Real Web Development

yeah, for real.

who am i?

- i'm still cyle
- i'm a systems developer and architect
- every day i'm developin'
- i like this kind of stuff

real?

- kind of ranty, sorry
- web *development* is more than just...
- coding
- and/or designing

real development

- is about *making technologies work together*
- it's language agnostic

web design

- is about making pages *look pretty*
- and that's cool

web programming

- is about making pages *actually work*
- and that's needed, sure

web dev

- is doing both.
- usually not as well, though.
- but hey, at least it works!
- the one-person band!

pieces of development

- server building (i.e. LAMP)
- prototyping
  - building small pieces before the whole
- designing / wireframing
- coding / debugging

on a team

- you have an admin (does the building)
- you have a designer (makes the pretty)
- you have a coder (makes it work)
- probably two of each, at least

the one-person band

- pros
  - can get things done faster
  - has a complete view of the whole process
  - doesn’t have to have meetings
  - cheaper than a team

the one-person band

- cons
  - can write bad code easily
  - is a single point of failure
  - doesn't do any one piece as well as a dedicated person

anyway

- all the cool kids are developers
- so let's get into being a developer

pieces of "the stack"

- a server layer (Linux)
- a web service layer (Apache, lighttpd)
- sometimes a cache layer (memcached)
- an application layer (PHP, Rails, Python)
- a database layer (MySQL, MongoDB, Redis)

LAMP

- is a stack
- you've probably heard of it
- Linux, Apache, MySQL, PHP

the LAMP stack in action

1. user types in whatever.com
2. Apache gets the request, knows that the index file for that domain is *index.php*
3. Apache opens a PHP instance to parse the file
4. PHP interprets the script, then returns its output to Apache
    1. ...maybe it gets data from MySQL
5. Apache returns the output to the user's browser as HTML/CSS/whatever

- if you can be in control of *every step* of that process...

- you're a real web developer

Linux

- runs the show
- should've been here for the earlier session
- all you have to do is know how to edit a lot of configuration files
- and how to start/restart/reload services

Apache

- keeps its configuration files most often within...
- /etc/apache2
- any changes, you need to reload apache
- service apache2 reload
- (however, a full restart or force-reload makes sure everyone sees the changes)

PHP

- is a *procedural*, *interpreted* language
- *procedural* means it follows the script one line at a time
- *interpreted* means it parses that script every time it's asked
- line 5 won't happen until line 4 is done
- and it'll do all the lines of a script each time it's requested

MySQL

- it's a database! it holds data.
- in databases, tables, rows, columns
- accessed using SQL, or *Structured Query Language*
- which is its own language for accessing data, commonly used alongside other languages (like PHP)

please notice

- I didn't mention HTML and CSS and Javascript
- because you should already know those, really
- and they are client-side things, not server-side things
- HTML/CSS/Javascript is like basic algebra, you're learning trigonometry right now
- if you don't know them like the back of your hand, you should learn them like the back of your hand

inside “development”

- when editing a PHP file, you probably see...
- PHP code
- SQL queries
- HTML / CSS / Javascript
- it’s messy, but that’s how you start

HTML, CSS, Javascript, PHP, and SQL

```
<!doctype html>
<html>
<head>
<title>oh dear</title>
<script type=”text/javascript”>
alert(‘welp.’);
</script>
<style type=”text/css”>
body { color: red; background-color: black; }
</style>
</head>
<body>
<?php

$mysql = new mysqli(‘localhost’, ‘username’, ‘password’, ‘some_db’);
$get_something = $mysqli->query(‘SELECT title FROM site_stuff WHERE id=1’);
if ($get_something->num_rows > 0) {
   $info = $get_something->fetch_assoc();
   echo ‘<h1>’.$info[‘title’].’</h1>’;
}

?>
</body>
</html>
```

