HTML Forms

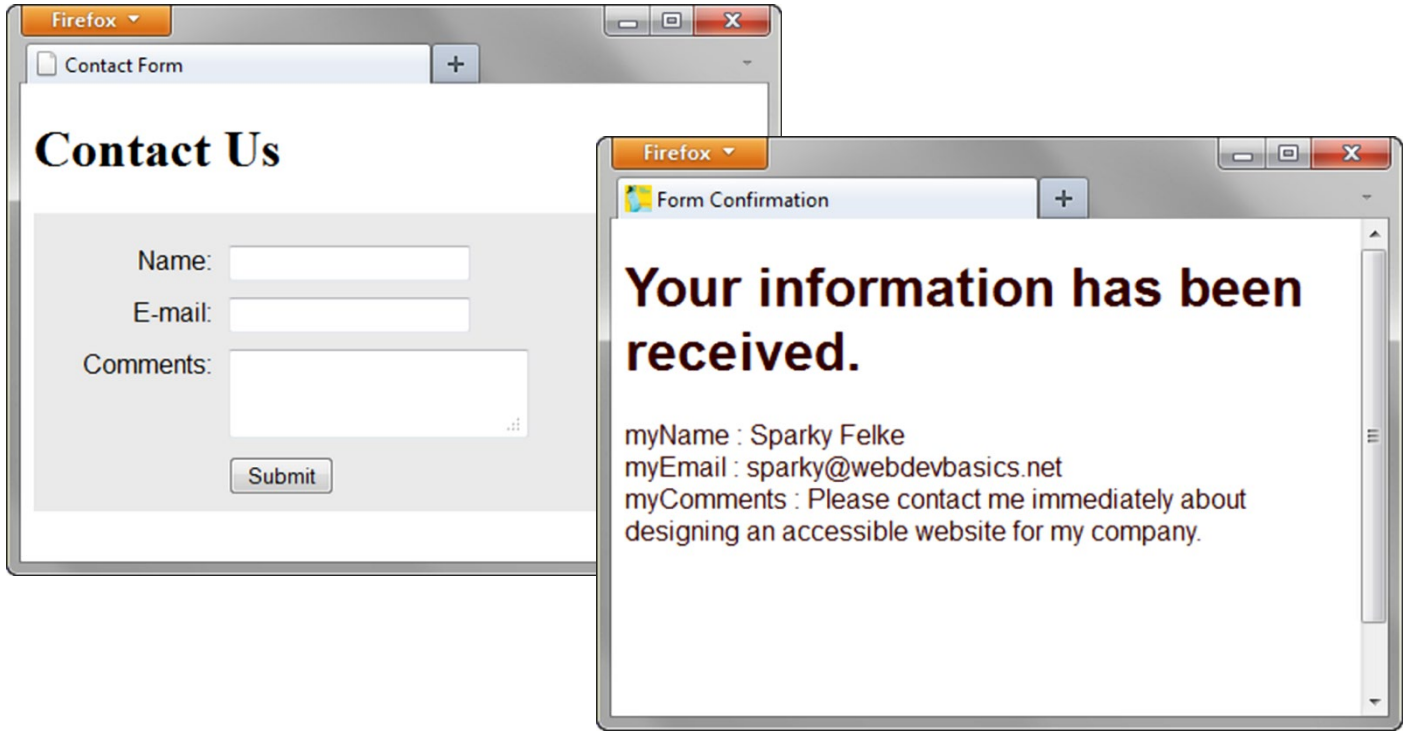
Part 2

Two Parts of Forms

- HTML form:
  - The user interface (client-side)
- Server-side processing:
  - The action (server-side)
  - send e-mail
  - write data to a text file
  - update a database
  - performs some other type of processing on the server

Sending information to a Server-side Script

```
<form method="post" action="somescript.php">
…
</form>
```



PHP – a server-side scripting language

- Required: PHP language installed and running on the web server
- Web developers: insert PHP code into an HTML document
- Requires the filename be `.php` …not `.html`
  - Tells the web server that there *may be* PHP code in the file
  - If there is PHP code in the file, the server will run it
- In the HTML document, use PHP by inserting the PHP tag:

```
<?php //PHP code here ?>
```

or

```
<?php
//PHP code here, on
//  multiple lines
?>
```

Variables in PHP

- Assign a variable using the assignment operator ("=")
  - Anything on the right gets inserted into anything on the left
  - Anything inserted replaces anything in the variable
  - In PHP, all variables start with `$`
  - E.g. `$myVariable = "Hello";`

