежность* смарт- контрактов, потому что они все зарезервированы. *Аннулирование* договора по причине потери его копии просто невозможно.

Следующим в списке идет *безопасность*, которая опять же связана с предыдущими двумя. Ваши смарт-контракты будут защищены современными методами шифрования данных. Это отсылает нас к вопросу доверия - вы можете полностью доверять безопасности методов шифрования. Смарт- контракт практически невозможно взломать.

Пятая причина превосходства смарт-контрактов над обычными - это *скорость* их передачи. На заключение традиционных договоров уходит уйма времени, поскольку в их эту работу вовлечено множество третьих лиц. Если

речь идет о распространении кода, смарт-контракты на высоте, поскольку позволяют решать задачи в разы быстрее.

Шестая причина - это *экономия денег* на заключении договоров. Нет необходимости прибегать к услугам адвокатов. Можно просто использовать технологию смарт-контрактов.

И, наконец, огромным преимуществом является *точность*. Если все подробности контракта указаны точно, то он будет выполнен значительно точнее, чем любой другой контракт.

Инструментарий и приложения экосистемы эфириума

Прежде чем погрузиться в написание кода, стоит изучить экосистему Ethereum. Давайте разберемся,какие инструменты и подходы существуют, как они называются и взаимодействуют.

В экосистеме Ethereum широко используются такие инструменты, как Geth, *Parity*, Solidity, Remix,Truffle, Webpack, Angular и так далее. Каждый из них используется для решения конкретных задач.

Узлы сети блокчейна: Go-Ethereum, Parity, CPP-Ethereum

Примерами узлов блокчейна выступают такие программы, как Geth, *Parity* или CPP-Ethereum. Все ониработают на клиентской стороне, то есть их можно загрузить и запустить на вашем компьютере, каки для всех других пользователей сети Ethereum. Они все выполняют одну и ту же задачу: реализуют протокол Ethereum. Несмотря на то, что разные инструменты выполняют одну и ту же роль, они написаны на разных языках программирования. Развитием инструментов занимаются различные команды, которые обязательно следят за тем, чтобы даже на разных языках программирования протокол Ethereum был реализован корректно. Если проводить аналогию, то эта схема похожа на использование среды MySQL в режиме "мульти-мастер", когда все узлы выполняют одну и ту же задачу по репликации *базы данных*. Это отлично описывает то, что делают все узлы в

сети блокчейна - они копируют все блоки на своих компьютерах. Поэтому при загрузке Geth, *Parity*, или CPP-Ethereum и запуске клиента после установки подключения к другим узлам будет загружено все содержимое блокчейна. *Исключение* составляет только режим "легкого клиента", когда загружаются только *заголовки* блоков.

Взаимодействие веб-сайтов и блокчейна

Рассмотрим популярные браузеры MetaMask и Mist. Оба они представляют собой связующее звено между обычным браузером для просмотра интернет-страниц и блокчейном С помощью корректно настроенного веб-сайта можно выполнять программы и отправлять команды в блокчейн. *Пользователь* сможет запустить любой *браузер*: например, *Chrome*, Firefox, *Internet* Explorer или другой *браузер*, зайти на такой *веб-сайт* и взаимодействовать с блокчейном. Для этого к блокчейну необходимо подключиться. MetaMask представляет собой надстройку для *Chrome* и Firefox, облегчающую подключение к блокчейну.

Рисунок 1 – надстройка MetaMask

Mist, в свою *очередь* - это полновесный *браузер*, оснащенный собственным узлом сети блокчейна.

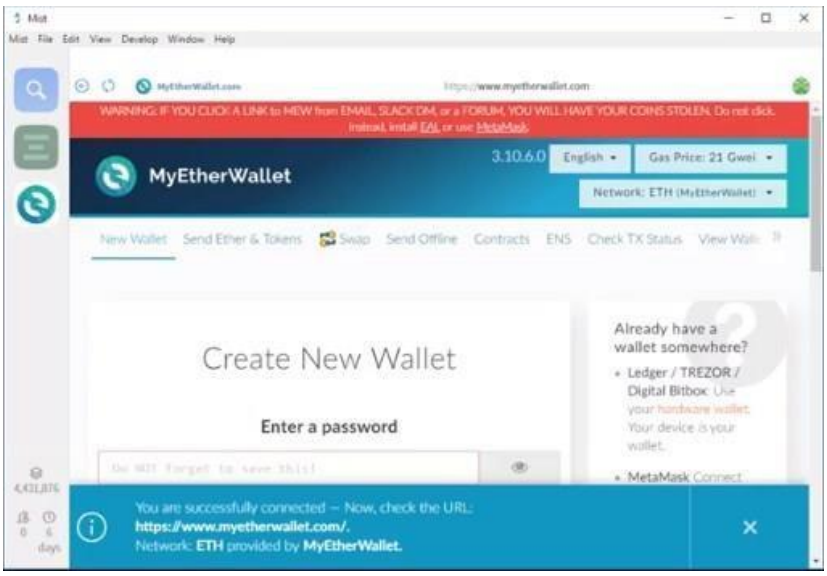


Рисунок 2 – Mist

В случае с Mist, узлом блокчейна является Geth, или Go-Ethereum, непосредственно встроенный в *браузер*. MetaMask для работы использует сервис под названием Infura. В среде Infura используются узлы Geth и *Parity*, которые запущены на стороне сервера, а не на компьютере клиента, а Infura реализует подключение к ним. Чтобы познакомиться с надстройкой MetaMask, можно открыть *браузер*, например, *Chrome*, найти раздел с иконками надстроек, далее открыть MetaMask и можно начинать работать с блокчейном. Кошелек Mist выглядит так: слева *доступ* к различным разделам, есть *отображение* статуса подключения и синхронизации данных, в центре располагается собственно *браузер* Это позволяет работать с блокчейном и просматривать веб-страницы можно одновременно.

Что такое Solidity

Solidity представляет собой *язык программирования* высокого уровня. Для ее работы требуется *компилятор* solc, который формирует байткод для виртуальных машин Ethereum.Встречаются мнения, что Solidity похож на JavaScript. В первых версиях так и было, однако сейчас эти два языка

значительно расходятся. Тем не менее, Solidity похож на JavaScript больше, чем любой другой *язык программирования*.

Remix, веб-среда разработки для Solidity

Remix - это облачная *среда разработки*, поддерживающая много полезных функций. *Доступ* к Remix можно получить по адресу http://remix.ethereum.org. Среда Remix позволяет создавать и запускать код на языке Solidity прямо в окне браузера. Remix оснащена встроенным отладчиком и статическим анализатором кода, а также многими другими инструментами.

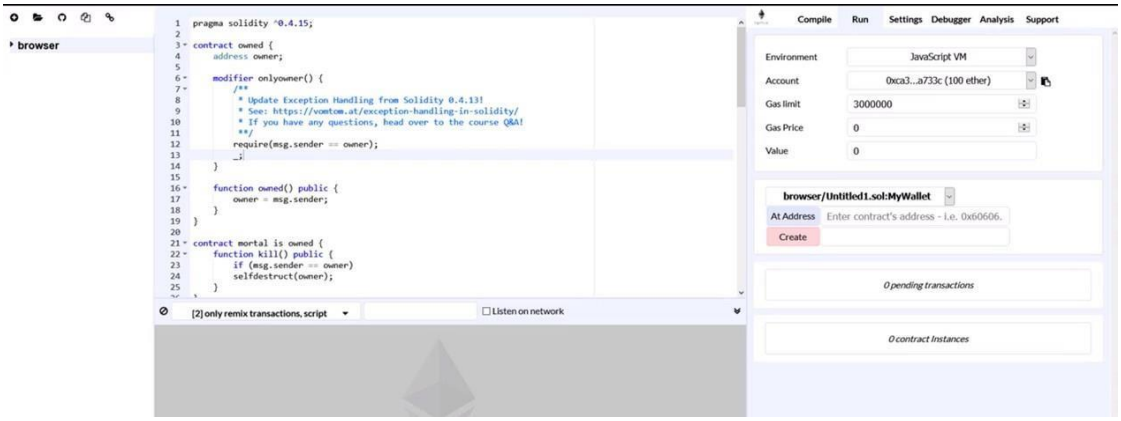


Рисунок 3 – Remix

На текущий момент Remix выглядит так. Слева расположен *браузер*, с помощью которого можно управлять файлами. В центре располагается окно для создания кода, а справа - *управляющие* элементы - вкладки для компиляции (*Compile*), запуска (Run), изменения настроек (Settings), отладки (*Debugger*), анализа (*Analysis*) и получения поддержки (*Support*).На вкладке Run можно выбрать среду запуска кода, например, виртуальную машину *Java*. Remix предоставляет *доступ* к нескольким счетам в эмулированной среде Ethereum для апробирования создаваемого кода. С их помощью можно размещать и обсчитывать контракты, а потом анализировать результаты благодаря наличию журнала исполнения кода.

