Оглавление

[Настройка среды 1](#_Toc474502367)

[Установка необходимых инструментов для работы 1](#_Toc474502368)

[Урок 17 2](#_Toc474502369)

[Получение текста построчно из тхт файла 2](#_Toc474502370)

[Send 2](#_Toc474502371)

[Method\_missing 3](#_Toc474502372)

[С помощью send можно динамически создавать новые методы 4](#_Toc474502373)

[Урок 24 5](#_Toc474502374)

[Важный момент при написании кода в HTML странице 5](#_Toc474502375)

[<script> 5](#_Toc474502376)

[Урок 25. SQLite3 6](#_Toc474502377)

[Основные команды 6](#_Toc474502378)

[Просмотр существующих таблиц в базе данных 6](#_Toc474502379)

[Добавление заголовков в таблицу и вставка пробелов между столбцами 6](#_Toc474502380)

[Удаление таблицы 6](#_Toc474502381)

[Урок 26 6](#_Toc474502382)

[datetime 6](#_Toc474502383)

[Ввод данных в таблицы SQL через RUBY 6](#_Toc474502384)

[Во избежание SQL injection 6](#_Toc474502385)

[Следует избегать ввода в таблицы таким кодом 7](#_Toc474502386)

[Вывод строк базы данных с RUBY 7](#_Toc474502387)

[Урок 27 8](#_Toc474502388)

# Настройка среды

## Установка необходимых инструментов для работы

1. Для начала необходимо установить [FAR](http://www.farmanager.com)
2. Далее, устанавливаем [RailsInstaller](http://railsinstaller.org/en). В это пакет входит несколько программ, в том числе Git Bash. Проверить, правильно ли прошла установка можно, набрав ruby –v. Также, необходимо проверить, добавились ли пути в ПЕРЕМЕННЫЕ СРЕДЫ. Для это щелкаем правой кнопкой на Компьютер -> свойства -> Дополнительно -> ПЕРЕМЕННЫЕ СРЕДЫ.
3. После установки FAR и RailsInstaller необходимо перезапустить FAR, чтобы новые переменные среды перезагрузились. Если что-то не запускается из под FAR, можно запустить из под Git Bash.
4. Теперь RUBY GEM.
   1. gem install sinatra
   2. gem install sinatra-reloader
5. Теперь [SQLite3](http://sqlite.org/download.html). Загрузить нужные библиотеки и скопировать их в Windows/System32/. Проверить, правильно ли выполнены действия – sqlite3 --version. Должна появиться версия.
6. Далее, скачиваем [SQLiteStudio](https://sqlitestudio.pl/index.rvt) и распаковываем в любую удобную директорию.
7. Теперь [Node.js](https://nodejs.org/en/). Скачиваем, устанавливаем.
8. После установки Node.js устанавливаем Ungit ---- > npm install ungit -g

# Урок 17

## Получение текста построчно из тхт файла

Output = File.open “file\_name.txt” , “r”

While (lines = output.gets)

………

end

## Send

Def mm

Puts “hello”

