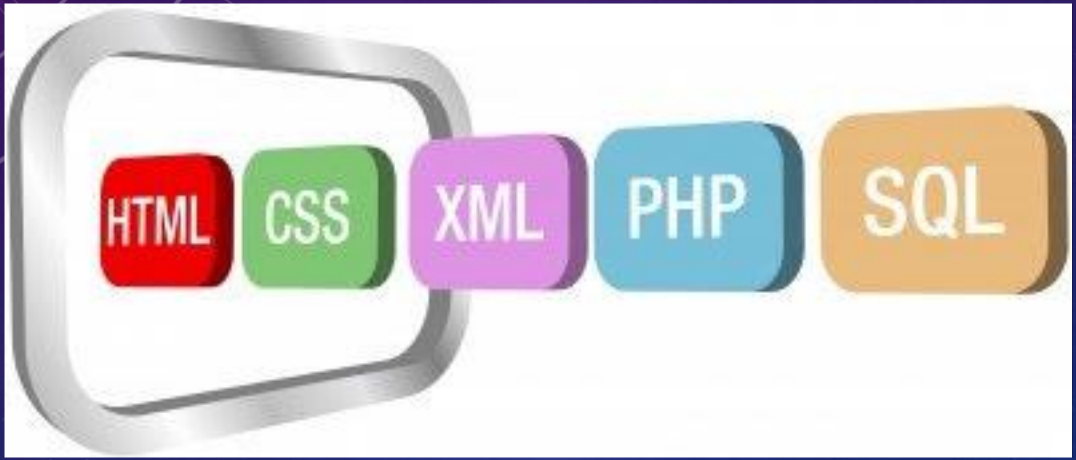
LECTURE 05 (01): HTML FORMS AND SERVER-SIDE DATA

Web Programming

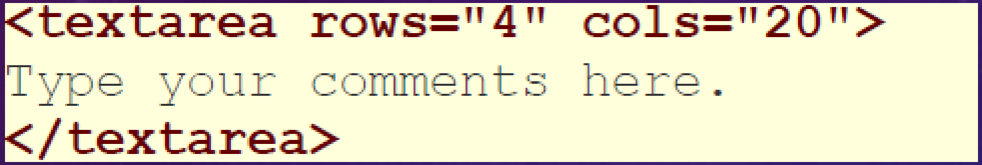


INSTRUCTOR: DR. HOSSAM ZAWBAA

TEXT BOXES: <TEXTAREA>

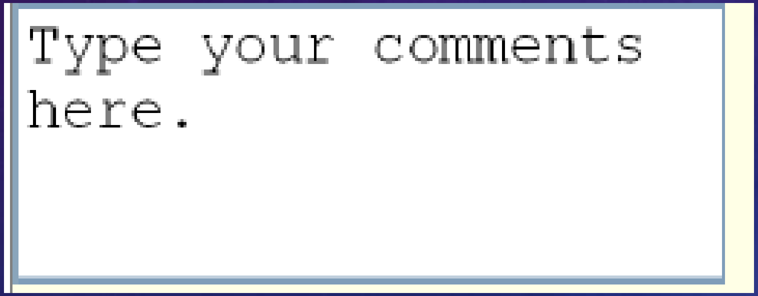
- a multi-line text input area (inline)

HTML

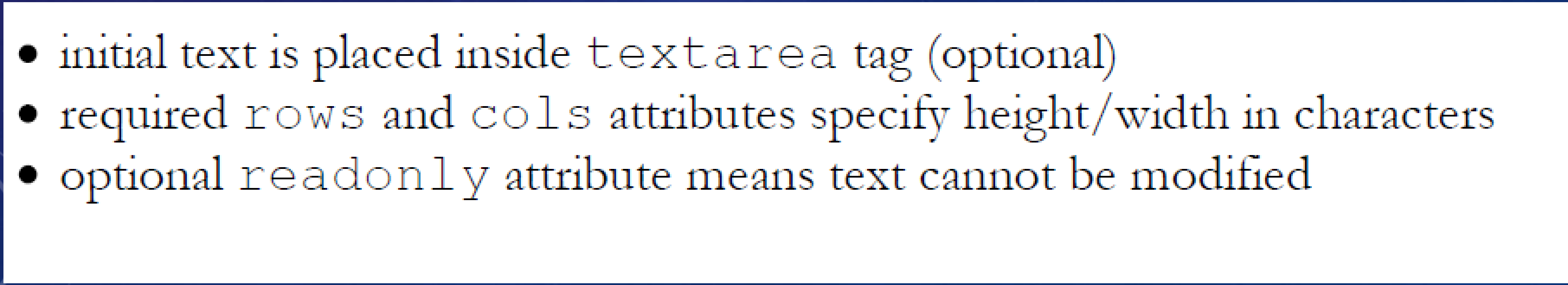


```html
<textarea rows="4" cols="20">
Type your comments here.
</textarea>
```