Использование библиотек Web3.js и Eth.js

Библиотеки Web3.js и Eth.js облегчают взаимодействие между браузером и блокчейном и позволяют работать узлами сети Ethereum по протоколу *RPC* посредством *HTTP* и кода JavaScript. Если запустить локальный узел блокчейна, он откроет *интерфейс HTTP-RPC,* что позволит браузеру отправлять узлу команды, чтобы узел, в свою *очередь,* переправлял данные в блокчейн.

Библиотека Truffle и ее отличие от Web3.js

Truffle и Embark являются инструментариями для среды Solidity и разработки распределенных приложений для работы с блокчейном. Оба они поддерживают управление контрактами, их *размещение* в блокчейне, или миграцию, оснащены встроенной системой тестирования приложений, а Truffle еще и предлагает решение Truffle Boxes - предварительно настроенные среды разработки распределенных приложений, значительно облегчающие работу, такие как Truffle-React, Truffle- Webpack и так далее. При серьезном подходе к разработке приложений для блокчейна стоит уделитьвнимание Truffle и Embark и постепенно отходить от использования только библиотеки Web3.js.

Использование Angular, Vue.js, React и Redux в разработке приложений дляблокчейна

Такие наборы инструментов, как Angular, Vue.js, React, Redux предназначены для разработки веб- страниц и непосредственно не работают с блокчейном, Truffle, Solidity и другими подобными средами. Для работы с Angular, Vue.js, React, Redux или другими инструментариями для создания веб-страниц обычно достаточно загрузить библиотеку Web3, подключиться к узлу блокчейна и настроить взаимодействие с блокчейном с помощью Web3.

Применение инструментов Browserify и Webpack

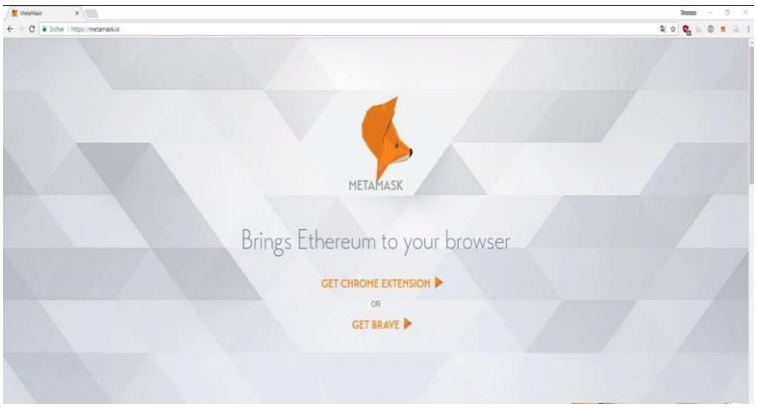
Webpack - это упаковщик файлов JavaScript, необходимо использовать тогда, когда *программа* использует большое количество файлов: Webpack собирает их воедино, разрешает все взаимозависимости между файлами, позволяя коду обращаться только к паре мастер-файлов, что значительно ускоряет загрузку веб-приложения, тк веб-серверу больше не приходится отправлять несколько сотен файлов.

Browserify делает примерно то же самое, но на базовом уровне - ведь Webpack сразу решает спектр задач по упаковке файлов для веб-разработки. Browserify представляет собой только упаковщик, разрешающий файловые взаимозависимости и объединяющий много файлов в один. *Node Package Manager*, или NPM, загружает и управляет пакетами для узлов сети Ethereum, что облегчает разработку веб-проектов.

Обзор и возможности MetaMask

Поговорим о надстройке для браузера *Chrome* - MetaMask. В этом разделе установим ее, разберем функционал, а также узнаем, как получить немного эфира для тестирования MetaMask и размещения контрактор в тестовой сети Rinkeby.

Установка MetaMask



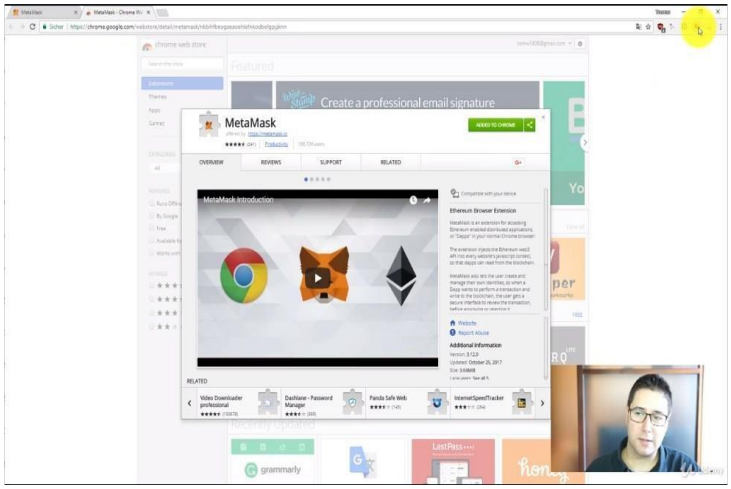


Рисунок 3 – веб-*сайт* MetaMask

При щелчке на *Get the Chrome Extension* (*браузер Chrome* существует для всех платформ) вы перейдете на страницу установки надстройки MetaMask и увидите кнопку *Install*, если MetaMask еще не установлена.

После установки MetaMask можно будет запустить с помощью ее иконки в верхнем правом углу браузера *Chrome*, а несколько простых шагов установки сопровождаются подробными инструкциями.Потребуется задать *пароль* для защиты ваших счетов, после этого можно работать с MetaMask.

Элементы MetaMask

Продолжим разговор о надстройке MetaMask. Для начала выбраем *сеть*, с которой вы будете взаимодействовать - ее можно выбрать в левом верхнем углу окошка MetaMask.

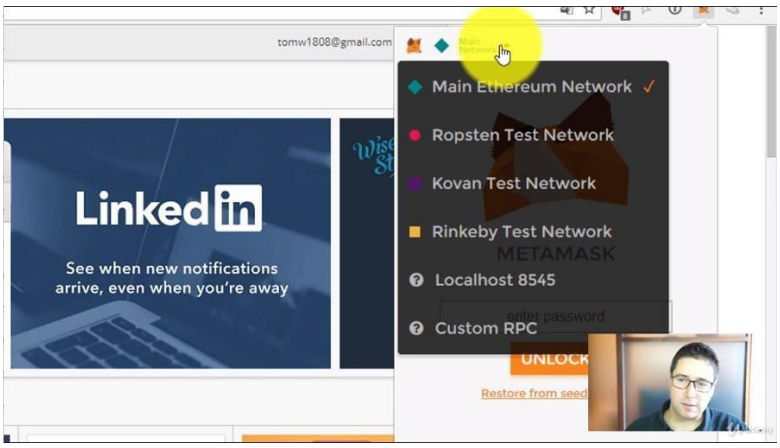


Рисунок 4 – выбор сети в MetaMask

После этого необходимо опубликовать свое *приложение* в главной сети Ethereum, которое будем использовать для взаимодействия с другими смарт-контрактами, опубликованными там.

В рамках данного курса будет достаточно тестовой сети Rinkeby. Мы узнаем, как получить немного эфира для использования в этой тестовой сети, разобраться с тем, как *сеть* реагирует на запросы, посмотрим на майнинг, задержки и проблемы одновременных вычислений, характерные для блокчейна.

Теперь авторизуемся в MetaMask. Используем *пароль* и после входа в систему будет создан счет.

Доступен обзор имеющихся счетов, просмотр совершенных транзакций, а также жетонов среды Ethereum.

Если на счету есть эфир, его можно отправить на другой *адрес* и добавить к транзакции данные. Обэтом мы поговорим позднее. На данном этапе ограничимся просто пересылкой эфира.

В *меню* обзора счета можно скопировать *адрес* в *буфер* обмена и экспортировать частный *ключ* вашего счета для использования его в другом приложении.

