Quiz 1 Review Session

Part 0: File I/O, Data Structures, and HTML

File I/O

- Review the important functions (what arguments they take, what they return, and when they are used)
  - fopen
  - fclose
  - fread
  - fwrite
  - fseek
- Don't forget…
  - Always check to make sure fopen doesn't return NULL
  - Always close a file after opening it

Data Structures

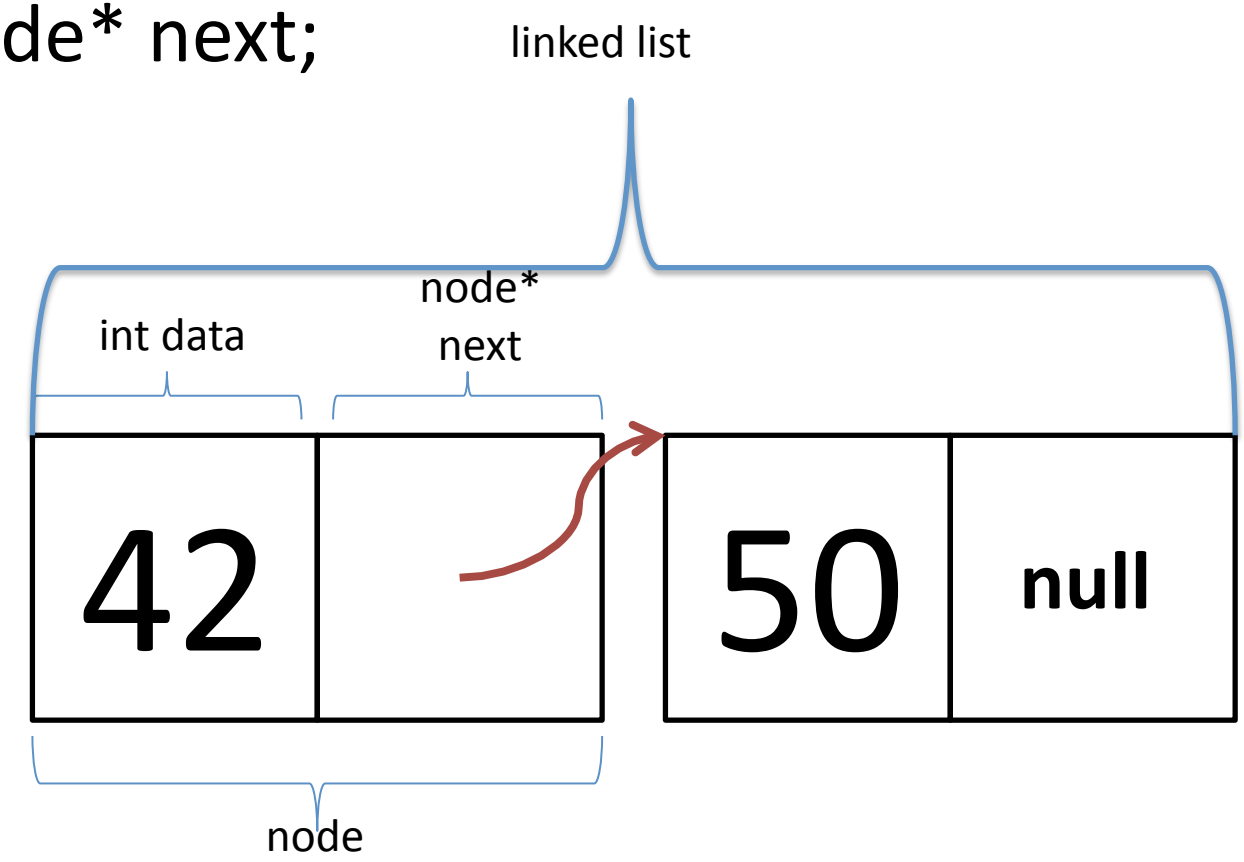
- Abstract, high-level
- Need to know
  - Conceptual descriptions
  - Common operations
  - How to implement in C
    - Particularly linked lists and hash tables
- Review pointers and structs!

Data Structures

- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

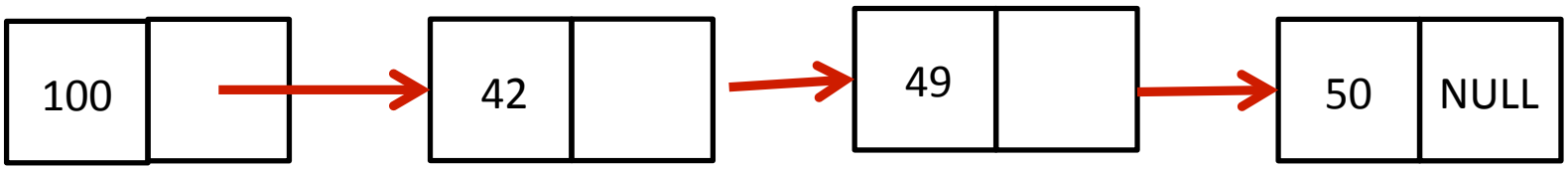
Linked Lists

- typedef struct node {
  int data;
  struct node* next;
  } node;



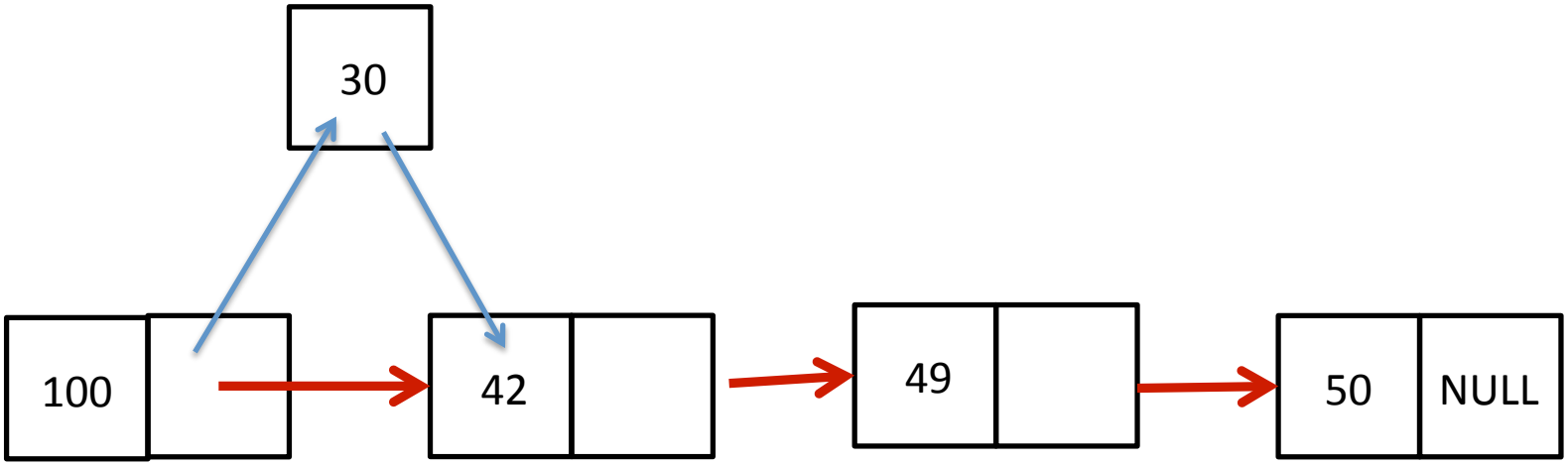
Linked Lists

- Unlike an array, linked lists allow us to insert or remove elements in constant time!
  - As long as we don't need the list to be sorted