End

Send :mm -> hello

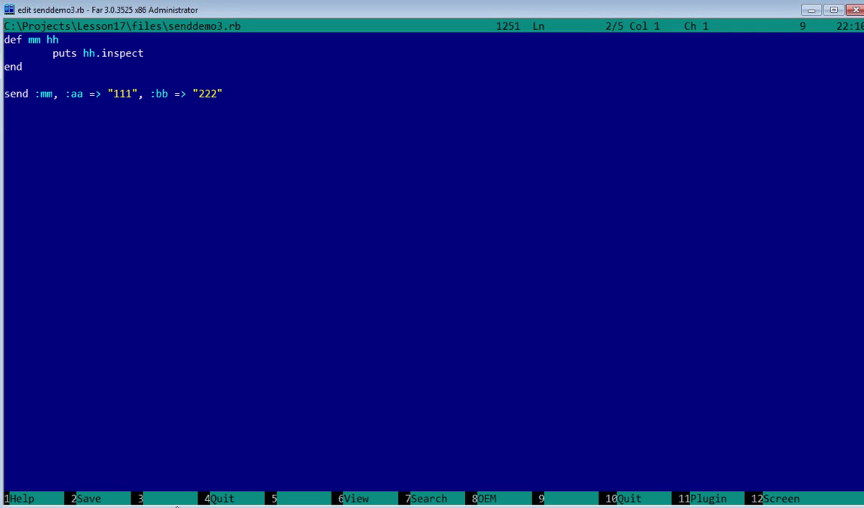
Def mm args

Puts args

End

Send :mm,1 -> 1

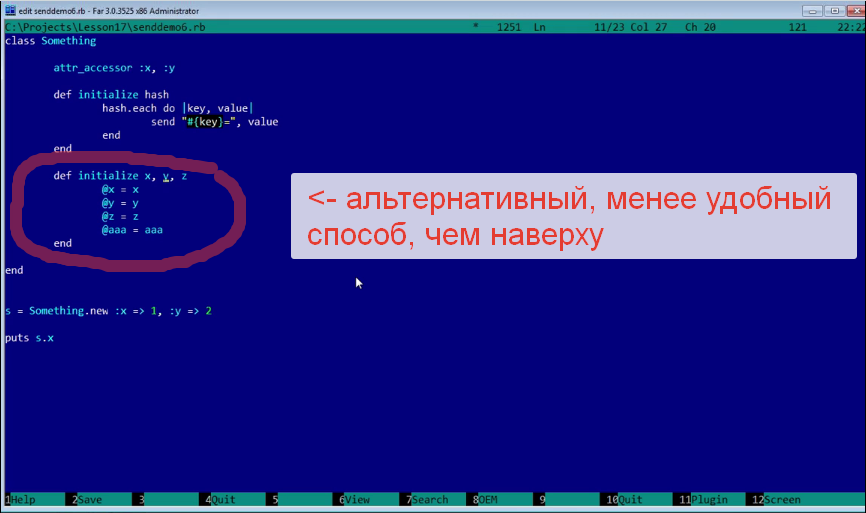
Вариант передачи хэша в метод. Send вызывает метод :mm и передает в него аргументы



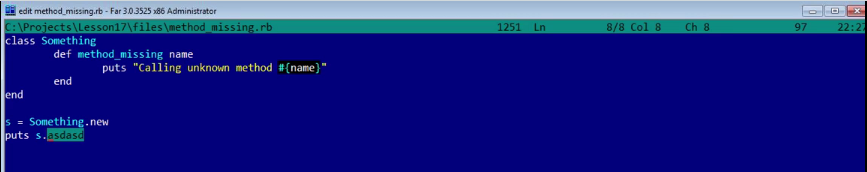
С помощью send можно организовать очень удобную передачу аргументов в метод в автоматическом режиме.

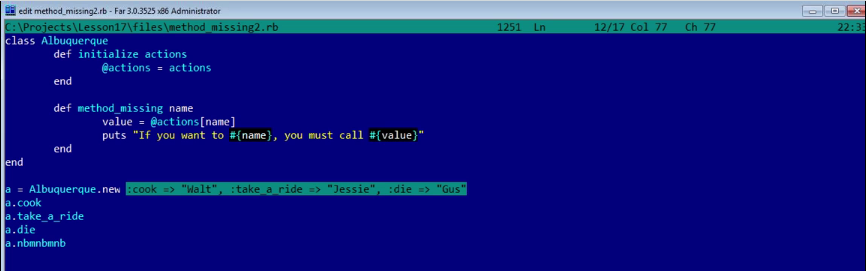
При увеличении колва аргументов в хэше нет необходимости прописывать каждый новый аргумент в метод, как указано в обведенном варианте.

Переменные добавляются в attr\_accessor при необходимости

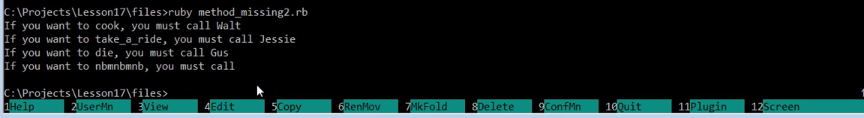


## Method\_missing

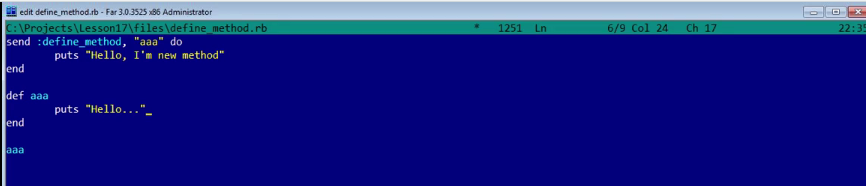


Зарезервированное системное имя. Вызывает какую-то операцию внутри метода, если имя вызываемого метода не существует. В данном случае puts. Или, как ниже, произойдет эдакий «хак» системы и, 