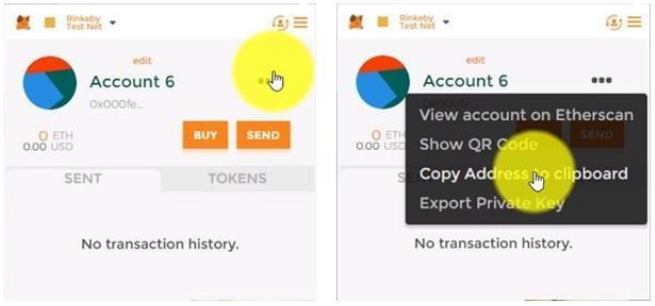


Рисунок 5 – счет в MetaMask

Обратите внимание, что *доступ* к счетам обеспечивается с помощью вашего частного ключа.

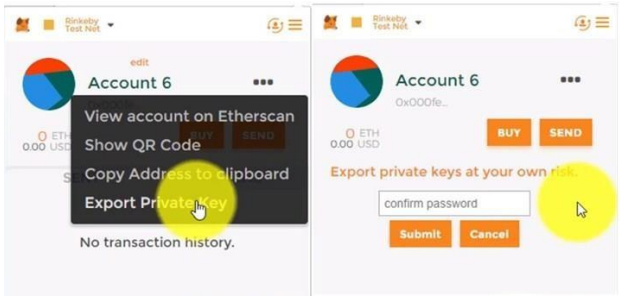


Рисунок 5 – счет в MetaMask

Для переключения между счетами можно использовать опцию в верхнем правом углу.

Можно создавать новые счета с помощью кнопки *Create Account* или импортировать счета с помощью *Import Account*.

Для этого потребуется частный *ключ* к счетам, предварительно созданным в Geth или другойсистеме, создающей ключи *JSON*.

Импорт осуществляется посредством этих файлов-ключей *JSON*.

Как получить эфир для тестовой сети Rinkeby

Чтобы получить некоторое количество эфира для использования в тестовой сети Rinkeby,потребуется открыть веб-*сайт* http://rinkeby.io.

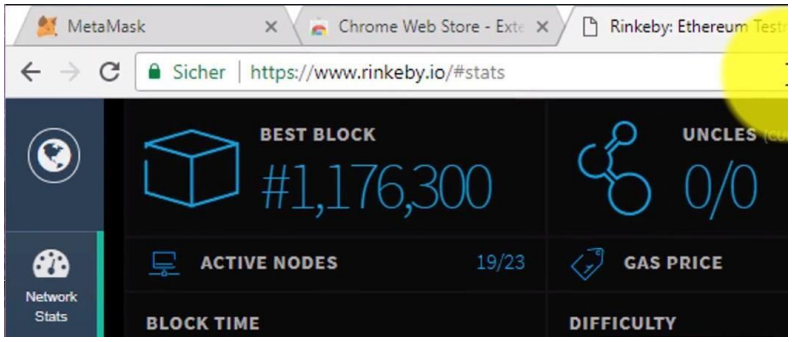


Рисунок 6 – Rinkeby

В его нижнем левом углу расположена иконка Crypto Faucet, при щелчке по которой разъясняется, как можно получить эфир на счет Rinkeby.

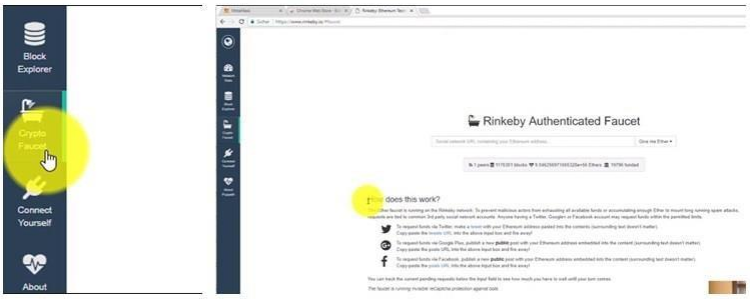


Рисунок 7 – Rinkeby

Для получения эфира достаточно опубликовать номер своего счета в *Twitter*, Google Plus или Facebook, а затем скопировать *адрес* веб-страницы с публикацией в форму на сайте rinkeby.io. Эти меры предосторожности необходимы для защиты от автоматического массового получения эфира.

Теперь зайдем в учетную *запись Twitter*. скопируем *адрес* счета из MetaMask, введем в *поле* для публикации нового твита, допишем комментарий, затем скопируем ссылку на твит.

То же самое можно сделать и с помощью Facebook, единственное, в чем нужно убедиться - этодолжен быть публичный пост. Так, *запрос* на выделение эфира был размещен.

Рисунок 8 – Rinkeby

Для корректной работы системы начисления эфира потребуется отключение блокировщика рекламы. Теперь убедимся, что *запрос* на выделение эфира был удовлетворен; через пару секунд на счету должен появиться эфир.

Наличие на счету эфира можно проверить с помощью сервиса Etherscan.

Пересылка эфира с помощью MetaMask

Теперь попробуем выполнить пересылку эфира между счетами с помощью MetaMask. Допустим, чтона счете номер восемь есть три единицы эфира.

Необходимо переслать часть этих средств на счет номер шесть.

Скопируем *адрес* счета номер шесть, переключимся на счет номер восемь и отправим 0,2 единицыэфира. Щелкнем по *Next*, посмотрим на транзакцию и отправим ее в блокчейн.

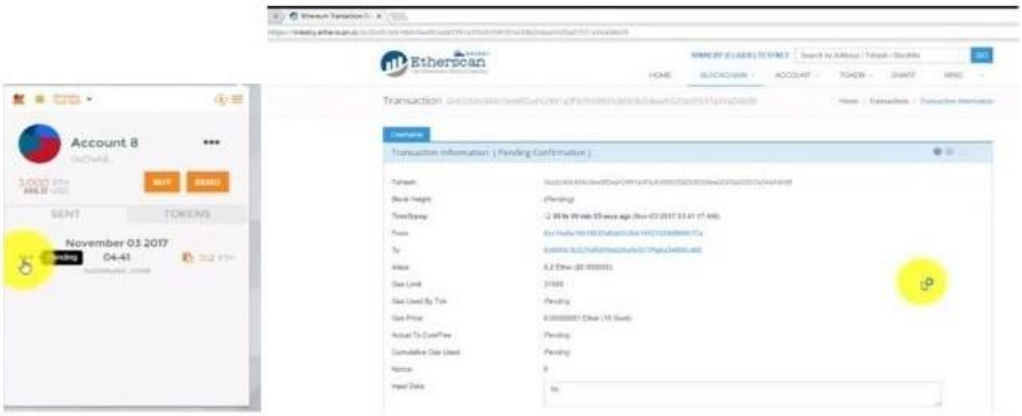


Рисунок 9 – Etherscan

Статус транзакции можно проверить с помощью сервиса Etherscan.

После успешно завершенного обсчета транзакции, средства будут отправлены, и будет отображаться *адрес* блока, в котором эта *транзакция* была впервые подтверждена.