Basic PHP Primer

- because Apache works pretty easily out of the box, we'll get to that later
- and MySQL needs to be built according to the app's needs, so we'll get to that later

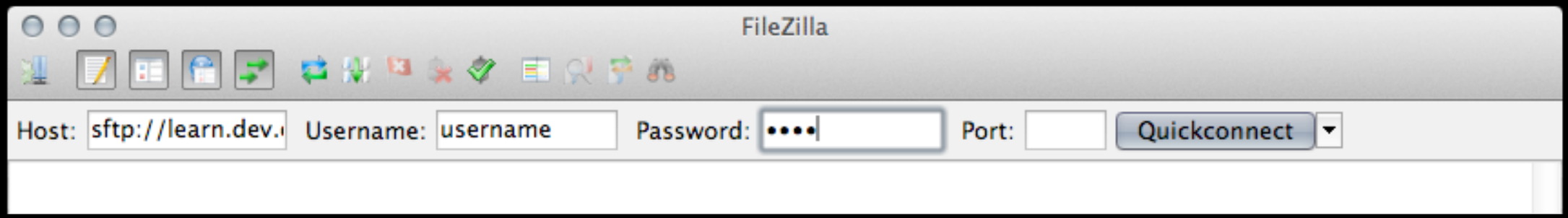
what you need

- a plain text editor
- somewhere that'll parse the PHP files
- luckily we have both!