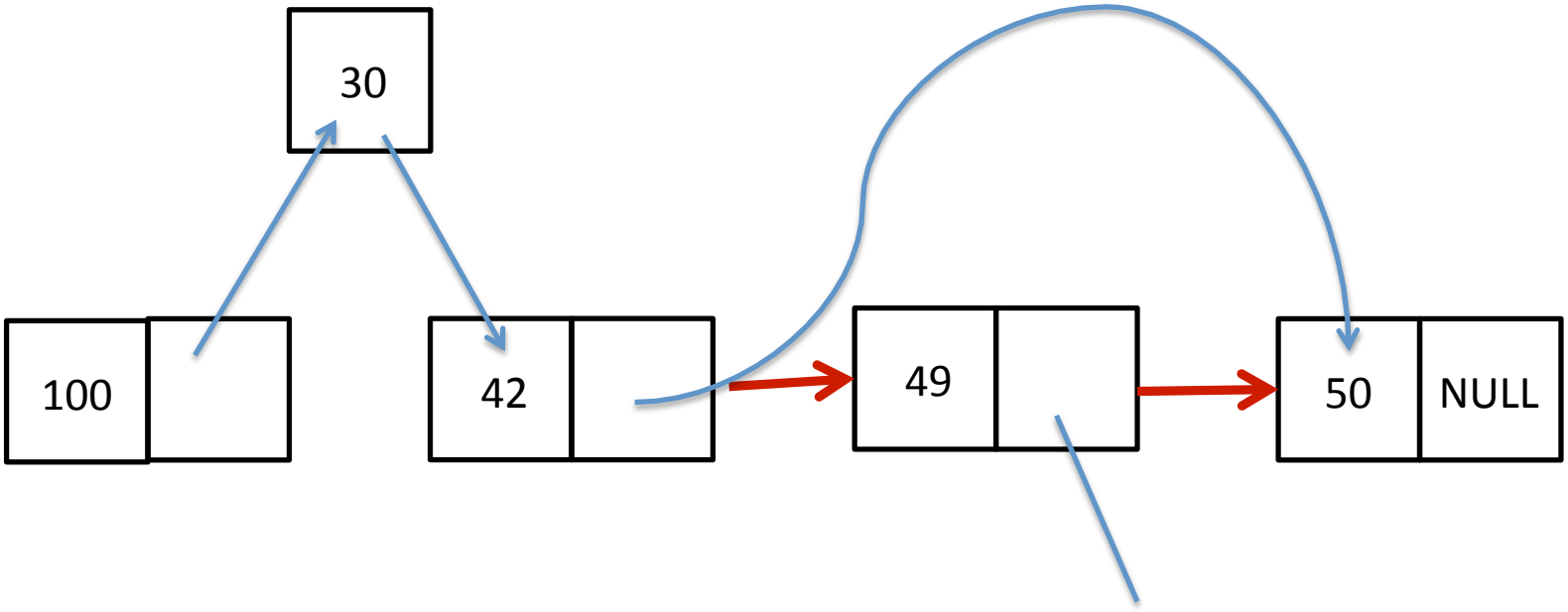
Linked Lists

- Unlike an array, linked lists allow us to insert or remove elements in constant time!
- Insert 30?



Linked Lists

- Unlike an array, linked lists allow us to insert or remove elements in constant time!
- Remove 49?



Don’t forget to free() this pointer after you remove the node

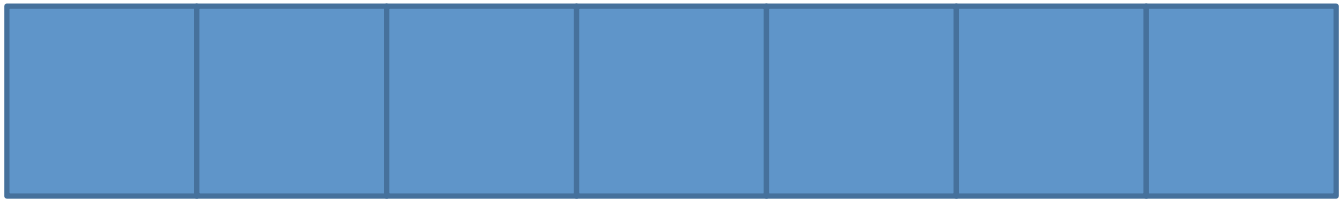
Doubly linked lists have nodes with pointers to both the next and previous node

Data Structures

- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

Queues

- First in, first out data structure = FIFO
- "Insert" and "Remove" operations.
  - aka "enqueue" and "dequeue"
  - Insert into the tail, remove from the head



Head **Tail**

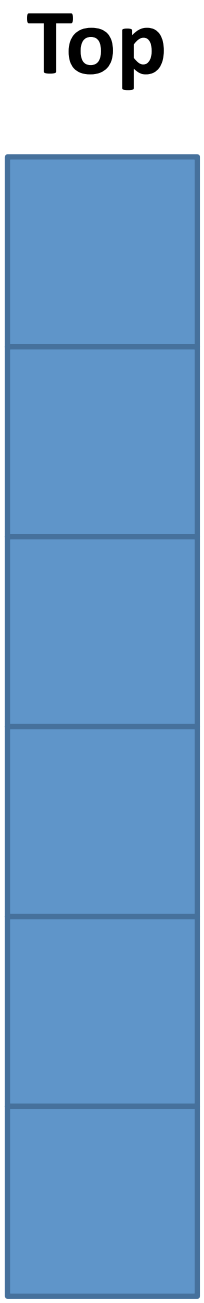
Queues

Data Structures

- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

Stacks

- Last in, first out data structure
  - LIFO
- Can ‘push’ elements to the top of the stack and ‘pop’ elements off
- A stack can only access its top element!



Stacks ‘n’ Queues

- How can we implement these data structure using linked lists?
  - Stack: add to beginning of list; remove from beginning of list
  - Queue: add to end of list (by traversing nodes until we find a NULL pointer) and remove from beginning of list

Data Structures

- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

Hash Tables

- An array of linked lists
- A hash function maps input to an index in the array.
  - Good hash functions are deterministic and well-distributed
- Collisions occur when the hash function maps two inputs to the same index in the array
  - In this case, we “chain” the newest input onto the linked list located in that index

Data Structures

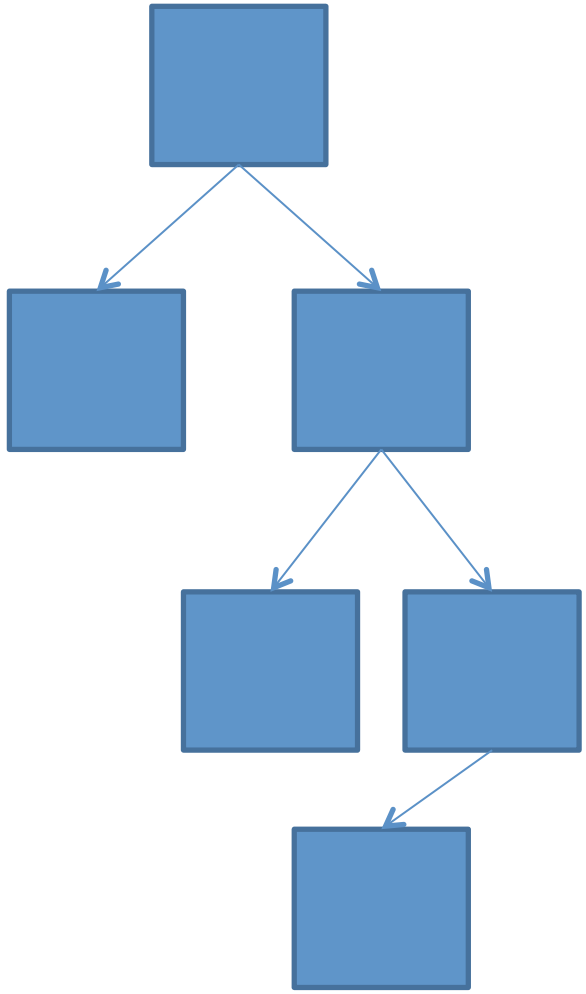
- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

Trees

- A way of organizing elements such that each element has a parent and a series of children
- Binary Tree
  - Each node has up to two children
- Binary Search Tree
  - Each node has up to two children AND nodes are arranged in order from left to right