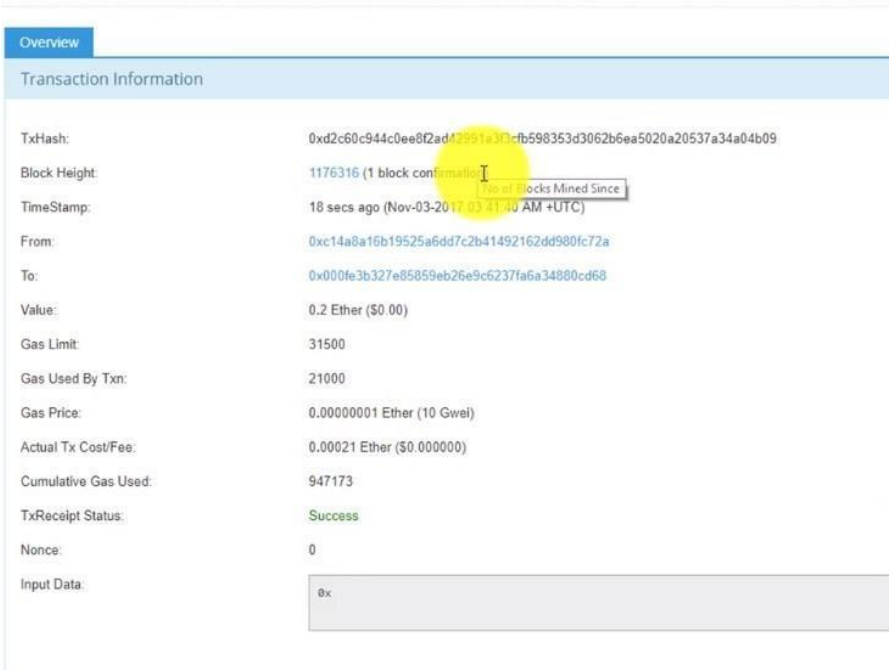


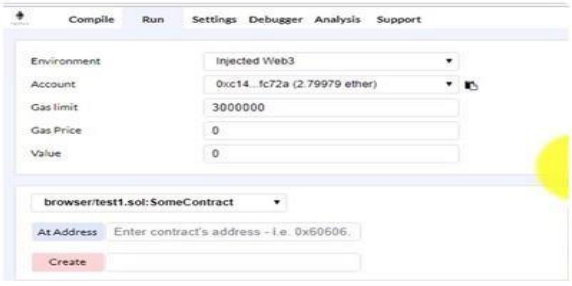
Рисунок 10 – Отображение *адреса* блока

Взаимодействие MetaMask и браузера

Следующей возможностью MetaMask является взаимодействие с блокчейном посредством веб-сайта. В этом случае веб-*сайт* подключается к MetaMask, MetaMask - к блокчейну с помощью сервиса Infura,а Infura, в свою *очередь*, содержит в себе запущенный клиент Geth. Позднее, когда будем разбирать тему подключения к блокчейну из браузера, данная схема будет рассмотрена подробнее - для нее возможны несколько реализаций. На текущем этапе достаточно посмотреть, что происходит в надстройке MetaMask, когда вы пытаетесь взаимодействовать с блокчейном. Откроем среду Remix, здесь есть простой смарт-контракт.

В правой стороне окна Remix видно, что в качестве опорной библиотеки выбрана Web3.

Надстройка MetaMask подключается непосредственно к окну браузера и таким образом обеспечивает*связь* с блокчейном. В раскрывающемся списке выбрана Injected Web3, а в окне разработки, доступном для любой веб-страницы, видно, что с помощью объекта *web3.currentProvider* можно работать с надстройкой MetaMask посредством обычного кода JavaScript.



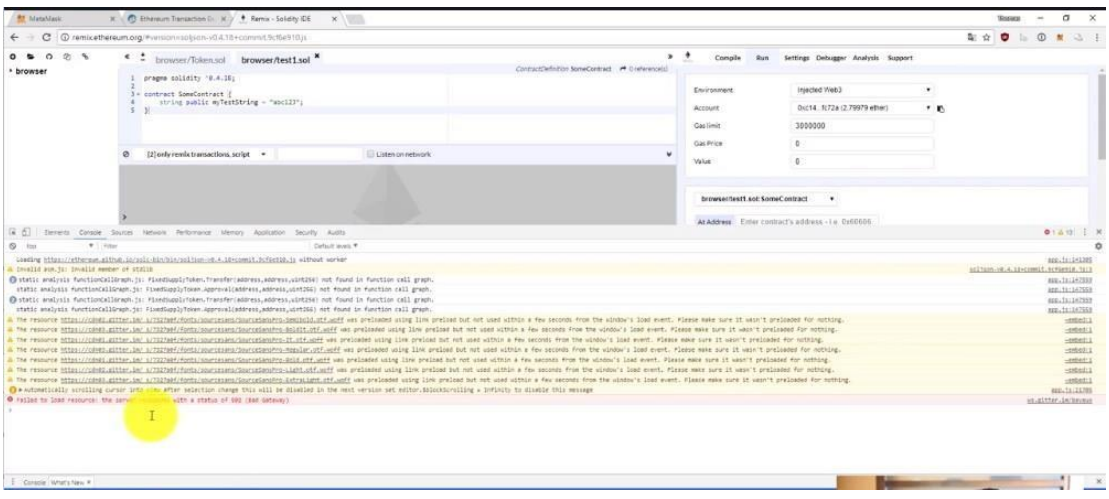


Рисунок 11 – работа с надстройкой MetaMask посредством кода JavaScript

Выберем Injected Web3, затем счет - тот же самый, который открыт в надстройке MetaMask. Теперь можно перейти на вкладку Run и создать контракт.

Отследим всю цепочку: контракт, написанный в среде Remix на языке JavaScript, исполняется, надстройка MetaMask отслеживает эту транзакцию и открывает всплывающее окно для подтверждения создания контракта.

После подтверждения MetaMask отправляет транзакцию в блокчейн.

MetaMask располагает некоторым числом узлов сети, расположенных на серверах разработчиков, которые играют роль посредника между вашим браузером и блокчейном. Если вы хотите создать и использовать свой собственный блокчейн, потребуется один из клиентов: Geth, *Parity*, Mist. Mist обеспечивает интегрированный в *браузер доступ* к блокчейну, а Geth - это работающий на вашем компьютере клиент, загружающий все блоки и предоставляющий к ним непосредственный *доступ*. Успешность завершения транзакции подтверждается обсчитанными блоками.

Основные понятия среды Ethereum

Блокчейн в среде Ethereum очень напоминает блокчейн для Bitcoin. Есть транзакции, есть эфир, подобный биткоину, есть майнеры и так далее. Однако есть и различия. Во-первых, майнеры всееще работают по механизму Proof-of-Work, то есть решают математические задачи по шифрованию данных. Это требует большого количества энергии и вычислительной мощности, а после успешного завершения к блокчейну добавляется новый блок. В скором будущем будет внедрен механизм Proof- of-Stake, который потребует приобретения определенного количества эфира, который будет использоваться для обсчета новых блоков.

Самым большим отличием от блокчейна Bitcoin является возможность размещать приложения непосредственно в блокчейне. Именно этой теме будет посвящен данный раздел.

Приложения с максимальной доступностью

Технология блокчейн подразумевает, что все приложения обладают невероятной устойчивостью и надежностью, поскольку исполняются непосредственно в блокчейне - единый *сервер* отсутствует,код выполняется на всех узлах блокчейна одновременно. Это открывает большие возможности для решения сложных вычислительных задач, а также построения высокораспределенных сред.

Основы работы среды Solidity

Среда Solidity представляет из себя высокоуровневый *язык программирования*. Существуют и другиеспособы разработки приложений для блокчейна, но общим правилом является необходимость компиляции текста на языке программирования в байткод, который затем размещается в блокчейн посредством транзакции. Такие транзакции очень похожи на используемые в блокчейне Bitcoin, когда вы отправляете биткойны с одного адреса на другой; в среде Ethereum для тех же целей используются единицы эфира. Если есть потребность отправить в блокчейн байткод, то он прикрепляется к транзакции как данные, а в самой транзакции при этом должно быть пустым *поле* получателя *To* - в этом случае блокчейн создаст новый *адрес* для размещения этого байткода.

Практический пример

Рисунок 12 – функция ABC

Например, предположим, что у нас есть *функция ABC*, требующая *параметр* a. Если a меньше 50, *функция* возвращает 10, в противном случае a. *Компилятор* обрабатывает функцию, и в блокчейн отправляется новая транзация со следующими значениями полей: *from* содержит *адрес* отправителя

- ваш *адрес*, *поле value* пусто, как и *поле to* (это самый важный момент), а *поле data* содержит байткод из функции, созданной в Solidity. В процессе обсчета транзакции этот код будет добавлен в очередной блок, получит собственный *адрес*, например, *0xabcdef001*, или какой-нибудь другой, и у каждого пользователя сети появится возможность взаимодействовать с этим кодом по заданному адресу.

Важность обсчета кода и учета валюты в одном блокчейне

Блокчейн представляет собой значительно распределенную базу данных. Это означает, что при сохранении в блокчейне какой-либо величины или участка кода их больше нельзя удалить, их доступность крайне высока. Ни какие-либо величины, ни участки кода не доступны изцентрализованного источника, поэтому никакому правительству не под силу ограничить *доступ* к ним или удалить данные из блокчейна, если только они не выключат все узлы сети по всему миру. Эта концепция напоминает сохраненные процедуры в MySQL, только применимо по отношению к коду. В частности, в среде MySQL вы можете запускать некоторые программы, изменяющие запросы *SELECT* или *RETURN* - аналогично в блокчейне, особенно в

Ethereum, с помощью смарт-контрактов можно изменять значения переменных или данные, отправлять валюту и другим образом взаимодействовать с другими смарт-контрактами.

В среде Ethereum и обработка кода, и валютные *операции* проводятся в едином *поле*. Это открывает широкие возможности для различных приложений из областей условного депонирования, краудфандинга, страхового дела, операций с недвижимостью, сферы услуг, юриспруденции и прочих. В настоящее время наблюдается множество проектов, использующих возможности запуска кода в блокчейне и работы с криптовалютой с последующим проведением краудфандинговых кампаний.