OUTPUT



```
Type your comments
here.
```



- initial text is placed inside `textarea` tag (optional)
- required `rows` and `cols` attributes specify height/width in characters
- optional `readonly` attribute means text cannot be modified

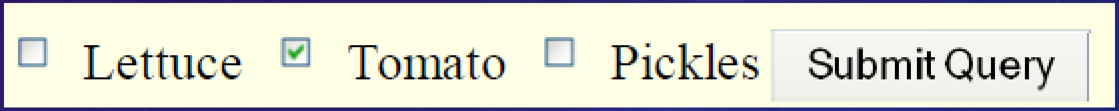
CHECKBOXES: <INPUT>

- yes/no choices that can be checked and unchecked (inline)

HTML

```
<input type="checkbox" name="lettuce" /> Lettuce
<input type="checkbox" name="tomato" checked="checked" /> Tomato
<input type="checkbox" name="pickles" /> Pickles
```

OUTPUT

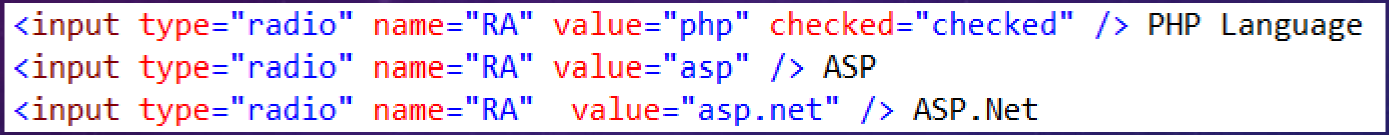


- none, 1, or many checkboxes can be checked at same time
- when sent to server, any checked boxes will be sent with value `on`:
  - `http://webster.cs.washington.edu/params.php?tomato=on&pickles=on`
- use `checked="checked"` attribute in HTML to initially check the box

RADIO BUTTONS: <INPUT>

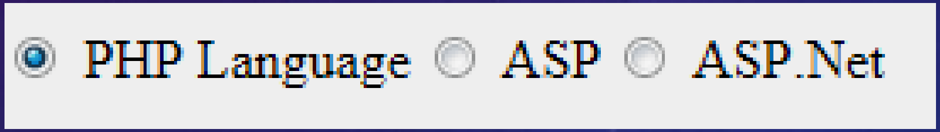
•sets of mutually exclusive choices (inline)

HTML

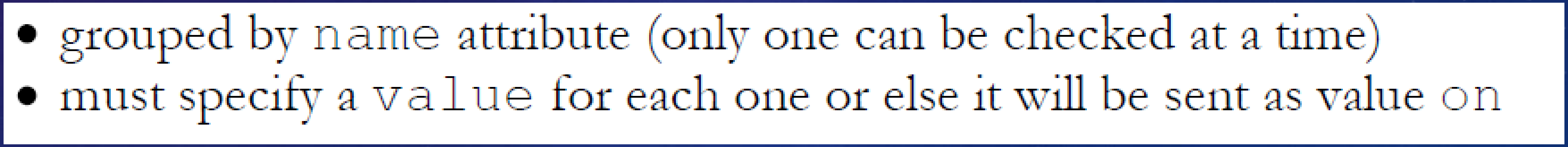


```
<input type="radio" name="RA" value="php" checked="checked" /> PHP Language
<input type="radio" name="RA" value="asp" /> ASP
<input type="radio" name="RA"  value="asp.net" /> ASP.Net
```