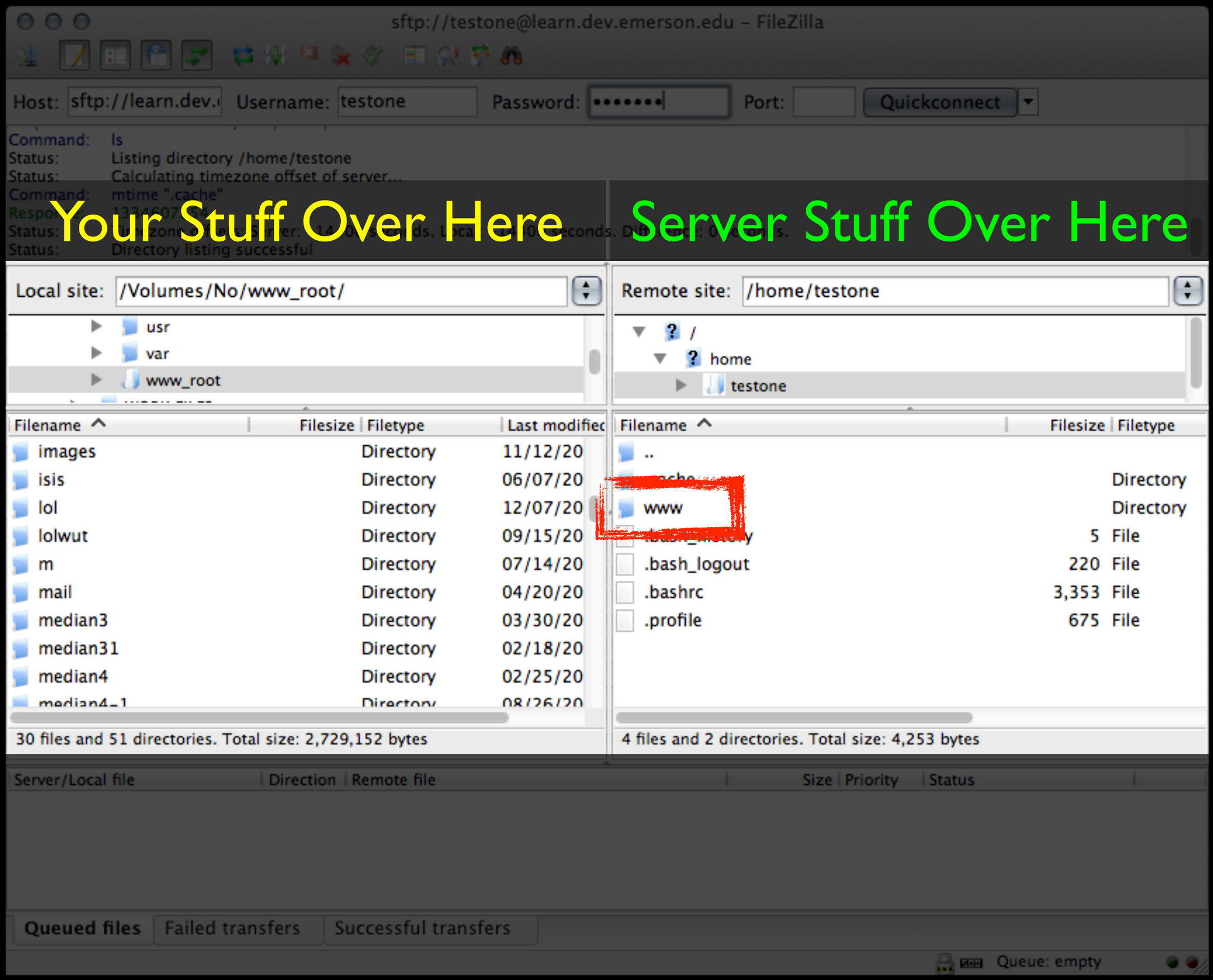
running stuff

- you script PHP in a plain text file
  - on these machines, use Kod or TextEdit
- then upload it to a server via FTP/SFTP
  - using FileZilla, for our purposes
- then visit that page in a browser

so open FileZilla...



Host: sftp://learn.dev.emerson.edu
Username: the one we set up
Password: learnit


sftp://testone@learn.dev.emerson.edu – FileZilla
Host: sftp://learn.dev.
Username: testone
Password: •••••••
Port:
Quickconnect
Command: ls
Status: Listing directory /home/testone
Status: Calculating timezone offset of server...
Command: mtime ".cache"
Status: Directory listing successful
Your Stuff Over Here
Server Stuff Over Here
Local site: /Volumes/No/www_root/
usr
var
www_root
Filename
Filesize
Filetype
Last modifiec
images Directory 11/12/20
isis Directory 06/07/20
lol Directory 12/07/20
lolwut Directory 09/15/20
m Directory 07/14/20
mail Directory 04/20/20
median3 Directory 03/30/20
median31 Directory 02/18/20
median4 Directory 02/25/20
30 files and 51 directories. Total size: 2,729,152 bytes
Remote site: /home/testone
/
home
testone
Filename
Filesize
Filetype
..
Directory
www
Directory
5 File
.bash_logout 220 File
.bashrc 3,353 File
.profile 675 File
4 files and 2 directories. Total size: 4,253 bytes
Server/Local file
Direction
Remote file
Size
Priority
Status
Queued files
Failed transfers
Successful transfers
Queue: empty

- things in your www folder show up here
- http://learn.dev.emerson.edu/~user/
- so an what.php shows up at
- http://learn.dev.emerson.edu/~user/what.php

- that's you uploading a file via SFTP
- going to a URL in your browser
- Apache parsing the request
- PHP interpreting the file
- and Apache returning your browser the result

ok

- now open that text editor...

hello_world.php

```
<?php

echo 'HELLO WORLD!';

?>
```

REMEMBER

a *computer* is reading what you're writing.

this is important.