Классические примеры распределенных приложений

Приведем несколько примеров распределенных приложений. Начнем с ДАО - демократических автономных организаций. Эта система представляет собой платформу для краудфандинга. Одно время широко обсуждалась в прессе, поскольку разработчики смогли привлечь шестьдесят миллионов долларов США в виде инвестиций. К большому сожалению, она впоследствии была взломана, но оставила значительный след в сознании людей, благодаря ясной логике и новому подходу к краудфандингу, при котором не представлялось возможным собрать средства и сбежать (в отличие, например, от Kickstarter, который тоже принимает средства для разработки новых продуктов, но нет гарантии, что он не обанкротится). В случае с ДАО отсутствует центральное передаточное звено, способное скрыться с деньгами, намеренно, поскольку все договоренности обеспечиваются смарт-контрактами.

Вторым распространенным типом приложений являются решения по обмену валюты. В настоящее время наблюдается всплеск числа ICO, - первичных размещений криптовалюты - реализуемых с помощью токенов стандарта *ERC*, которые можно напрямую обменивать в среде блокчейна, то

есть менять токены на эфир или токены на токены, и вся эта логика хранится непосредственно вблокчейне.

Токены - это участки кода, связанные с пользовательскими учетными записями и базами данных и используемые как валюта, баллы в программах лояльности, индикаторы доли в компании, жетоны в играх виртуальной реальности и так далее. Все эти платежные средства учитываются и хранятся в среде блокчейна.

И, наконец, *базы данных*. Снабдив их некоторой логикой, можно вести в блокчейне учет владельцев земельных участков, дипломов об образовании (например, Массачусетский технологический институт выпускал свои сертификаты в блокчейне), даже законов. В случае использования блокчейна нет необходимости прописывать положения закона в каком-то документе, вместо этого можно задать условия работы смарт-контрактов, сразу разрешая или запрещая какие-то действия в блокчейне. В перспективе можно даже прийти к автоматическому списанию средств со счета, например, в качестве штрафа за неправильную парковку.

Как работает доступ к блокчейну

Для обеспечения доступа к блокчейну используются узлы сети Ethereum, взаимодействующие друг с другом посредством протокола Ethereum. Каждый *узел сети* может обращаться к любому другому. Одним из узлов, доступных для свободной загрузки, является Go-Ethereum. Он, как и остальные реализации, подключается и взаимодействует с сетью посредством протокола Ethereum.

С другой стороны, для выполнения операций в блокчейне можно применять *удаленный вызовпроцедур* (*Remote Procedure Call, RPC*), запуская файлы *JSON*, созданные на JavaScript.

Рисунок 13 – Вызов процедур (Remote Procedure Call, RPC)

Удаленный вызов процедур можно реализовать через протокол *HTTP*, что позволяет взаимодействовать пользователям с узлами сети, а самим узлам - друг с другом посредством протокола Ethereum. Важно понять схему: *пользователь* работает с файлами *JSON* для удаленного вызова процедур, а узлы передают эту информацию между собой по протоколу Ethereum. Это напоминает работу в консоли MySQL, когда *пользователь* задает запросы MySQL (в случае с Ethereum отправляются команды для удаленного вызова процедур в формате *JSON*), а узлы сети MySQL обмениваются информацией по протоколу MySQL (в блокчейне узлы взаимодействуют по протоколу Ethereum).

Задание

1. Установить кошелек MetaMask. Подключаемся только к тестовой сети Rinkeby.
2. Откройте несколько счетов (минимум 3).
3. Получите эфир для дальнейшей работы.
4. Распределите полученный эфир между тремя счетами.
5. Ознакомьтесь с возможностями кошелька MetaMask.

Контрольные вопросы

1. Что такое Go-Ethereum?
2. Что такое Web3?
3. Что такое Remix?
4. Что такое Solidity?
5. Что такое MetaMask?

6. Как MetaMask взаимодействует с браузером?
7. Укажите несколько различий сети Ethereum и блокчейна Bitcoin?

Лабораторная работа №3 Разработка простейшей веб-страницы использованием языка гипертекстовой разметки HTML.

Цель работы: Изучение языка гипертекстовой разметки HTML. Получение основных навыков создания HTML-документов.

Подготовка к работе

В процессе подготовки к лабораторной работе необходимо изучить:

методы создания HTML документов;

рабочее задание и методические указания к его выполнению.

World Wide Web, HTML и HTTP

World Wide Web (Web) - это глобальная сеть информационных ресурсов. Для того, чтобы сделать эти ресурсы доступными наиболее широкой аудитории, в Web используются три механизма:

Единая схема наименования для поиска ресурсов в Web (например, URI).

Протоколы для доступа к именованным ресурсам через Web (например, HTTP).

Гипертекст для простого перемещения по ресурсам (например, HTML).

Каждый ресурс в Web - документ HTML, изображение, видеоклип, программа и т.д. - имеет адрес, который может быть закодирован с помощью универсального идентификатора ресурсов (Universal Resource Identifier), или URI.

URI обычно состоят из трех частей:

Схема наименования механизма, используемого для доступа к ресурсу.

Имя машины, на которой располагается ресурс.

Имя собственно ресурса, зачданное в виде пути.

Рассмотрим URI спецификации четвертой версии HTML на сервере W3C:

http://www.w3.org/TR/PR-html4/cover.html

Этот URI может читаться следующим образом: этот документ можно получить по протоколу HTTP (см. [RFC2068]), он располагается на машине www.w3.org, путь к этому документу - "/TR/PR-html4/cover.html".

Кроме того, в документах в формате HTML можно увидеть схемы "mailto" для электронной почты и "ftp" для протокола FTP. Приведем пример URI, относящийся к почтовому ящику пользователя:

...текст...

Комментарии отправляйте <A href="mailto:joe@someplace.com"> Джо Кулу</A>.

Некоторые URI указывают на местоположение внутри ресурса. Этот тип URI заканчивается символом "#", за которым следует указатель (идентификатор фрагмента). Например, следующий URI указывает на фрагмент с именем section_2:

http://somesite.com/html/top.html#section_2

Относительный URI не содержит информации о схеме наименования. Путь в нем указывает на ресурс на машине, на которой находится текущий документ. Относительные URI могут содержать компоненты относительного пути (например, ".." означает один уровень выше в иерархии) и идентификаторы фрагментов.

Относительные URI приводятся к полным URI с помощью базового URI. В качестве примера приведения относительного URI предположим, что у нас имеется базовый URI "http://www.acme.com/support/intro.html". Относительный URI в следующей ссылке:

<A href="suppliers.html">Suppliers</A>

будет преобразован в полный URI "http://www.acme.com/support/suppliers.html", а относительный URI в следующем фрагменте

<IMG src="../icons/logo.gif" alt="logo">

будет преобразован в полный URI "http://www.acme.com/icons/logo.gif".

В HTML URI используются для:

ссылки на другие документы или ресурсы (элементы A и LINK).

ссылки на внешние таблицы стилей или скрипты (элементы LINK и SCRIPT).

включения в страницу изображений, объектов или апплетов (элементы IMG, OBJECT, APPLET и INPUT).

создания изображений-карт (элементы MAP и AREA).

отправки форм (FORM).

создания документов с использованием кадров (элементы FRAME и IFRAME).

ссылок на внешние источники (элементы Q, BLOCKQUOTE, INS и DEL).

ссылок на соглашения о метаданных, описывающих документ (элемент HEAD).

Чтобы представить информацию для глобального использования, нужен универсальный язык, который понимали бы все компьютеры. Языком публикации, используемым в World Wide Web, является HTML (HyperText Markup Language - язык разметки гипертекстов).

HTML дает авторам средства для:

публикации электронных документов с заголовками, текстом, таблицами, списками, фотографиями и т.д.

загрузки электронной информации с помощью щелчка мыши на гипертекстовой ссылке.

разработки форм для выполнения транзакций с удаленными службами, для использования в поиске информации, резервировании, заказе продуктов и т.д.

включения электронных таблиц, видеоклипов, звуковых фрагментов и других приложений непосредственно в документы.

Язык HTML был разработан Тимом Бернерс-Ли во время его работы в CERN и использовался в браузере Mosaic, разработанным в NCSA. В 1990-х годах он получил широкое распространение благодаря быстрому росту Web.

В это время HTML был расширен и дополнен. Важнейшие черты языка были отражены в ряде спецификаций.

HyperText Transfer Protocol (HTTP) - это протокол высокого уровня (а именно, уровня приложений), обеспечивающий передачу данных, требующуюся для распределенных информационных систем гипермедиа. HTTP используется в World Wide Web с 1990 года.

