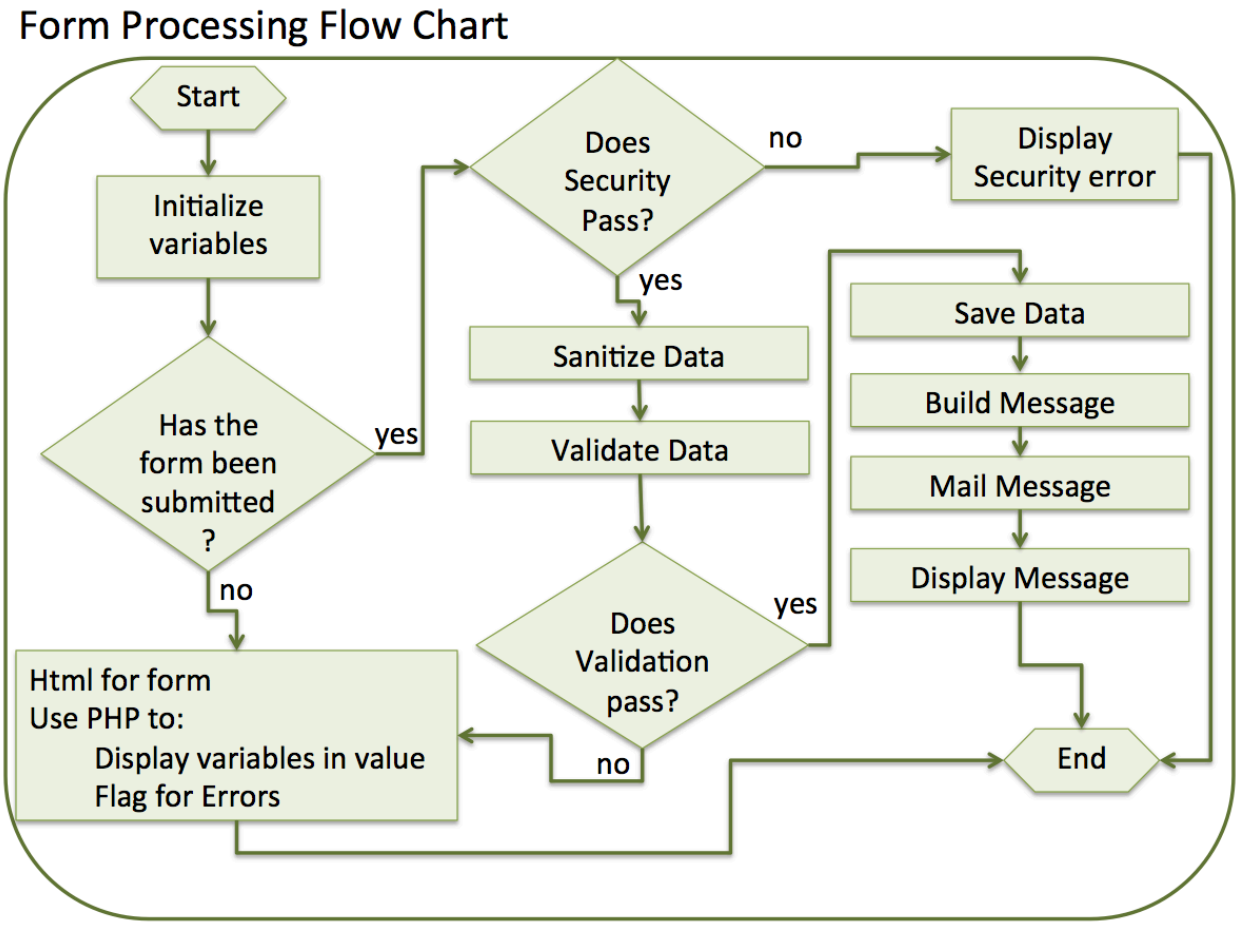


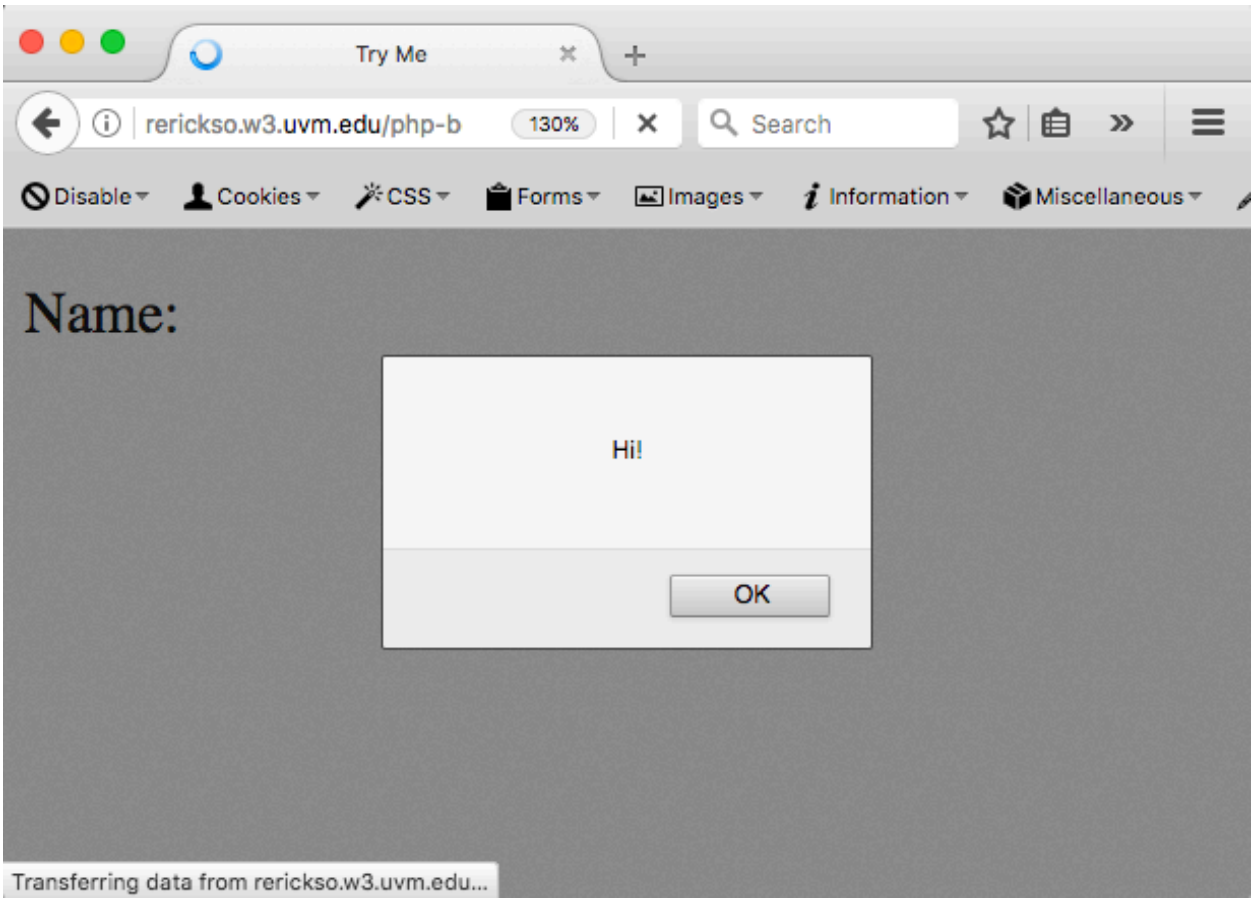
Next on our flow diagram list is to Sanitize our data which is one of the most important items we need to do, something we always have to do on the server EVERY TIME. This step is in Github Commit seven (along with validation which we talk about in the next chapter). You can sanitize your data with JavaScript on the client but since a person can bypass JavaScript we must ALWAYS be