Binary Search Tree

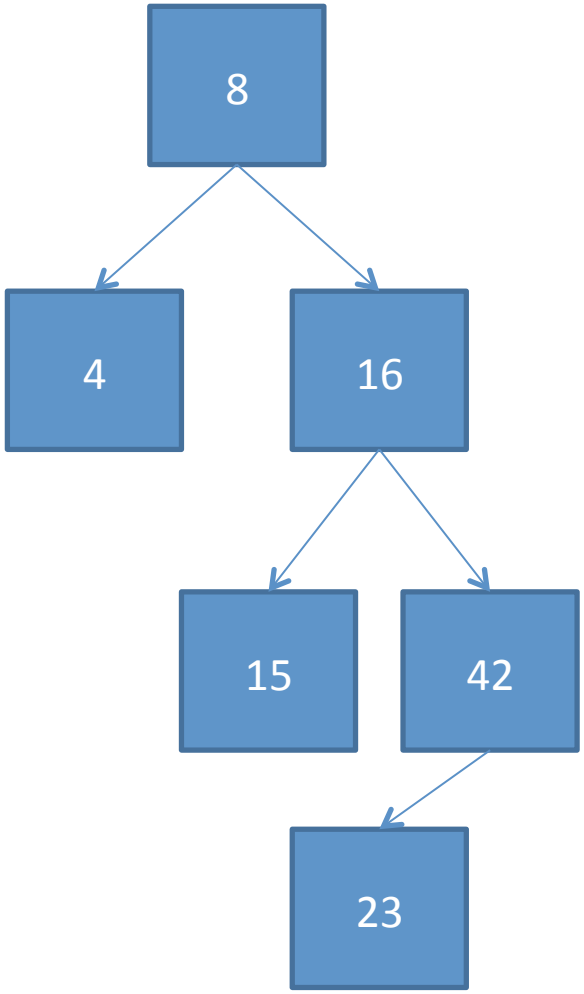
```
typedef struct tree_node
{
    struct tree_node* left;
    int val;
    struct tree_node* right;
} branch;
```



Binary Search Tree

BST is such that:

1) Left subtree of each node contains only lesser nodes.
2) Right subtree of each node contains only greater nodes.
3) Left and right subtrees of each node are also binary search trees: **recursive data structure**



Binary Search Tree

- How might we find a value in a BST?
- Check value of node
  - If value is greater than input, search for input in **left branch**
  - If value is less than input, search from input in **right branch**
  - If value is equal to input, return true!
  - If we find a NULL branch, return false!

Data Structures

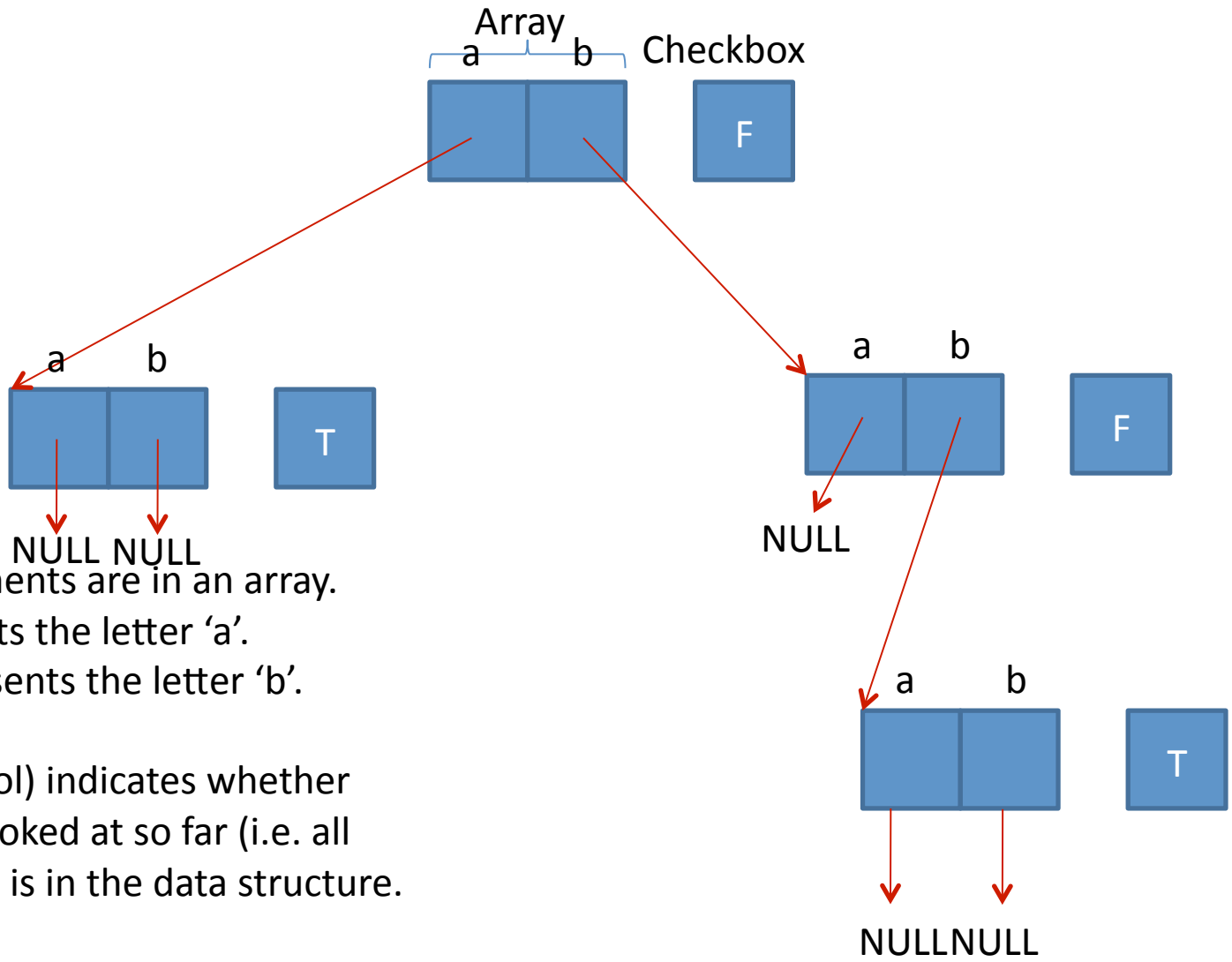
- Linked Lists
- Queues
- Stacks
- Hash Tables
- Trees
- Tries

Tries

- Tree of Arrays
- Fast Lookup, High Memory Use

```
typedef struct trie_node
{
   struct trie_node* array[N];
   bool checkbox;
} trie_node;
```

Tries



First two elements are in an array.
First represents the letter ‘a’.
Second represents the letter ‘b’.

Checkbox (bool) indicates whether what we’ve looked at so far (i.e. all parent nodes) is in the data structure.

“a”, “bb” are in this structure.

When might a trie be useful?

- Instant search for filtering contacts on your phone?
- Words in a dictionary?

HTML

- Review the basic structure of an HTML page
- Important tags
  - <head>
  - <body>
  -
  - <img>
  - <a>
  - <form>
  - <style>
  - <link>
  - <script>

HTML

-
  - img is the tag name; src, alt, and id are the attributes
- Hello world!
  - "Hello world!" is the inner HTML

CSS

- Cascading Style Sheets
- 3 ways to specify style
  - In a separate .css file, linked to your page with a <link> tag in the head
  - In a <style> tag in the head
  - In the style attribute of any individual tag
    - e.g.
- These three methods have a set order of precedence

Good luck!

CS50 Quiz 1

This is CS50. (Quiz 1 Review)

o hai!

Joseph Ong

CS50: Quiz 1

this is a cat.

CS50: Quiz 1

PHP

PHP: Hypertext Preprocessor

- What it is:
  - Meta
  - Server-side scripting language
  - Let’s us develop the backend, or logical underpinnings, of our website.

Some Syntax

- `<?php` or `<?` marks the beginning of PHP code, `?>` marks the end.

- All the variables start with $. $ ALL THE VARIABLES.
  - "*Loosely typed.*" **Does not mean no types.**
  - You can freely switch and compare variables between types.

```
<?
    $number = 1;
    $number2 = "1";
    $number3 = 1.0;

    if($number == $number3)
        echo "Will be echoed";

    if($number === $number3)
        echo "Won't be echoed";
?>
```

== checks across types

=== is strict equality, type must match too.