Практические информационные системы требуют большего, чем примитивный поиск, модификация и аннотация данных. HTTP/1.0 предоставляет открытое множество методов, которые могут быть использованы для указания целей запроса. Они построены на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется URI, в виде местонахождения (URL) или имени (URN). Формат сообщений сходен с форматом Internet Mail или Multipurpose Internet Mail Extensions (MIME-Многоцелевое Расширение Почты Internet).

HTTP/1.0 используется также для коммуникаций между различными пользовательскими браузерами и шлюзами, дающими гипермедиа доступ к существующим Internet протоколам, таким как SMTP, NNTP, FTP, Gopher и WAIS. HTTP/1.0 разработан, чтобы позволять таким шлюзам через proxy серверы, без какой-либо потери передавать данные с помощью упомянутых протоколов более ранних версий.

HTTP основывается на парадигме запросов/ответов. Запрашивающая программа (клиент) устанавливает связь с обслуживающей программой-получателем (сервер) и посылает запрос серверу в следующей форме: метод запроса, URI, версия протокола, за которой следует MIME-подобное сообщение, содержащее управляющую информацию запроса, информацию о клиенте и, может быть, тело сообщения. Сервер отвечает сообщением, содержащим строку статуса (включая версию протокола и код статуса - успех или ошибка), за которой следует MIME-подобное сообщение, включающее в себя информацию о сервере, метаинформацию о содержании ответа, и, вероятно, само тело ответа

В Internet коммуникации обычно основываются на TCP/IP протоколах. Для WWW номер порта по умолчанию - TCP 80, но также могут быть использованы и другие номера портов - это не исключает возможности использовать HTTP в качестве протокола верхнего уровня.

Для большинства приложений сеанс связи открывается клиентом для каждого запроса и закрывается сервером после окончания ответа на запрос. Тем не менее, это не является особенностью протокола. И клиент, и сервер должны иметь возможность закрывать сеанс связи, например, в результате какого-нибудь действия пользователя. В любом случае, разрыв связи, инициированный любой стороной, прерывает текущий запрос, независимо от его статуса.

Структура HTML документа

Данные в формате HTML похожи на текстовой файл, за исключением того, что некоторые из символов интерпретируются как разметка. Разметка придает документу некую структуру.

Данные представляют собой иерархию элементов. Каждый элемент имеет имя, атрибуты и несет некую информацию. Большинство элементов представлены в документе в виде начальной метки, указывающей имя и атрибуты. Далее следует собственно содержание элемента. И наконец, заканчивает все это конечная метка.

HTML документ состоит из следующих базисных элементов:

тип документа <HTML></HTML> (начало и конец файла);

заголовок <HEAD></HEAD> (описание документа, например его имя);

имя документа <TITLE></TITLE> (должно быть в заголовке);

тело <BODY></BODY> (содержит публикуемые данные).

Например:

<HTML>

```
<HEAD>
    <TITLE>Название документа</TITLE>
</HEAD>
<BODY>
Информация, содержащаяся в документе.
</BODY>
</HTML>.
```

Каждый элемент HTML документа начинается с метки, меткой же и заканчивается каждый непустой элемент. Начальные метки выделяются символами < и > , а конечные - символами </ и > .

Имя элемента следует в метке сразу за символом открытия <. Имя начинается с буквы, за которой могут следовать еще 33 буквы, цифры, пробелы или дефисы. В именах игнорируется разница между прописными и строчными буквами.

Начальная метка позволяет вставить между именем и символом > пробелы и атрибуты. Атрибут состоит из имени, символа равенства и значения. Слева и справа от символа равенства можно оставлять пробелы.

Значение атрибута указывается в виде строки, заключенной в одинарные или двойные кавычки.

Элементы языка HTML

Документы должны (но не обязательно) содержать элемент HEAD, за которым следует элемент BODY.

Тэг <HEAD>

Элемент HEAD содержит всю информацию о документе в целом. Однако он не содержит какого-либо текста. Последний должен находиться в элементе BODY. В элементе заголовка HEAD можно использовать лишь строго заданный набор элементов.

Нижеприведенные элементы определяют общие свойства документа. Они должны появляться в элементе HEAD. Порядок следования элементов значения не имеет.

<TITLE> - название элемента.

<ISINDEX> - элемент, посылаемый серверу вместе с документом, предназначенным для информации к поиску.

<LINK> - элемент, определяющий связь этого документа с другими. В , документе может присутствовать несколько элементов LINK.

<BASE> - запись, сделанная на языке URL при фиксировании данного , документа.

<META> - метаинформация.

Например:

```
<HEAD>
    <LINK REV="made" TITLE="Vasia Ivanov" HREF="mailto:vasia@ivanov.ru">
    <META NAME="KeyWords" CONTENT="ключевые слова для поисковых машин">
    <META NAME="Description" CONTENT="краткое описание документа">
    <TITLE>Домашняя страничка Васи Иванова</TITLE>
</HEAD>
```

Тэг <BODY>

В противоположность элементу HEAD элемент BODY содержит всю ту информацию, из которой собственно и состоит рассматриваемый документ.

Далее рассматриваются элементы, применяемые для представления и форматирования информации в документе.

Определение структуры текста

Для определения структуры текста используются следующие тэги.

<H?></H?> - заглавие (стандарт определяет 6 уровней).

<H? ALIGN=LEFT|CENTER|RIGHT></H?> - заглавие с выравниванием.

<DIV></DIV> - секция.

<DIV ALIGN=LEFT|RIGHT|CENTER></DIV> - секция с выравниванием.

<BLOCKQUOTE></BLOCKQUOTE> - цитата (обычно выделяется отступом).

<EM></EM> - выделение (обычно изображается курсивом).

<STRONG></STRONG> - дополнительное выделение (обычно изображается жирным шрифтом).

<CITE></CITE> - отсылка, цитата (обычно курсив).

<CODE></CODE> - код (для листингов кода).

<SAMP></SAMP> - пример вывода.

<KBD></KBD> - ввод с клавиатуры.

<VAR></VAR> - переменная.

<BIG></BIG> - большой шрифт.

<SMALL></SMALL> - маленький шрифт.

Внешний вид

Внешний вид текста определяется с использованием следующих тэгов.

<B></B> - жирный.

<I></I> - курсив.

 - верхний индекс.

 - нижний индекс.

<TT></TT> - печатная машинка (изображается как шрифт фиксированой ширины).

<PRE></PRE> - форматированый (сохранить формат текста как есть).

<PRE WIDTH=?></PRE> - ширина (в символах).

<CENTER></CENTER> - центрировать (как текст, так и графика).

<FONT SIZE=?></FONT> - размер шрифта (от 1 до 7).

<FONT SIZE="+|-?"></FONT> - изменить размер шрифта.

<FONT COLOR="#$$$$$$"></FONT> - цвет шрифта.

<FONT FACE="***"></FONT> - выбор шрифта.

Ссылки и графика

Следующие тэги используются для организации ссылок и работы с графикой.

<A HREF="URL"></A> - ссылка.

<A HREF="URL#***"></A> - ссылка на закладку (в другом документе).

<A HREF="#***"></A> - ссылка на закладку (в том же документе).

<A HREF="URL" TARGET="***| |_blank|_self|_parent|_top"></A>

- ссылка на другое окно. Например: <A HREF="http://www.torrentino.com/" TARGET="_blank"> Торрентино (Большое количество разных файлов)</A>

<A NAME="***"></A> - определить закладку.

<IMG SRC="URL"> - графика.

<IMG SRC="URL" ALIGN=TOP|BOTTOM|MIDDLE|LEFT|RIGHT> - графика с выравниванием.

<IMG SRC="URL" ALT="***"> - альтернатива (выводится если картинка не изображается).

<IMG SRC="URL" WIDTH=? HEIGHT=?> - размеры (в точках).

<IMG SRC="URL" BORDER=?> - окантовка (в точках).

<IMG SRC="URL" HSPACE=? VSPACE=?> - отступ (в точках).

Разделители

Следующие тэги используются в качестве разделителей.

<P></P> параграф (закрывать элемент часто не обязательно).

<P ALIGN=LEFT|CENTER|RIGHT></P> - выравнивание.

 - новая строка (одиночный перевод строки).

<BR CLEAR=LEFT|RIGHT|ALL> - убрать выравнивание.

<HR> - горизонтальный разделитель.

<HR ALIGN=LEFT|RIGHT|CENTER> - выравнивание.

<HR SIZE=?> - толщина (в точках).

<HR WIDTH=?> - ширина (в точках).

<HR WIDTH="%"> - ширина в процентах (в процентах от ширины страницы).