- Concatenation
  - Add to a variable using the concatenation operator (".=")
  - Same as the assignment operator, but instead of replacing, it appends
  - E.g.

```
$myVariable = "Hello";
$myVariable .= " Professor!";
```

Some useful PHP commands

- `mail()`
  - Send mail using the server's mail server software
  - Uses four arguments: TO, SUBJECT, BODY, FROM
  - E.g.

```
mail("someone@example.com",
  "The subject line",
  "Hello World!",
  "From:<someoneElse@example.com>
);
```

- `echo`
  - displays the next item on the webpage
  - E.g.

```
echo "these words will appear on the webpage"
echo $myWords
```

Collect data from the HTML form

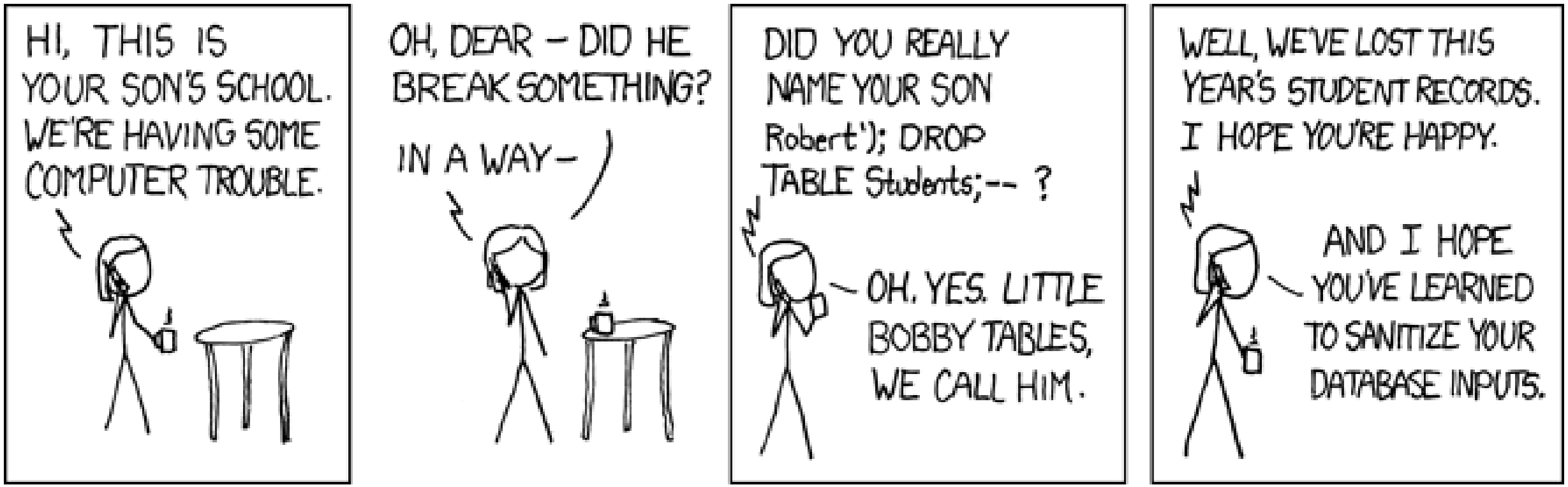
- Superglobals: **$_POST[]** or **$_GET[]**
- Pull data submitted via an HTML form
  - E.g. `$customerName = $_POST['name'];`

File 1: index.html

```
<form method="post" action="welcome.php">
   Name: <input type="text" name="name">
   E-mail: <input type="text" name="email">
   <input type="submit">
</form>
```

File 2: welcome.php

```
Welcome <?php echo $_POST["name"]; ?>
Your email address is: <?php echo $_POST["email"]; ?>
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY-
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Scrub incoming form data

- Nefarious users can trick servers to run code by entering it into a web form and submitting it
- Protect your web server by scrubbing any text inputs
- Use the PHP trim() and stripslashes() commands

```
$message = Trim(stripslashes($_POST['message']));
```

or (for database connections)…

```
$message = mysqli_real_escape_string($connection, $_POST['message']);
```