OUTPUT



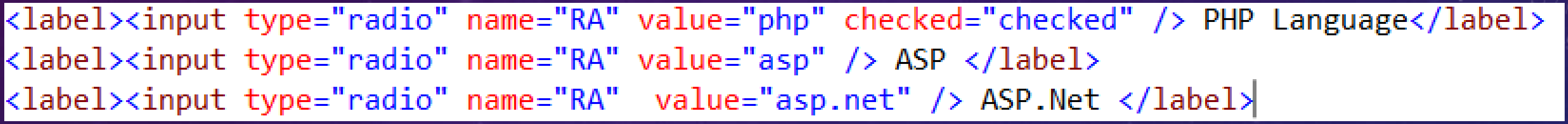
◉ PHP Language ○ ASP ○ ASP.Net



- grouped by `name` attribute (only one can be checked at a time)
- must specify a `value` for each one or else it will be sent as value `on`

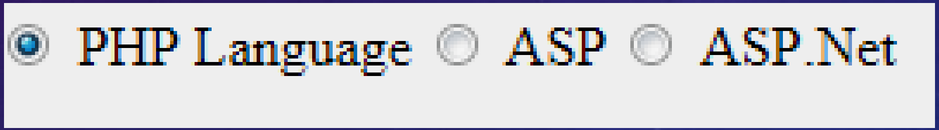
TEXT LABELS: <LABEL>

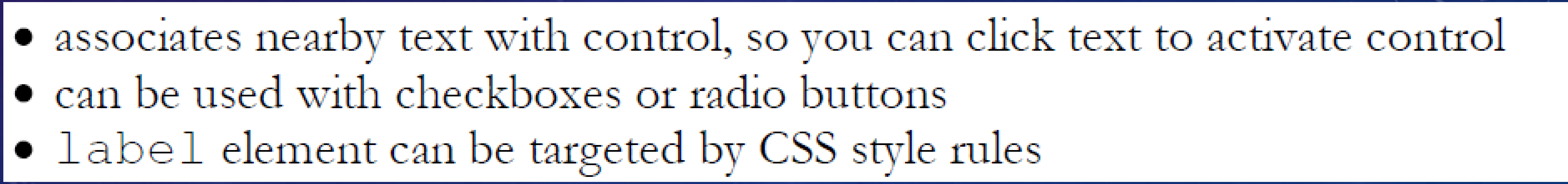
HTML



```
<label><input type="radio" name="RA" value="php" checked="checked" /> PHP Language</label>
<label><input type="radio" name="RA" value="asp" /> ASP </label>
<label><input type="radio" name="RA"  value="asp.net" /> ASP.Net </label>
```

OUTPUT



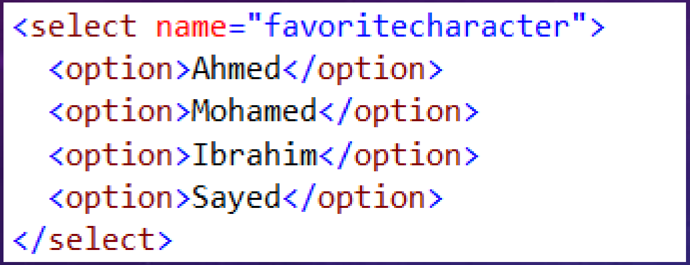


- associates nearby text with control, so you can click text to activate control
- can be used with checkboxes or radio buttons
- `label` element can be targeted by CSS style rules

DROP-DOWN LIST: <SELECT>, <OPTION>

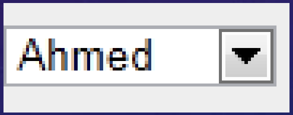
- menus of choices that collapse and expand (inline)

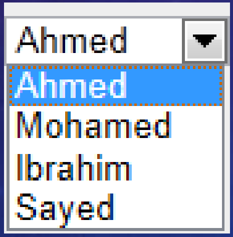
HTML



```
<select name="favoritecharacter">
  <option>Ahmed</option>
  <option>Mohamed</option>
  <option>Ibrahim</option>
  <option>Sayed</option>
</select>
```