<HR NOSHADE> - сплошная линия (без трехмерных эффектов).

Списки

Для реализации списков используются следующие тэги.

<UL><LI></UL> - неупорядоченный (<LI> перед каждым элементом).

<UL COMPACT></UL> - компактный.

<UL TYPE=DISC|CIRCLE|SQUARE> - тип метки (для всего списка).

<LI TYPE=DISC|CIRCLE|SQUARE> - (этот и последующие).

<OL><LI></OL> - нумерованый (<LI> перед каждым элементом).

<OL COMPACT></OL> - компактный.

<OL TYPE=A|a|I|i|1> - тип нумерации (для всего списка).

<LI TYPE=A|a|I|i|1> - (этот и следующие).

<OL START=?> - первый номер (для всего списка).

<LI VALUE=?> - (этот и следующие).

<DL><DT><DD></DL> - список определений (<DT>=термин, <DD>=определение).

<DL COMPACT></DL> - компактный.

<MENU><LI></MENU> - меню (<LI> перед каждым элементом).

<MENU COMPACT></MENU> - компактное.

<DIR><LI></DIR> - каталог (<LI>перед каждым элементом).

<DIR COMPACT></DIR> - компактный.

Фон и цвета

Следующие тэги определяют фон и цвета.

<BODY BACKGROUND="URL"> - фоновая картинка.

<BODY BGCOLOR="#$$$$$$"> - цвет фона.

<BODY TEXT="#$$$$$$"> - цвет текста.

<BODY LINK="#$$$$$$"> - цвет ссылки.

<BODY VLINK="#$$$$$$"> - пройденная ссылка.

<BODY ALINK="#$$$$$$"> - активная ссылка.

Специальные символы

(обязаны быть в нижнем регистре)

&#?; - специальный символ (где ? это код ISO 8859-1).

< - <.

> - >.

& - &.

" - ".

® - торговая марка TM.

© - copyright.

  - неразделяющий пробел.

Формы

Для работы с формами используются следующие тэги.

<FORM ACTION="URL" METHOD=GET|POST></FORM> - определить форму.

<INPUT TYPE="TEXT|PASSWORD|CHECKBOX|RADIO|IMAGE|HIDDEN|SUBMIT|RESET"> - поле ввода.

<INPUT NAME="***"> - имя поля.

<INPUT VALUE="***"> - значение поля.

<INPUT CHECKED> - отмечен (checkboxes и radio boxes).

<INPUT SIZE=?> - размер поля (в символах).

<INPUT MAXLENGTH=?> - максимальная длина (в символах).

<SELECT></SELECT> - список вариантов.

<SELECT NAME="***"></SELECT> - имя списка.

<SELECT SIZE=?></SELECT> - число вариантов.

<SELECT MULTIPLE> - множественный выбор (можно выбрать больше одного).

<OPTION> - опция (элемент который может быть выбран).

<OPTION SELECTED> - опция по умолчанию.

<TEXTAREA ROWS=? COLS=?></TEXTAREA> - ввод текста, размер.

<TEXTAREA NAME="***"></TEXTAREA> - имя текста.

Таблицы

Для организации таблиц используются следующие тэги.

<TABLE></TABLE> - определить таблицу.

<TABLE BORDER=?></TABLE> - окантовка таблицы.

<TABLE CELLSPACING=?> - расстояние между ячейками.

<TABLE CELLPADDING=?> - дополнение ячеек.

<TABLE WIDTH=?> - желаемая ширина (в точках).

<TABLE WIDTH="%"> - ширина в процентах (проценты от ширины страницы).

<TR></TR> - строка таблицы.

<TR ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.

<TD></TD> - ячейка таблицы (должна быть внутри строки).

<TD ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.

<TD NOWRAP> - без перевода строки.

<TD COLSPAN=?> - растягивание по колонке.

<TD ROWSPAN=?> - растягивание по строке.

<TD WIDTH=?> - желаемая ширина (в точках).

<TD WIDTH="%"> - ширина в процентах (проценты от ширины страницы).

<TD BGCOLOR="#$$$$$$"> - цвет ячейки.

<TH></TH> - заголовок таблицы (как данные, но жирный шрифт и центровка).

<TH ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM> - выравнивание.

<TH NOWRAP> - без перевода строки.

<TH COLSPAN=?> - растягивание по колонке.

<TH ROWSPAN=?> - растягивание по строке.

<TH WIDTH=?> - желаемая ширина (в точках).

<TH WIDTH="%"> - ширина в процентах (проценты ширины таблицы).

<TH BGCOLOR="#$$$$$$"> - цвет ячейки.

<CAPTION></CAPTION> - заглавие таблицы.

<CAPTION ALIGN=TOP|BOTTOM> - выравнивание (сверху/снизу таблицы).

Фреймы

Для работы с фреймами используются следующие тэги.

<FRAMESET></FRAMESET> - документ с фреймами (вместо <BODY>).

<FRAMESET ROWS=,,,></FRAMESET> - высота строк (точки или %).

<FRAMESET ROWS=*></FRAMESET> - высота строк (* = относительный размер).

<FRAMESET COLS=,,,></FRAMESET> - ширина колонок (точки или %).

<FRAMESET COLS=*></FRAMESET> - ширина колонок (* = относительный размер).

<FRAMESET BORDER=?> - ширина окантовки.

<FRAMESET FRAMEBORDER="yes|no"> - окантовка.

<FRAMESET BORDERCOLOR="#$$$$$$"> - цвет окантовки.

<FRAME> - определить фрейм (содержание отдельного фрейма).

<FRAME SRC="URL"> - документ.

<FRAME NAME="***"|_blank|_self| _parent|_top> - имя фрейма.

<FRAME MARGINWIDTH=?> - ширина границы (правая и левая границы).

<FRAME MARGINHEIGHT=?> - высота границы (верхняя и нижняя границы).

<FRAME SCROLLING="YES|NO|AUTO"> - скроллинг.

<FRAME NORESIZE> - постоянный размер.

<FRAME FRAMEBORDER="yes|no"> - окантовка.

<FRAME BORDERCOLOR="#$$$$$$"> - цвет окантовки.

<NOFRAMES></NOFRAMES> - содержание без фреймов (для просмотрщиков не поддерживающих фреймы).

Комментарий

<!-- *** --> - тэг комментария (игнорируется браузером).

Список используемых терминов и сокращений

CGI (Common Gateway Interface) - интерфейс, позволяющий взаимодействовать программам клиента с программами, запущенными на сервере.

Cookie - порция информации, оставляемая на компьютере WEB-клиента программой, запущенной на стороне WEB-сервера. Применяется для сохранения данных, специфичных для данного клиента, например: имя пользователя, количество посещений сервера, регион пользователя и т.п.

CSS (Cascading Style Sheets) - -язык иерархических стилевых спецификаций. Главная цель CSS - отделить структуру документа от его оформления и позволить автору страницы самому решать, как должен выглядеть тот или иной элемент содержания. CSS не только освобождает от "обязательного" форматирования тех или иных тегов (например, полужирного начертания заголовков), но и добавляет множество новых степеней свободы, о которых раньше не приходилось и мечтать (например, возможность изменения интерлиньяжа - расстояния между строками текста).

FTP (File Transfer Protocol) - протокол передачи файлов, а также программа, его реализующая. Протокол был разработан для передачи файлов между компьютерами, использующими сеть на основе TCP/IP, в том числе и в Internet. Для доступа к некоторой информации посредством FTP на компьютере, с которого осуществляют доступ, должен быть установлен т.н. FTP-клиент, а на другом, соответственно, FTP-сервер. В WEB-практике FTP-доступ используется для доступа к страничкам WEB-сайта, обычно расположенным на сервере провайдера.

JAVA - межплатформенный язык программирования. Программы, написанные на JAVA, запускаются на стороне клиента, используя т.н. виртуальную машину (VM) Java. Применяется для создания динамических страничек, организиции доступа к базам данных посредством Internet и т.п.

JAVAScript - язык программирования, основанный на объектном представлении броузера. Текст программы встроен непосредственно в HTML-документ и интерпретируется самим броузером. Применяется в основном для создания таких эффектов, как: бегущая строка, рисунки, изменяющие свой вид при подведении курсора и т.д.

Perl - язык программирования. Программы, написанные на Perl, запускаются на стороне сервера. В основном применяется на UNIX-ориентированных WEB-серверах. Применяется для обеспечения доступа к базам данным, создания динамических страничек и т.п.

Script - программа, написанная на каком-либо языке программирования для взаимодействия клиента с сервером. Например: Script на Perl для подсчета количества посещений.