sure to sanitize our data on the server. What sanitize data means is to remove any potential injection attacks. For example if we ask for a persons name someone could type in:

<script>alert("hi")</script>

Then if we display the persons name with php it will print out the JavaScript and have the JavaScript alert box pop up. Of course that is harmless but if a hacker can have an alert box pop up they can do as much JavaScript as they want.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Try Me</title>
        <meta charset="utf-8">
```

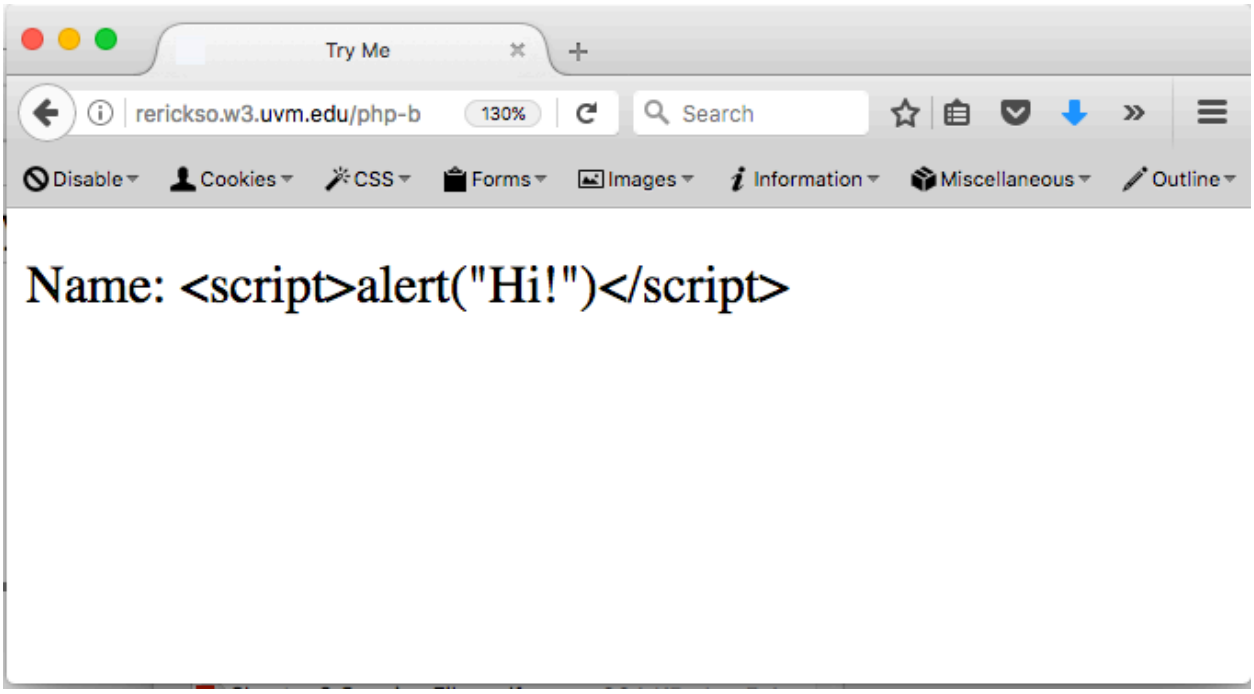
```
        <meta name="description" content="just a testing file">
    </head>
    <body>
    <?php
        $_POST["txtFirstName"] = '<script>alert("Hi!")</script>';
        $firstName = $_POST["txtFirstName"];
        print "<p>Name: " . $firstName;
    ?>
    </body>
</html>
```



We have three main things that we can do to sanitize our data.