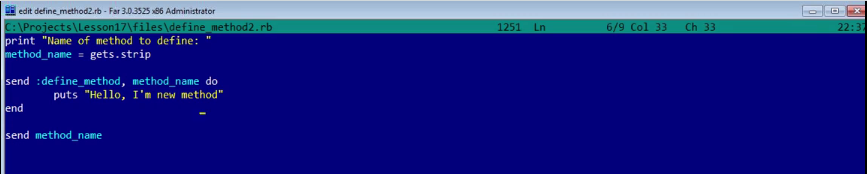
несмотря на то, что в классе Альбукерке нет методов cook, take\_a\_ride, die будет выведено следующее:



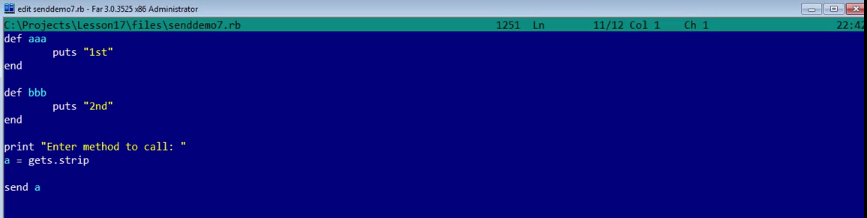
С помощью send можно динамически создавать новые методы даже во время выполнения программы:



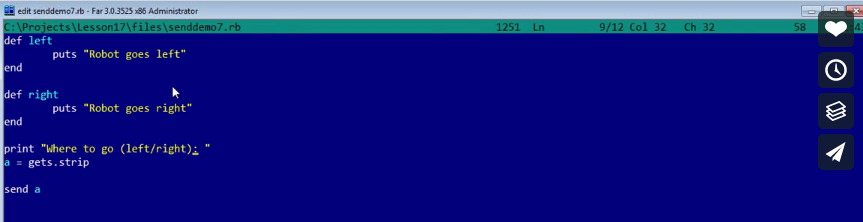
Def ааа, расположенный ниже send и блока do end делает то же самое, но статически. С помощью send можно создать новый метод с любым названием, например, полученным с клавиатуры, как в примере ниже…



С помощью send можно управлять развитием выполнения программы:



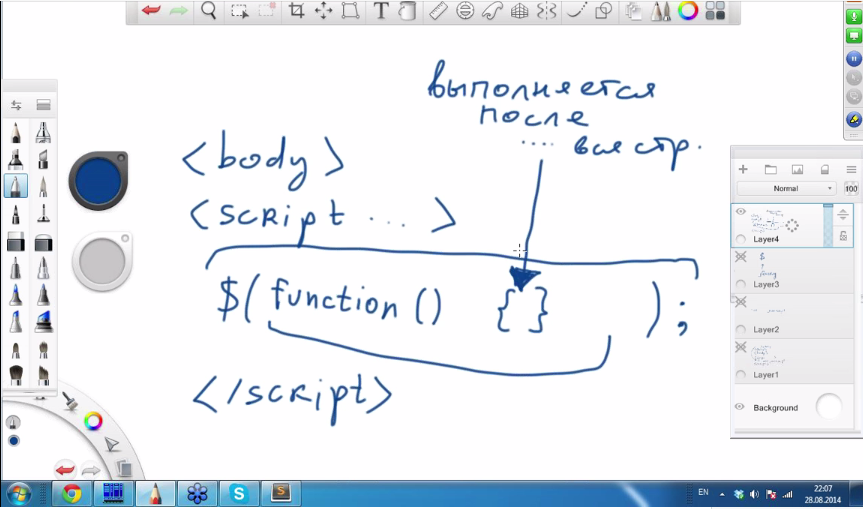
Запрашивая с клавиатуры имя метода, переключать программу на определенное направление. То же справедливо и для игр, например….