<?php ... ?>

this tells the computer that
everything *in between* is PHP!

use a semicolon at the end of every line.

or your script won't work

expressions

- an expression returns something!
- 4 + 10
- that's an expression, it returns 14
- the number (a type of data), 4
- the addition operation (a function), +
- the number (more data), 10

data types, variables

- numbers
  - integers, i.e. whole numbers, 2928
  - floating-points, i.e. decimals, 3.14
- strings
  - text between ‘single quotes’
  - or “double quotes”

- booleans, true or false
- arrays of things, i.e.
  - array(“apple”,“banana”,“grape”);

- variables to store these!

```
<?php

$what = “hello!”;
echo $what;

?>
```

- the $ before a word denotes a variable
- the = sign turns the result of an expression into a variable!
- so $variable = 4 + 10;
- $variable will be 14

a variable, storing an array of strings

```
<?php

$fruits = array(“apple”, “banana”);

print_r($fruits);

?>
```

- arrays hold a lot of data, potentially
- you access the info inside it with indexes
- $fruit[0] is the first string in the list, so “apple”
- $fruit[1] is the second, so “banana”
- the number inside [ ] is the index, starts at 0
- you could use count() to find out how many elements are in an array

- print_r() and count() are *functions built into PHP*
- like an expression, functions *return something* (usually)
- in this case, print_r() takes whatever you give it and just prints out what's inside
- it'll show the contents of the $fruit array
- count() takes an array and returns the number of elements inside

- there are a lot of functions
- http://gotapi.com/php/
- and you can make your own!

protip

- *learning to read documentation* is the second greatest skill you can have
- the first greatest is being able to troubleshoot problems with deductive logic, usually known as "debugging"

- nine times out of ten, what you want to do has already been done

- and it's covered in some documentation somewhere...

- google it.

```php
<?php

function addition($first, $second) {
  return $first + $second;
}

echo addition(4, 10);

?>
```

congrats, a function that replicates existing functionality

(don’t ever do that, really)

don’t be afraid

- people get scared when it comes to ( )s and { }s and [ ]s and whatnot
- don’t be scared, it’s just separating parts of the script so the computer can understand it better
- could be worse, could be LISP.

oh jeez, the parentheses!!

```
(defun factorial (n)
   (if (<= n 1)
       1
       (* n (factorial (- n 1)))))
```

gonna give me a heart attack.

anyway...

- that's expressions, data types, variables, functions, reference docs...
- now for some control

if, else

```php
<?php

if (4 + 10 > 13) {
  echo “14 is definitely greater than 13”;
} else {
  echo “you will never see this.”;
}

?>
```

operators!

- >, <, <=, =>
  - greater than, less than, etc
- == and !=
  - if the same, if NOT the same
- NOT a single = this isn't math
- use DOUBLE == to signify "if the same"

```php
<?php

$variable = true;

if ($variable == true) {
  echo “weee!”;
}

if ($variable >= 1) {
  echo “weeeee!”;
}

if ($variable) {
  echo “weeeeeeee!”;
}

?>
```

more than one

- if you want to evaluate more than one condition in an IF statement...
- use || and &&
- || means “or”
- && means “and”

```php
<?php

$variable = 10;

if ($variable > 5 && $variable < 15) {
  echo “you’ll see this.”;
}

if ($variable == 10 || $variable == 11) {
  echo “you’ll also see this.”;
}

?>
```

for

```php
<?php

for ($i = 0; $i < 10; $i++) {
  echo “loop iteration $i \n”;
}

?>
```

```
($i = 0; $i < 10; $i++)
```

- how to start the loop!
- a variable, $i, with a value of 0

```
($i = 0; $i < 10; $i++)
```

- as long as this is going on, keep looping.
- when this is *false*, stop looping.
- so...
  - as long as $i is less than 10, keep going.

```
($i = 0; $i < 10; $i++)
```

- what to do at the end of each iteration
- it does everything between the { }, and then this
- so every time, increment $i by one
- (that's what ++ does, it's an operator)

foreach

```php
<?php

$fruits = array(“apple”, “orange”);

foreach ($fruits as $fruit) {
  echo ‘FRUIT: ’.$fruit.’<br />’;
}

?>
```

- foreach takes an array
- goes through everything inside of it
- one at a time

while

```php
<?php

$variable = 5;

while ($variable < 10) {
  echo $variable;
  $variable++;
}

?>
```