String Concatenation

- Done using the . operator

```
<?
    $courseName = "CS50";
    $title = "Walkthrough Boy";

    $tommy = $courseName . " " . $title;

    echo $tommy;
?>
```

Tommy

Loops & Conditionals

- Same looping constructs and conditionals as C.
  - if, for, while, switch (unlike C, can switch with strings), ...

```
<?
    for($i = 0; $i <= 8; $i++)
       doProblemSet($i);

    print("DID ALL THE PROBLEM SETS!");
?>
```

Function Declarations

- `function` keyword to declare a function.
  - Functions don't specify a return type, since PHP has weak types!

```
<?
    function doProblemSet($problemSetNumber)
    {
        slaveAway();
        // prints to screen based on whether pset completed
        if(triumphant())
          print("Did Problem Set " . $problemSetNumber . "!");
        else
          print("ummmmm... oops.");
    }

    function triumphant()
    {
        return false;
    }
?>
```

Arrays

- Can work like normal arrays.

```
<?
    $number = array();

    // pushes 1, then 2, then 3 into the array
    $number[] = 1;
    $number[] = 2;
    $number[] = 3;

    // changes things at locations 1 and 2
    $number[1] = 6;
    $number[2] = 4;

    // prints out 164
    echo $number[0] . $number[1] . $number[2];
?>
```

Arrays

- You can also mix types.

```
<?
    $number = array();

    // pushes 1, then "fish", into the array.
    $number[] = 1;
    $number[] = "fish";

    // prints out 1fish
    echo $number[0] . $number[1];
?>
```

Arrays

- Can also use strings as "**keys**" into an array (aka indices).
  - The term for a string-indexable array is an **associative array.**

```
<?
    $tf = array();

    $tf["name"] = "Joseph";
    $tf["course"] = "CS50";
    $tf["likes"] = array("cake", "cats");

    // alternatively
    $tf = array( "name" => "Joseph Ong",
                 "course" => "CS50",
                 "likes" => array("cake", "cats")
               );
?>
```

PHP Classes

- Similar to a struct in C, but can contain functions as well. Simply put, a blueprint for creating similar variables.

```
<?
    class Stock
    {
        public $symbol = NULL;
        public $name = NULL;
        public $price = NULL;

        public convert($exchangeRate) {
            return $price * $exchangeRate;
        }
    }

    // create an instance of Stock, -> to access field of instance
    $stock = new Stock();
    $stock->symbol = "GOOG";
    $stock->name = "Google";
    $stock->price = 630.95;

    // convert
    echo $stock->convert(.75);
?>
```

Some Statements & Functions

- `require_once(pathtofile)` statement includes PHP code from the the specified file and evaluates it. Often used to make libraries, etc.
- `echo` does the same thing as `print`.
- `exit` stops the further execution of any code.
- `empty` checks if a variable is empty. These are considered empty:
  - `""` `0` `0.0` `"0"` `NULL` `FALSE` `[]` uninitialized variable

includes/common.php

```
<?
     function apologize($message)
     {
          require_once("apology.php");
          exit;
     }
?>
```

login.php

```
<?
     require_once("includes/common.php");

     if(empty($_GET["username"]))
     {
          apologize("Type in a username!");
     }

     echo "Not executed on empty username.";
?>
```

ONLY GLOBAL SCOPE. NOOOOOO.

(with one exception: **functions**).

```
<?
    function aFunction($i)
    {
        $i++;
        echo $i;
    }

    for($i = 0; $i < 3; $i++)
      echo $i;

    aFunction($i);

    echo $i;
 ?>
```

```
0
```

ONLY GLOBAL SCOPE. NOOOOOO.

(with one exception: **functions**).

```
<?
    function aFunction($i)
    {
        $i++;
        echo $i;
    }

    for($i = 0; $i < 3; $i++)
      echo $i;

    aFunction($i);

    echo $i;
?>
```

```
0
1
```

ONLY GLOBAL SCOPE. NOOOOOO.

(with one exception: **functions**).

```
<?
    function aFunction($i)
    {
        $i++;
        echo $i;
    }

    for($i = 0; $i < 3; $i++)
      echo $i;

    aFunction($i);

    echo $i;
?>
```

```
0
1
2
```

ONLY GLOBAL SCOPE. NOOOOOO.

(with one exception: **functions**).

```
<?
    function aFunction($i)
    {
        $i++;
        echo $i;
    }

    for($i = 0; $i < 3; $i++)
      echo $i;

    aFunction($i);

    echo $i;
?>
```

This is local to the function.

When $i is 3, we exit the loop.

This is not local to the loop.

```
0
1
2
4
```

ONLY GLOBAL SCOPE. NOOOOOO.

(with one exception: **functions**).

```
<?
    function aFunction($i)
    {
        $i++;
        echo $i;
    }

    for($i = 0; $i < 3; $i++)
      echo $i;

    aFunction($i);

    echo $i;
?>
```

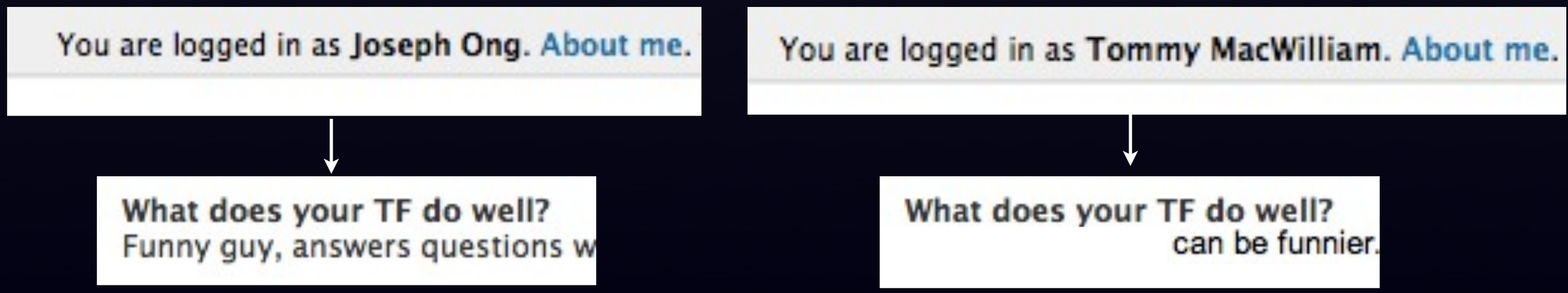
This is not local to the loop.

```
0
1
2
4
3
```

PHP and HTML

- PHP is used to make web pages dynamic.
  - With just HTML we serve the same static page to all users.
  - PHP gives us the power to alter the page's HTML prior to loading, based on the users actions, who they are, logic we've written up, etc.

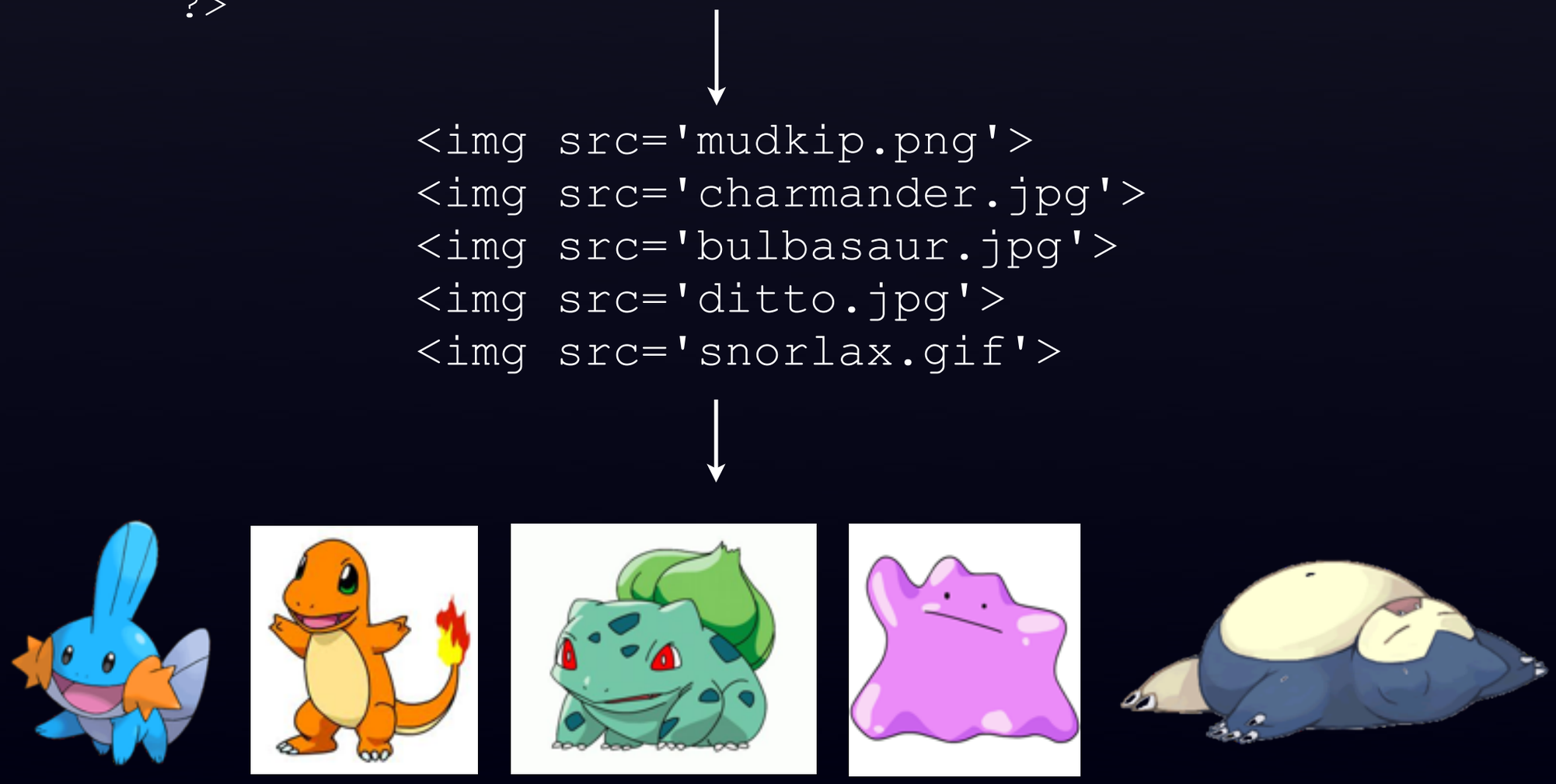
```
<?= "You are logged in as " . $name ?>
```