1. Push everything through html entities.

```
<?php
$_POST["txtFirstName"] = '<script>alert("Hi!")</script>';
$firstName = htmlentities($_POST["txtFirstName"], ENT_QUOTES, "UTF-8");
print "<p>Name: " . $firstName;
?>
```

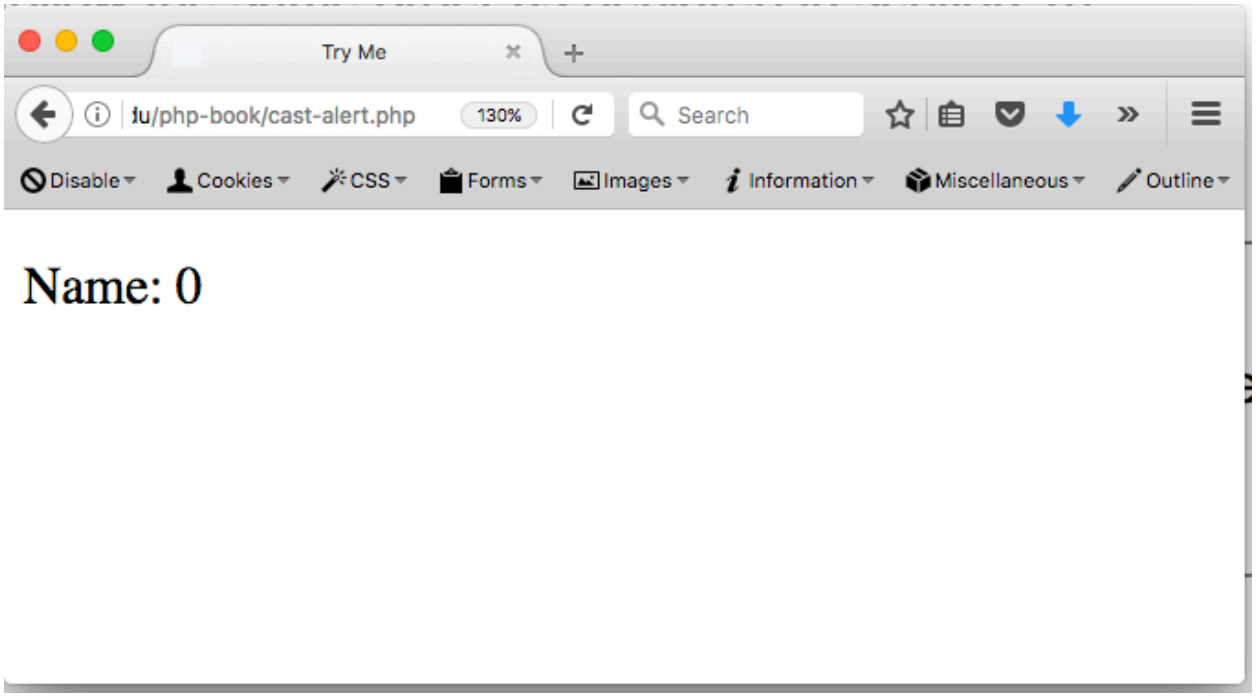


2. Use PHP built in filter vars [http://php.net/manual/en/filter.filters.php]. Filter Vars is a fancy html entities (or htmlspecialchars: Try googling: htmlentities vs htmlspecialchars)We have not used filter vars yet but they have one just for an email address since it is so common to input an email address.

```
$email = filter_var($_POST["txtEmail"], FILTER_SANITIZE_EMAIL);
```

3. Cast [http://php.net/manual/en/language.types.type-juggling.php]the value to a specfic data type for example an integer to make a whole number:

```
<?php
$_POST["txtFirstName"] = '<script>alert("Hi!")</script>';
$firstName = (int) $_POST["txtFirstName"];
print "<p>Name: " . $firstName;
?>
```



Of course this makes the first name a number zero but it helps when someone types in something they should not when we are expecting a number. You would not of course cast first name to an integer. You would cast quantity ordered to an integer.

Summary

Security is a large problem on the web and we do to everything we can to help stop the problem.

Php Htmlentities is a simple built in function we can use that will convert all applicable characters to HTML entities. This will convert JavaScript to just plain text and not execute.

Self Test Questions

1. **Email sanitizer**
   Write some php code that loads a list of emails from a csv file, sanitizes each email, and displays the email on screen.

2. **Name sanitizer**
   Write some php code that loads a csv file full of names with extra characters (!, ?,&,123, etc) and sanitizes them so they are just the names. Print the list of names on screen.

3. **URL sanitizer**
   Write some php code that loads a list of URLS from a csv file and sanitizes them. Print the list of URL's on screen.

4. **Integer sanitizer**

Write some php code that loads a file full of random characters such as integers, letters and symbols and only parses out the integers. Display only the integers on screen.

5. **Floating point sanitizer**
   Write some php code that loads a file full of random characters such as floating point numbers, integers, letters and symbols and only parses out the integers. Display only the floating point numbers on screen. (i.e. 0.01, 5.35, 99.32)