- while keeps doing whatever is in the block
- the “block” is denoted by the curly braces { }
- while the condition is still true
- be careful, you can accidentally make something loop *forever*
- *and that’s not cool.*

that's all i'm gonna teach you

- about PHP anyway
- the rest is just... coding. a lot.
- reading documentation, learning functions.

anyway, a MySQL primer

- MySQL has databases
- each database can have tables
- each table has columns
- each table has rows with info for those columns
- each table typically has an index

- typically you want a database for each app
- or a database for info to be used between apps
- common idea: “users” might have its own database if it’s gonna be shared,
- or you have a “users” table for each thing you build

- a basic database is a lot like an excel spreadsheet
- (and by that i mean *gross*)

the “users” table inside a “cool_app” database

first_name	*last_name*	*email*	*id*
cyle	gage	lol@whatever.com	1
frankie	frain	frankie@whatever.com	2

- an index helps distinguish unique rows
- columns separate data
- i mean, yeah, it's a spreadsheet, pretty much
- it has data types just like PHP
  - strings, numbers, etc
  - used to define columns

SQL

- getting to that data!

```
SELECT * FROM users WHERE id=1;
```

- oh jeepers

```
SELECT * FROM users WHERE id=1;
```

- SQL is made up of KEYWORDS
- SELECT means.... select
- there are other words to use, like
- UPDATE and DELETE and INSERT

```
SELECT * FROM users WHERE id=1;
```

- the asterisk means return *all columns*
- here, you could tell it to just return specific columns, if you wanted

```
SELECT * FROM users WHERE id=1;
```

- from... what table?
- oh, yeah, the *users* table

```
SELECT * FROM users WHERE id=1;
```

- where... the following condition(s) are met
- so, where the id column equals one

```
SELECT * FROM users WHERE id=1;
```

- DON'T FORGET THAT SEMICOLON!
- this is a running theme
- however, you really only need this if you are inside the MySQL shell

a practical example

```php
<?php

$mysql = new mysqli('localhost', 'user', 'password', 'db_name');

$get_users = $mysql->query("SELECT * FROM users");

while ($row = $get_users->fetch_array()) {
  echo "a row was returned!";
  print_r($row);
}

?>
```

- PHP connects to the database
- then PHP queries the database
- then PHP does something with each row that returns

more SQL

```
INSERT INTO users (first_name, last_name)
VALUES (‘george’, ‘takei’);
```

```
UPDATE users SET name=‘tom’ WHERE id=1;
```

```
DELETE FROM users WHERE id=1;
```

MySQL admin

- direct via the shell on the server
- or phpMyAdmin

easy projects

- a Twitter clone
- a blog, or forum
- really... clone what's already out there

a note about PHP

- PHP is not a good language
- it is not *elegant*, it is not *advanced*
- but it's used everywhere
- and it can do a lot
- it's a good start
- *...but at least it's not java.*

things to be concerned about

- web standards
  - adhere to them
    - nobody should care about IE6
- common best practices
  - jQuery, MVC, clean code, commenting
- getting the process out of the way
  - so you can get to the developing part

things to read up on

- regular expressions
- node.js (server-side javascript)
- NoSQL (MongoDB, Redis)
- other “web” languages!
  - Ruby, Python
- non “web” languages!
  - C, Obj-C, Erlang, Scala

things to take away from this

- try everything. learn stuff.
- ultimately, you're writing for a computer.
- learn how to read documentation.

development tools

- All-in-ones:
  - Coda, Aptana, Espresso, Eclipse
- Text editors (with code in mind)
  - TextMate, Kod, TextWrangler
- File transferring:
  - FileZilla, Cyberduck

build something,
do something,
learn

bonus time?

- Apache config demo
- lighttpd demo ?
- mongodb demo ?

more learning

- http://arepository.com/of/learning/
- email me
  - cyle_gage@emerson.edu