Generating HTML

- We've see it used together with HTML, where we used it to generate HTML.

```
<?
  for($i = 0; $i < 5; $i++)
    print("<img src='{$pokemon[$i]}'>");
?>
```



Generating HTML, again

- To help make the separation between HTML and PHP clearer, we have alternative control flow syntax.

```
<? for($i = 0; i < 5; i++): ?>
    <img src="<?= pokemon[$i] ?>">
<? endfor; ?>
```

↓

```
<img src='mudkip.png'>
<img src='charmander.jpg'>
<img src='bulbasaur.jpg'>
<img src='ditto.jpg'>
<img src='snorlax.gif'>
```

↓

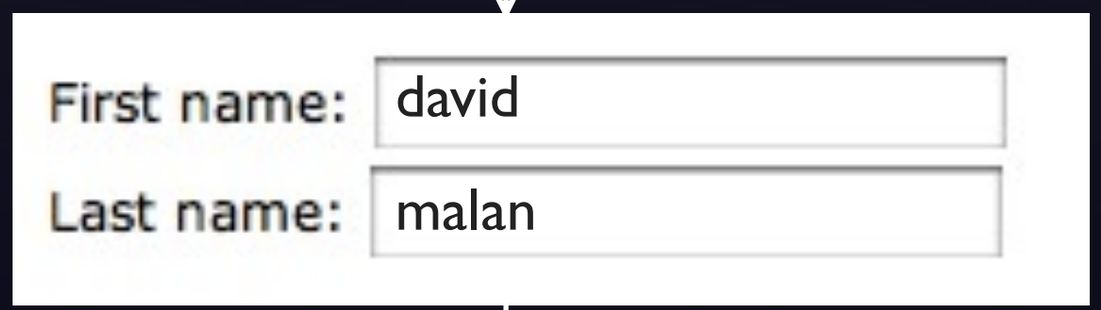
Forms and Requests

- We can pass data from HTML forms to PHP files.
- If we're using a form, then
  - `action` attribute tells us where to send the data.
  - `method` attribute tells us how to send the data (GET or POST)

- We can also make a GET request by constructing a URL directly.
  - `?paramname=value`, like below
  - `http://www.youtube.com/watch?v=oHg5SJYRHA0&value=4`

GET request

```
<form action="printName.php" method="get">
  First Name: <input type="text" name="firstname"><br>
  Last Name: <input type="text" name="lastname">
</form>
```



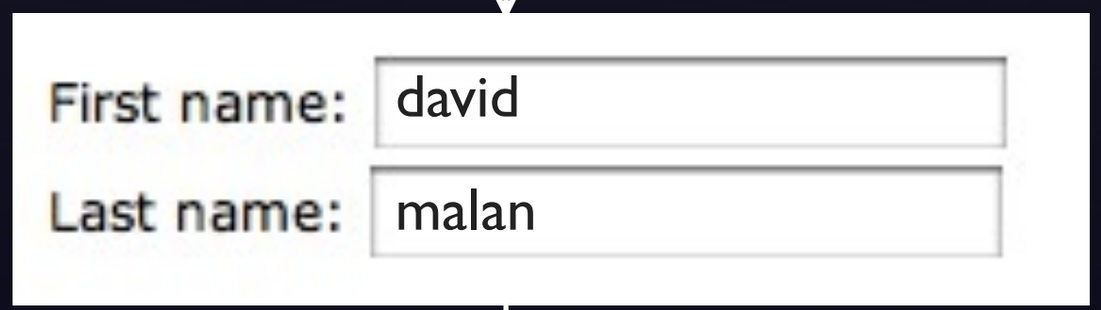

```
<?
    echo $_GET["firstname"];
    echo $_GET["lastname"];
?>
```

$_GET array
(indexed by name attribute)

sent in URL

POST request

```
<form action="printName.php" method="post">
  First Name: <input type="text" name="firstname"><br>
  Last Name: <input type="text" name="lastname">
</form>
```




```
<?
    echo $_POST["firstname"];
    echo $_POST["lastname"];
?>
```

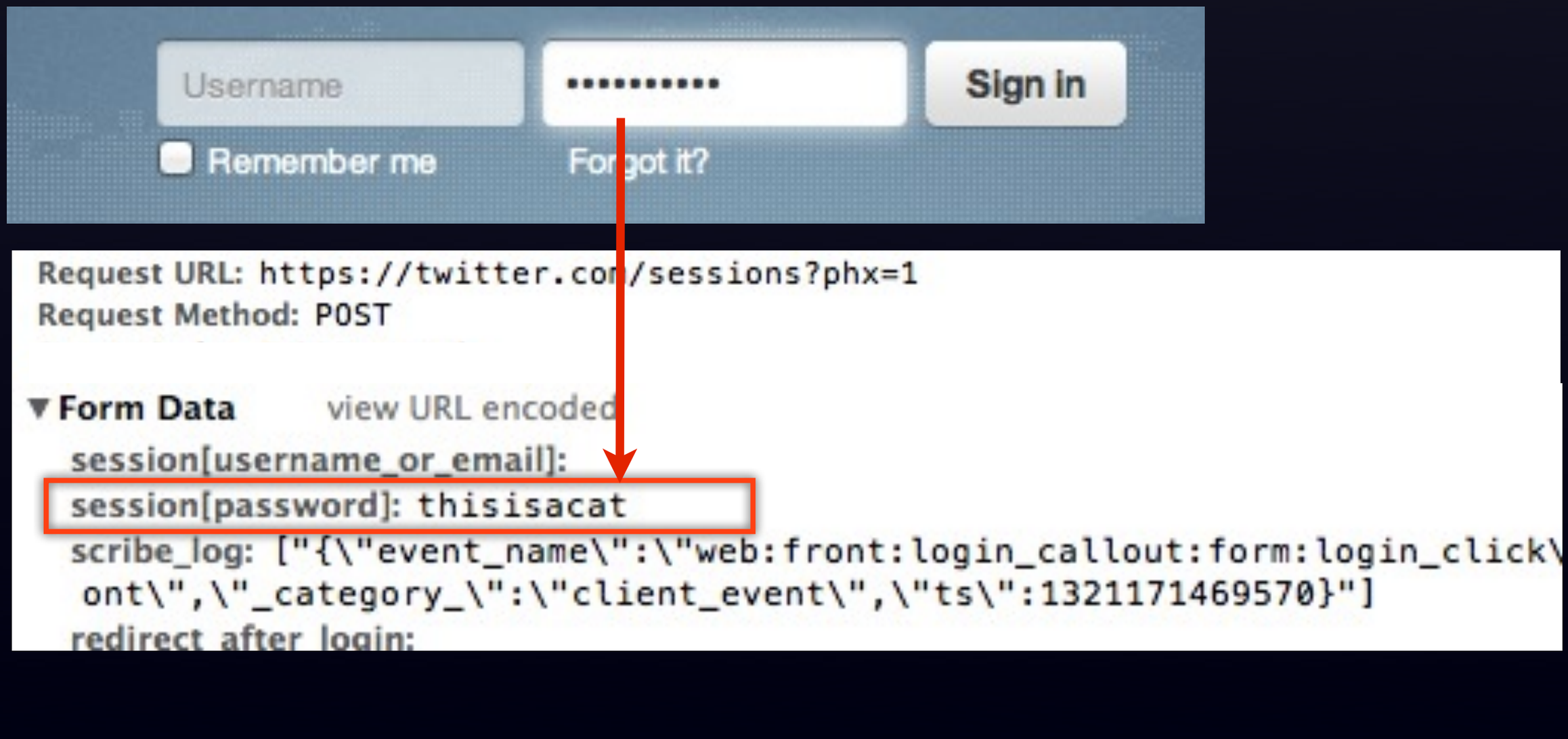
$_POST array
(also indexed by name attribute)

not sent in URL

POST and GET, equally insecure!