Tag (Тэг) - элемент HTML, представляет из себя текст, заключенный в угловые скобки <>, является активным элементом, изменяющим представление следующей за ним информации. Может иметь некоторые атрибуты. Обычно имеются два тэга - открывающий и закрывающий. Например и - данные тэги описывают текст, находящийся между ними, как полужирный.

Доменное имя (Domain Name) - уникальный идентификатор, который назначается определенному IP-адресу. Доменное имя дает возможность обращаться к компьютеру по имени типа www.company.com, вместо запоминания его числового эквивалента.

Кодовая таблица - таблица соответствий символов и их положений в таблице. Исторически сложилось так, что в России есть несколько несовместимых кодировок, т.е. одинаковые символы имеют различные коды в разных кодировках. Это приводит к тому, что при просмотре страничек не в

той кодировке, на которую настроен броузер, экран засоряется непонятными символами. В России распространены следующие кодировки: WIN1251 (Windows), KOI-8 (Unix), CP866(DOS), Macintosh, ISO-8859-5 (Unix).

Фреймы (Frames) - элементы HTML, появившиеся в броузерах версий 3.0. Позволяют разделить страницу на несколько независимых окон и в каждом из них размещать свою собственную WEB-страничку. Возможна ссылка из одного окна в другое. Применяется в основном для организации постоянно находящихся на экране меню, в то время как в другом окне располагается непосредственно сама информация.

Задание на лабораторную работу

Создать на языке гипертекстовой разметки HTML пример Вашей персональной домашней странички.

Страница должна содержать:

Заголовок (тэг <HEAD>), включающий в себя:

название страницы (тэг <TITLE>), содержащее Ваше имя;

обозначение кодировки страницы (рекомендуется использовать кодировку WINDOWS-1251, тэг <meta http-equiv=Content-Type content="text/html; charset=windows-1251">);

Описание страницы (тэг <meta name="Description">);

Описание ключевых слов (тэг <meta name="Keywords">).

Тело страницы (тэг <BODY>), включающее в себя:

Меню с ссылками внутри документа;

Текст с краткой информацией о себе;

Изображение, "обтекаемое" вышеуказанным текстом;

Ссылку на адрес Вашей электронной почты;

Ссылки на различные сайты (сайты должны открываться в новом окне);

Таблицу, содержащую данные о Ваших оценках за прошлый семестр.

Текст на странице должен быть оформлен разными стилями, размерами и шрифтами.

Порядок выполнения работы:

Включить ПЭВМ и запустить любой текстовый редактор (желательно использовать текстовый редактор NOTEPAD.EXE, находящийся в стандартной поставке WINDOWS, так как он позволяет редактировать и сохранять текст в кодировке windows-1251).

Написать пример персональной страницы согласно п. 7.

Сохранить готовую программу в файл с форматом HTM или HTML.

Просмотреть файл-страницу в окне браузера (например Internet Explorer) и при необходимости устранить ошибки.

Продемонстрировать страницу преподавателю.

Содержание отчета:

Отчет должен содержать:

Листинг файла-страницы на языке HTML.

Описание используемых тэгов.

Распечатку окна броузера с загруженной персональной страницей.

Контрольные вопросы:

Поясните понятие URI.

Что из себя представляет язык HTML?

Что из себя представляет протокол HTTP?

Опишите структуру HTML документов.

Назовите элементы языка HTML, применяемые внутри тэга <HEAD>.

Назовите элементы языка HTML, применяемые для разметки текста.

Назовите элементы языка HTML, применяемые для построения таблиц.

Каким образом можно определить цвета текста и фона?

Лабораторная работа №4 Продвижение сайта компании

Цель работы: составить бриф и медиа-план продвижения сайта.

Составление брифа

1) Прочитайте описание деятельности компании в соответствии с выданным вариантом.

2) Определите 2-3 конкурентов компании.

3) Проведите сравнительный анализ сайтов компаний конкурентов, выделите их сильные и слабые стороны.

4) На основе проведенного анализа составьте бриф на разработку сайта вашей компании.

Исходные данные:

Бриф

	Информция	Значение
1	Название компании:	
2	Название сайта:	
3	URL для размещения сайта:	
4	Задачи разработчика:	
5	Цель создания сайта:	
6	Сфера деятельности компании:	
7	Целевая аудитория:	
8	На какие географические регионы должен быть ориентирован сайт:	
9	Языковые версии сайта:	
10	Сайты конкурентов:	
11	Материалы,	

	предоставленные заказчиком:	
12	Структура сайта:	
13	Компоненты сайта (новости, анкетирование, интернет-магазин, баннерная реклама, счетчики посещений, др.):	
14	Цветовая гамма сайта:	
15	Вид верстки:	
16	Ширина веб-страницы:	
17	Браузеры, в которых будет просматриваться сайт, и их минимальная версия:	
18	График работ:	
19	Передача разработки:	

Составление медиа-плана

1) Прочитайте описание деятельности компании в соответствии с выданным вариантом.

2) Опишите возможные способы продвижения товаров (услуг) в сети Интернет, которые целесообразно применять для возможной целевой аудитории компании.

3) Определите 2–3 площадки, где возможно разместить рекламу вашей деятельности. Определите ценовую политику данных площадок.

4) Определите 1–2 возможных варианта реализации традиционной рекламы для вашего интернет-проекта и их стоимостные характеристики.

5) Составьте список базовых затрат, которые могут потребоваться до начала активной рекламы сайта (например разработка рекламных материалов, регистрация в каталогах, оптимизация содержания сайта и др.)

6) Составьте медиа-план на 1 месяц, включающий интернет рекламу и традиционные способы рекламы из 4–5 позиций.

Реклам о-носитель	Обосно-вание выбора рекламон о-сителя	Форма рекламы	Размер	Мест о разме-щени я	Охват (тира ж или ауди-тория)	Частот а охвата (коли-чество и дни выхода в недел ю)	Стоимос ть одного выхода (руб.)	Сумма за все расход ы (руб.)
1.								
2.								
3.								
….								
Итого								

Контрольные вопросы

1. В чём суть понятия эффективности ЭК? Термины и их определения?
2. Раскройте содержание понятий критерия и показателя эффективности.
3. Укажите методический подход к определению эффективности ЭК.
4. Каковы методы оценки эффективности систем ЭК?

5. Какие Вам известны способы оценки экономической эффективности систем ЭК?

6. Перечислите маркетинговые показатели эффективности систем ЭК.

7. Охарактеризуйте известные Вам показатели эффективности рекламы в Интернете.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ашманов И. С. Продвижение сайта в поисковых системах / И. С. Ашманов, А. А. Иванов. – М. : И.Д. Вильяме, 2007. – 304 с.
2. Бирюков П. И. Интернет-шоппинг. – М. : Феникс, 2012. – 160 с.
3. Бокарёв Т. Энциклопедия Интернет-рекламы / Т. Бокарёв. PROMO.RU, 2000. – 416 с.
4. Гаврилов Л. П. Мобильные телекоммуникации в электронной коммерции и бизнесе : учебное пособие / Л. П. Гаврилов, С. В. Соколов. – М. : Финансы и статистика, 2006. – 336 с.
5. Гаврилов Л. П. Мобильный бизнес : практ. пособие / Л. П. Гаврилов, С. В Соколов. − М. : Моск. университет потреб. кооперации, 2004. − 170 с.
6. Гаврилов Л. П. Основы электронной коммерции и бизнеса / Л. П. Гаврилов. – М. : СОЛОН-ПРЕСС, 2009. – 592 с.
7. Гаврилов Л. П. Электронная коммерция. Учебное пособие по выполнению практических работ / Л. П. Гаврилов. – М. : СОЛОН-ПРЕСС, 2008. – 112 с.
8. Генкин А. С. Планета Web-денег / А. С. Генкин. – М. : КноРус, 2008. – 576 с.
9. Гитомер Дж. Бизнес в социальных сетях. Как продавать, лидировать и побеждать. – СПб : Питер, 2012. 192 с.
10. Дэвид Козье. Электронная коммерция / Козье Дэвид. – М. : Русская редакция; 2007. – 288 с.
11. Зуев М. Б. Продвижение сайтов в поисковых системах. Спасательный круг для малого бизнеса / М. Б. Зуев, П. А. Маурус, А. Г. Прокофьев. – М. : Бином. Лаборатория знаний, 2007. – 304 с.
12. Интернет-магазин. С чего начать, как преуспеть / под ред. А. Рябых. – СПб : Питер, 2012. – 208 с.