# Урок 24

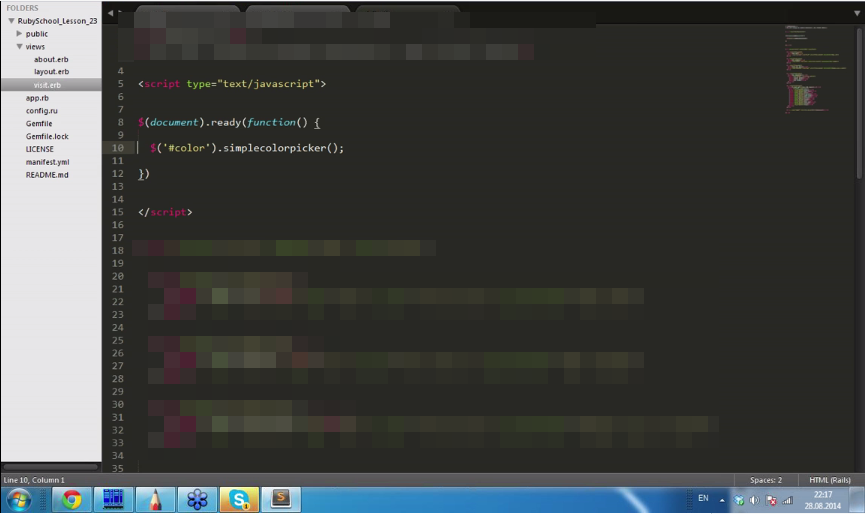
## Важный момент при написании кода в HTML странице

### <script>

Если писать код без применения нижеуказанной конструкции, помещая его в тегах <script> </script>, то код может быть исполнен раньше, чем браузером будут подгружены элементы управления, к которым код обращается. Это может повлечь некорректную работу страницы или просто ошибку.

Если же включать свой код в фигурные скобки после function(), то код будет выполнен после загрузки всех элементов страницы, что не вызовет ошибок и корректно отобразит страницу.

Ниже приведен равноценный эквивалент вышеуказанной конструкции .



# Урок 25. SQLite3

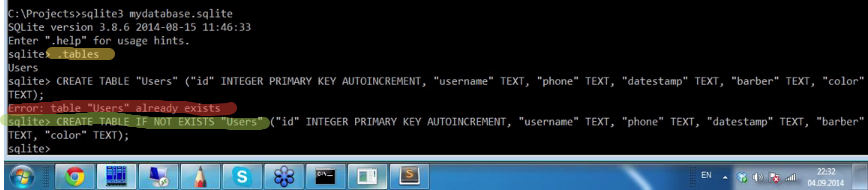
## Основные команды

### Просмотр существующих таблиц в базе данных

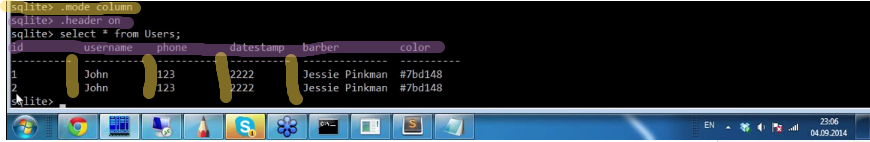
Просмотр существующих таблиц в базе данных из консоли осуществляется при помощи команды .tables. На картинке выделено желтым.

При попытке создать таблицу с существующим именем выдается ошибка – выделена красным.

Чтобы избежать подобного, необходимо добавить в команду IF NOT EXISTS. Выделено зеленым.

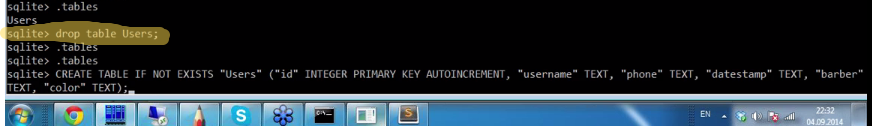


### Добавление заголовков в таблицу и вставка пробелов между столбцами



### Удаление таблицы

Удаление таблицы из консоли происходит командой drop. Выделено желтым



# Урок 26

## datetime

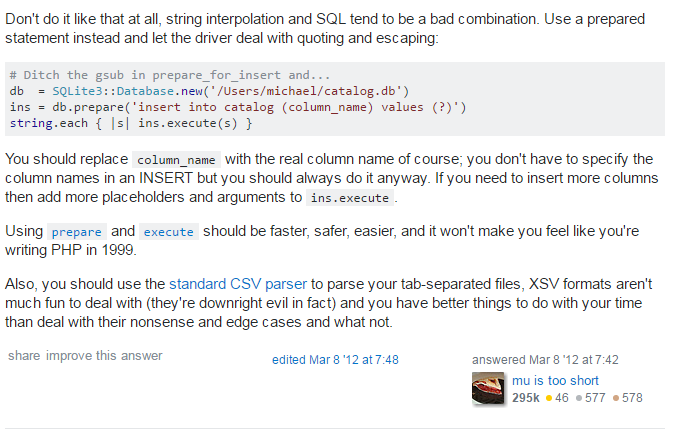
datetime является зарезервированным именем в SQL. То есть, для названия столбцов лучше выбирать другое имя. В RUBY принято писать date-stamp или datestamp или с использованием верблюжьего синтаксиса DateStamp.