- It's still sent in plaintext, regardless. One just shows up in the URL, while the other doesn't.

  Don't get lulled into a false sense of security!



$_SESSION

- Used to store information about the current HTTP session.

```
$uid = $_SESSION["id"];
```

this is a cat

d'aaaaaaaaaaw

CS50: Quiz 1

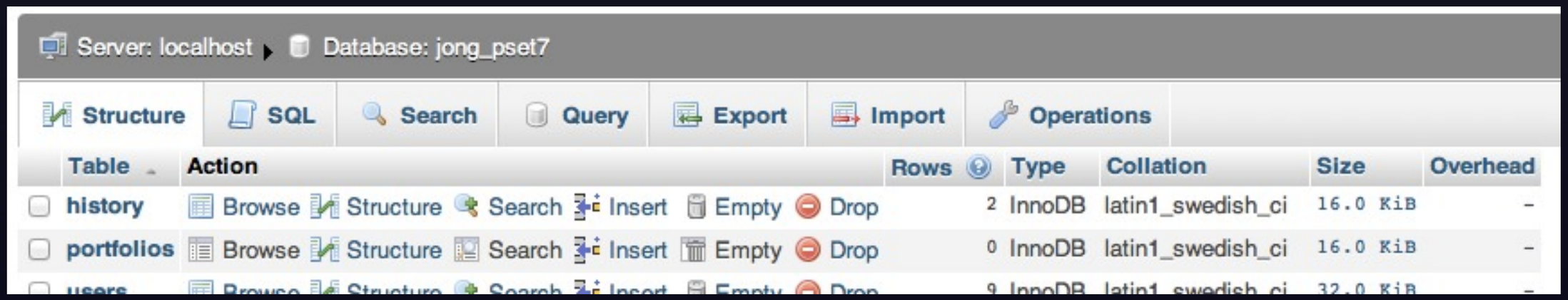
SQL

Structured Query Language

- What it is:
  - Not Meta.
  - Officially pronounced "s-q-l".
    - but everyone says "see-kwel".
      - except me : (. I try but it's so hard.
  - Programming language designed for managing databases.

Database? Whazzat.

- Not a cat.
- A database is a collection of tables.



Each table represents a collection of similar objects.
For example, a table of users:

←T→	id	username	hash	cash
Edit Copy Delete	1	caesar	1Y01fprd3$BA4jQZMm2rmb46EgU7RwN/	10000.0000
Edit Copy Delete	2	chartier	1NhaqO3f8$g4zPyTt2KSKdD7HnMl/nK0	10000.0000
Edit Copy Delete	3	guest	$1$3urY0m7m$PAsveAdEcMgzIyxSKF4cs0	10000.0000

Database

- What is it useful for?
  - Permanent store for objects, way to track and manage those objects easily -- think of something like user accounts.
  - Very easy paradigms, most essential in SQL are
    - SELECT
    - INSERT
    - DELETE
    - UPDATE

SELECT

- Select rows from a database matching a criteria.

	#	Column	Type
☐	1	**classid**	int(16)
☐	2	**classname**	varchar(255)
☐	3	**awesome**	tinyint(1)
☐	4	**slogan**	varchar(255)

+ Options

←T→	classid	classname	awesome	slogan
☐ Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
☐ Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

```
SELECT * FROM classes WHERE awesome = '1';
```

+ Options

←T→	classid	classname	awesome	slogan
☐ Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
☐ Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

SELECT

- Select rows from a database matching a criteria.

	#	Column	Type
☐	1	**classid**	int(16)
☐	2	**classname**	varchar(255)
☐	3	**awesome**	tinyint(1)
☐	4	**slogan**	varchar(255)

+ Options

←T→	classid	classname	awesome	slogan
☐ Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
☐ Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

```
SELECT * FROM classes WHERE classname = 'CS50';
```


+ Options

←T→	classid	classname	awesome	slogan
☐ Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?

SELECT

- How does this look like in code? Integrate PHP!

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

```
<?
     //construct sql string
     $sql = "SELECT * FROM classes WHERE awesome = '1'";
     $result = mysql_query($sql) or die("Query failed!");

     while($row = mysql_fetch_array($result))
     {
          echo $row["classname"] . ": " . $row["slogan"] . "<br>";
     }
?>
```

quit with error message if query didn't work

same as

```
if(!result)
  die("Query failed");
```

SELECT

- How does this look like in code? Integrate PHP!

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

$row

```
<?
    //construct sql string
    $sql = "SELECT * FROM classes WHERE awesome = '1'";
    $result = mysql_query($sql) or die("Query failed!");

    while($row = mysql_fetch_array($result))
    {
        echo $row["classname"] . ": " . $row["slogan"] . "<br>";
    }
?>
```

CS50: Wanna Learn HTML?

SELECT

- How does this look like in code? Integrate PHP!

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

$row

```
<?
    //construct sql string
    $sql = "SELECT * FROM classes WHERE awesome = '1'";
    $result = mysql_query($sql) or die("Query failed!");

    while($row = mysql_fetch_array($result))
    {
        echo $row["classname"] . ": " . $row["slogan"] . "<br>";
    }
?>
```

CS50: Wanna Learn HTML?
STAT110: FIND ALL THE MOMENTS

SELECT

- How does this look like in code? Integrate PHP!

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

$row

```
<?
     //construct sql string
     $sql = "SELECT * FROM classes WHERE awesome = '1'";
     $result = mysql_query($sql) or die("Query failed!");

false! ⟶ while($row = mysql_fetch_array($result))
     {
          echo $row["classname"] . ": " . $row["slogan"] . "<br>";
     }
?>
```

CS50: Wanna Learn HTML?
STAT110: FIND ALL THE MOMENTS

SELECT

- * selects all the columns. We can specify if we want fewer.

classname	slogan
CS50	Wanna Learn HTML?
STAT110	FIND ALL THE MOMENTS

```
<?
    //construct sql string
    $sql = "SELECT classname, slogan FROM classes WHERE awesome = '1'";
    $result = mysql_query($sql) or die("Query failed!");

    while($row = mysql_fetch_array($result))
    {
        echo $row["classname"] . ": " . $row["slogan"] . "<br>";
    }
?>
```

CS50: Wanna Learn HTML?
STAT110: FIND ALL THE MOMENTS

INSERT

- INSERT a row into a database!

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

```
<?
        //construct sql string
        $sql = "INSERT INTO classes(classname, awesome, slogan)
                VALUES ('CS164', 1,  'Learn iPhone development!')";

        mysql_query($sql) or die ("Query failed!");
    ?>
```

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS
Edit Copy Delete	4322	CS164	1	Learn iPhone development!

DELETE

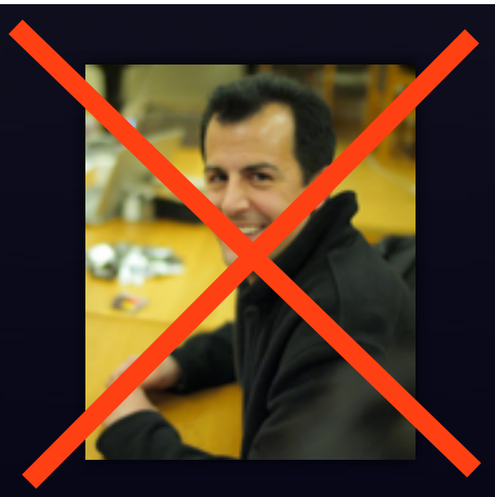
- DELETE rows from the database!

+ Options

←T→	classid	classname	instructor	awesome	slogan
Edit Copy Delete	1234	CS50	malan	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	blitzstein	1	Awesome!
Edit Copy Delete	4322	CS164	malan	1	Learn iPhone development!

```
<?
    //construct sql string
    $sql = "DELETE FROM classes WHERE instructor='malan'";

    mysql_query($sql) or die ("Query failed!");
?>
```



+ Options

←T→	classid	classname	instructor	awesome	slogan
Edit Copy Delete	4321	STAT110	blitzstein	1	Awesome!

UPDATE

- UPDATE a row in the database!

+ Options

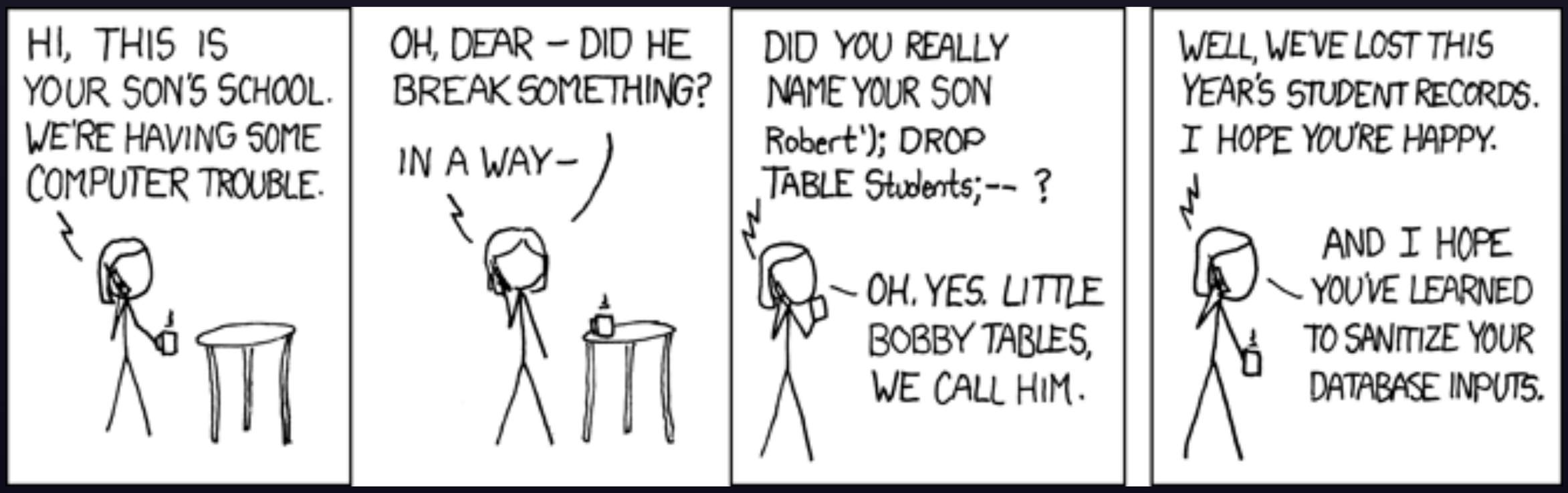
←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Wanna Learn HTML?
Edit Copy Delete	4321	STAT110	1	FIND ALL THE MOMENTS

```
<?
    //construct sql string
    $sql = "UPDATE classes SET slogan ='Awesome!' where awesome='1'";

    mysql_query($sql) or die ("Query failed!");
?>
```

+ Options

←T→	classid	classname	awesome	slogan
Edit Copy Delete	1234	CS50	1	Awesome!
Edit Copy Delete	4321	STAT110	1	Awesome!



CS50: Quiz 1

SQL Vulnerabilities

SQL Injection

- How does this work?

```
<?
    $username = $_POST["username"];
    $password = $_POST["password"];

    $sql = "SELECT * FROM users
            WHERE username='$username'
            AND password='$password'";

    $result = mysql_query($sql) or die("Query failed");

    if(mysql_num_rows($result) != 0)
    {
        $user = mysql_fetch_array($result);
        $_SESSION["id"] = $user["id"];
    }
?>
```

SQL Injection

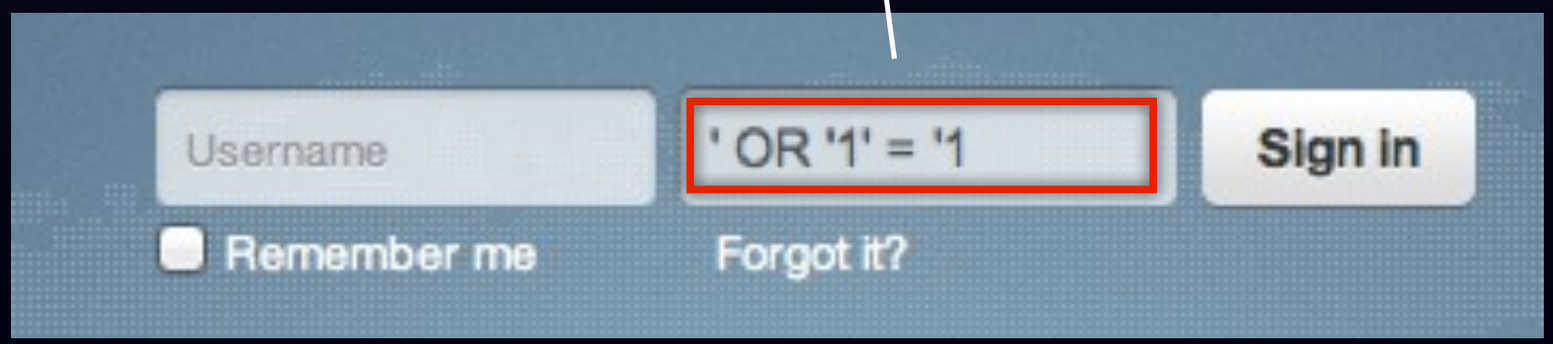
- How does this work?

```
<?
    $username = $_POST["username"];
    $password = $_POST["password"];

    $sql = "SELECT * FROM users
            WHERE username='$username'
            AND password='$password'";

    $result = mysql_query($sql) or die("Query failed");

    if(mysql_num_rows($result) != 0)
    {
        $user = mysql_fetch_array($result);
        $_SESSION["id"] = $user["id"];
    }
?>
```



SQL Injection

- How does this work?

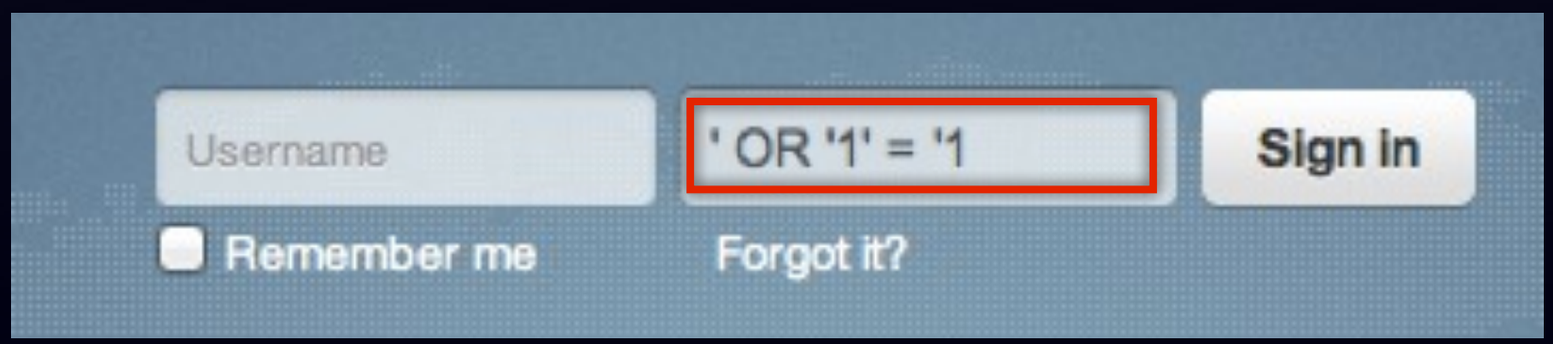
```
<?
    $username = $_POST["username"];
    $password = $_POST["password"];

    $sql = "SELECT * FROM users
            WHERE username=''
            AND password='' OR '1' = '1'";
    $result = mysql_query($sql) or die("Query failed");

    if(mysql_num_rows($result) != 0)
    {
        $user = mysql_fetch_array($result);
        $_SESSION["id"] = $user["id"];
    }
?>
```

true, always

so, returns ALL the rows



Solution

```
<?
    $username = mysql_real_escape_string($_POST["username"]);
    $password = mysql_real_escape_string($_POST["password"]);

    $sql = "SELECT * FROM users
            WHERE username='$username'
            AND password='$password'";

    $result = mysql_query($sql) or die("Query failed");

    if(mysql_num_rows($result) != 0)
    {
        $user = mysql_fetch_array($result);
        $_SESSION["id"] = $user["id"];
    }
?>
```

note: for one, this is still pretty terrible because you should never, ever, store passwords in plaintext in your database... hash first.