## Ввод данных в таблицы SQL через RUBY

### Во избежание SQL injection

Во избежание SQL injection следует использовать следующий код. Взято по ссылке ниже.

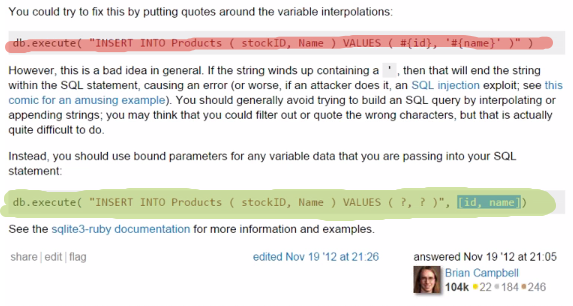
<http://stackoverflow.com/questions/9614236/escaping-strings-for-ruby-sqlite-insert>



### Следует избегать ввода в таблицы таким кодом

как выделено красным, отдавая предпочтение такому вводу, как выделено зеленым.

Взято здесь <http://stackoverflow.com/questions/13462112/inserting-ruby-string-into-sqlite>

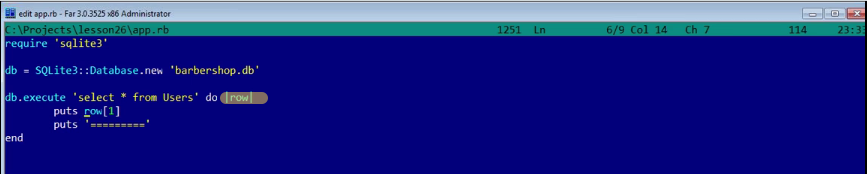


В зеленом варианте данные будут браться из массива [id, name] и подставляться в VALUES в том порядке, в каком они идут в массиве. При этом весь массив передается как одно значение. Подробнее об этом в документации по [SQLite3](http://www.rubydoc.info/github/luislavena/sqlite3-ruby)

Вопросительный знак в VALUES это фишка библиотеки .execute.

### Вывод строк базы данных с RUBY

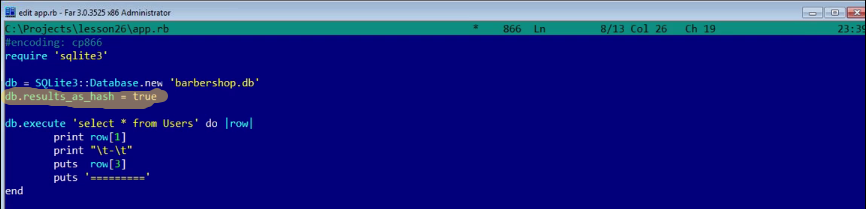
При выводе строк базы данных можно использовать следующий код…



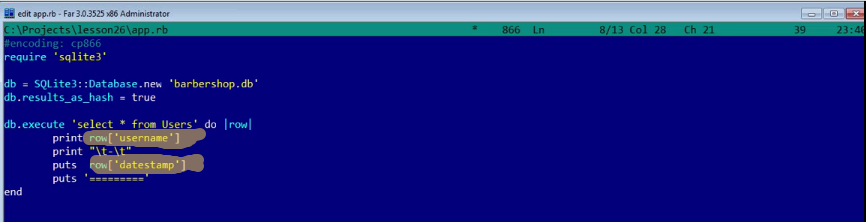
Желтым отмечена переменная, имеющая тип array. Таким образом, можно получать доступ к данным по индексу. Стандартный запрос SELECT \* FROM USERS равносилен запросу SELECT id, Name, Phone и тд из базы данных. Удобнее получать доступ по индексу, как указано выше.

Однако, глядя на такую программу тяжело понять, какие данные выводит программа. Row[1] неинформатино.

Удобнее будет получать вывод в виде хэша. SQLite3 позволяет это сделать. Для чего нужно добавить следующий код (помечен желтым)



После этого нужно немного изменить код вывода на следующий:



Таким образом, сразу понятно, какие именно данные выводит код, а хэш более удобен в работе.

# Урок 27