... and this is a cat :3

... and this is a Lexi o.O

CS50: Quiz 1

This was PHP/SQL

(bai, glhf!)

quiz1

Tommy MacWilliam

tmacwilliam@cs50.net

November 13, 2011

Today

- JavaScript <3333333333333333333333
- DOM
- AJAX

This was Joseph

JavaScript is the best programming language ever.

JavaScript

- PHP: server-side
  - runs on server, produces output, browser downloads
- JavaScript: client-side
  - browser downloads, runs code

Syntax

- syntax (also) very similar to C and PHP
  - `if`, `else`, `for`, `while`, etc.
  - strings are built in (just like PHP)
  - variables don't need dollar signs (yay!)
- no types for variables or functions
  - `x = 5;`
  - `function increment(x) { return ++x; }`

Script

- JavaScript can be inserted into your page using the `<script>` tag
  - `<script>` inside `<head>`: will be evaluated before page loads (used for functions and events)
  - `<script>` inside `<body>`: will be evaluated as page loads
- just like CSS, JavaScript can also be placed in an external file using the `src` attribute (not `href`)
  - `<script src="script.js"></script>`
  - CANNOT say `<script src="script.js" />`
    - :(

JavaScript

- example time!
  - `simple.html`

Arrays

- arrays declared using `[]`

```
var array = [2, 4, 6];
array[1] == 4
```

- length of array: `array.length`

Arrays

- arrays are dynamically-sized

```
var array = [2, 4, 6];
array[3] = 8;
array == [2, 4, 6, 8]
```

Associative Arrays

- associative array: set of key/value pairs (hash tables anyone?)
- declared using `{}`

```
var associative = { name: "tommy",
                    age: 20 };
associative["students"] = 15;
associative["name"] == "tommy"
```

Objects

- associative arrays are also objects

```
var object = { name: "tommy",
               age: 20 };
object.name == "tommy"
object.students = 15;
```

- syntax is interchangeable!
  - except in cases like `object["Mather House"]`

Iterating

- `for-in` loop for iterating over both arrays and associative arrays

```
for (var index in array)
    alert(index + "is: " + array[index]);
```

- for an array, `index` will be an integer ranging from `0` to `array.length - 1`
- for an associative array, `index` will be the keys in the associative array

Scope

- JavaScript scope is a bit different than other languages
- the `var` keyword limits a variable's scope
  - with `var`: scope limited to current function (not loop, as in C!)
  - without `var`: global scope

Scope

- example time!
  - `scope.html`

DOM

- the **D**ocument **O**bject **M**odel provides a way to access and manipulate HTML as objects
  - DOM is language-independent
  - languages like JavaScript, Python, and Ruby have DOM implementations
- each element is an object
  - attributes are properties of the object
  - contained tags are children of a parent tag

DOM

- in JavaScript, the DOM is loaded into the global `document` object
- `document.getElementById(id)`: get the element with the given ID
- `document.getElementsByTagName(tag)`: get an array of all elements with the given tag

DOM

- properties of JavaScript DOM objects
  - `innerHTML`: text contained within the element
  - `nodeName`: name of the tag of the element
  - `parentNode`: parent of current element, represented as a DOM object
  - `children`: array of child elements, represented as DOM objects
  - `style`: object representing CSS properties of element
  - `<attribute>`: each tag attribute has its own property

DOM

- example time!
  - `dom.html`

DOM

- the DOM also provides us a way to attach events to elements
  - event is a user interaction, like a mouse click
- common events
  - `onclick`: mouse click
  - `ondblclick`: double click
  - `onmouseover`: mouse moves over an object
  - `onmouseout`: mouse moves off an object
  - `onkeypress`: user pressed a key on the keyboard
  - `onload`: DOM has finished loading
  - `onchange`: value of a form changes
  - `onsubmit`: form is submitted

DOM

- function that fires when event occurs is called an “event handler”
- two ways to attach event handlers to elements
  - JavaScript: get DOM object, then add property for event name
  - HTML: use attributes like `onclick` and `onmouseover` and set value equal to name of function

DOM

- example time!
  - `events.html`

AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML
- allows a page to make dynamic HTTP requests without reloading the page
- has nothing to do with XML most of the time

AJAX

- AJAX functionality provided by the `XMLHttpRequest` class
- making an AJAX request
  - create an `XMLHttpRequest` object
  - construct the URL to make the request to
  - create an event handler to handle the server response
  - open/send the request

AJAX

- an AJAX request has 5 states
  - 0: not initialized
  - 1: connection established
  - 2: request received
  - 3: answer in progress
  - 4: done

AJAX

- HTTP status codes tell browser the result of the request
  - 200: OK
  - 301: Moved Permanently
  - 401: Unauthorized
  - 403: Forbidden
  - 404: Not Found

AJAX

- example time!
  - `ajax.html, time.php`

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">

function init() {
    // create ajax object
    var xhr = new XMLHttpRequest();
    // request url
    var url = "time.php";
    // event handler
    xhr.onreadystatechange = function() {
        // make sure request was valid
        if (xhr.readyState == 4 && xhr.status == 200)
           // display clock
           document.getElementById("time").innerHTML = xhr.responseText;
    }

    // refresh every 1000ms (1 second)
    setInterval(function() {
        xhr.open("GET", url, true);
        xhr.send();
    }, 1000);
}

    </script>
  </head>
  <body onload="init()">
    <h1>The SUPER AWESOME AJAX-enabled Web 2.0 CLOCK!!1!</h1>
    <h2 id="time" style="font-size: 36px; color: blue; margin-left: 100px;"></h2>
  </body>
</html>

```

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">

alert(document.images.length);

    </script>
  </head>
  <body>
    <h1 id="title">TOMMY'S AWESOME LINKS</h1>
    <a id="a-google" href="http://google.com">
       <img id="img-google" src="http://www.google.com/images/logos/ps_logo2.png" />
    </a>
    <br />
    <a id="a-cs50" href="http://cs50.net">
      <img id="img-cs50" src="https://www.cs50.net/images/red.png" />
    </a>

    <script type="text/javascript">

alert(document.getElementById("title").innerHTML);
alert(document.getElementById("a-google").children[0].nodeName);

    </script>
  </body>
</html>

```

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">

function init() {
    alert("page is loaded, let\'s do this!");
    // event handler when link is clicked
    document.getElementById("link").onclick = function() {
        // change href attribute to redirect to another page!
        alert("HAHA FOOLED YOU");
        document.getElementById("link").href = "http://google.com";
    };
}

function makeRed() {
    // make text red
    document.getElementById("paragraph").style.color = "red";
}

function makeBlue() {
    // make text blue
    document.getElementById("paragraph").style.color = "blue";
}


    </Script>
  </head>
  <body onload="init()">
    <h1>ANOTHER ONE OF TOMMY'S SWEET PAGES</h1>
    <a id="link" href="http://cs50.net">this is TOTALLY going to take you to cs50.net</a>
    <br />
    <p id="paragraph" onmouseover="makeRed()" onmouseout="makeBlue()">man javascript is so baller</p>
  </body>
</html>

```

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">

function f() {
    // no var: global variable
    x = 0;
    // var: local variable
    var y = 0;
    for (var i = 0; i < 5; i++) {
       // var limits scope to FUNCTION, not loop!
       var z = 4;
       x++;
       y++;
   }
   alert("z is " + z);
}

function g() {
   x++;
   alert("x is " + x);
}

f();
g();

   </script>
  </head>
  <body>
    <h1>ta-da!</h1>
  </body>
</html>
```

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">

alert("hello from head!");

function saySomething(something) {
    alert("I\'m supposed to tell you that " + something);
}

    </script>
  </head>
  <body>
    <script type="text/javascript">

alert('hello from body!')
saySomething("tommy rocks");

    </script>
  </body>
</html>

```

```php
<?php

echo date('h:i:s A');

?>
```