OUTPUT





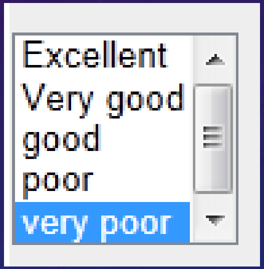
- `option` element represents each choice
- `select` optional attributes: `disabled`, `multiple`, `size`

USING <SELECT> FOR LISTS

HTML

```
<select name="favoritecharacter[]" size="5" multiple="multiple">
  <option>Excellent</option>
  <option>Very good</option>
  <option>good</option>
  <option>poor</option>
  <option selected="selected">very poor</option>
</select>
```

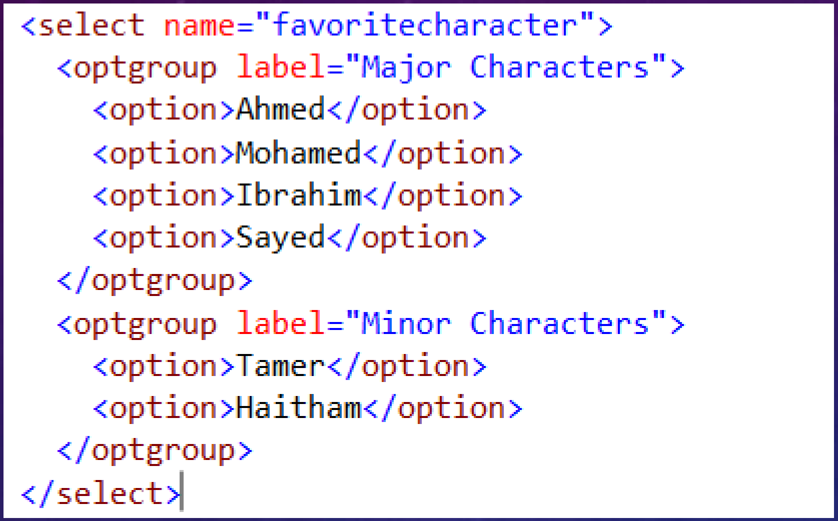
OUTPUT



- optional `multiple` attribute allows selecting multiple items with shift- or ctrl-click
    - must declare parameter's name with `[]` if you allow multiple selections
- `option` tags can be set to be initially `selected`

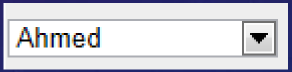
OPTION GROUPS: <OPTGROUP>

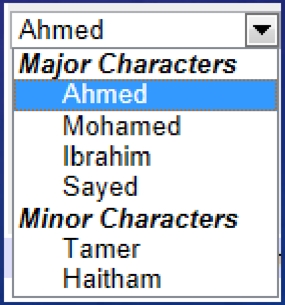
HTML



```
<select name="favoritecharacter">
  <optgroup label="Major Characters">
    <option>Ahmed</option>
    <option>Mohamed</option>
    <option>Ibrahim</option>
    <option>Sayed</option>
  </optgroup>
  <optgroup label="Minor Characters">
    <option>Tamer</option>
    <option>Haitham</option>
  </optgroup>
</select>
```

OUTPUT





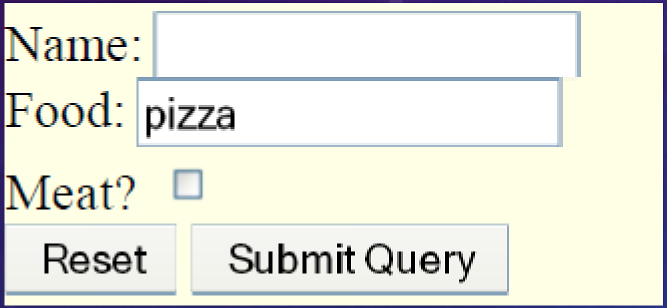
```
Ahmed
Major Characters
    Ahmed
    Mohamed
    Ibrahim
    Sayed
Minor Characters
    Tamer
    Haitham
```

RESET BUTTONS

HTML

```
Name: <input type="text" name="name" /> <br />
Food: <input type="text" name="meal" value="pizza" /> <br />
<label>Meat? <input type="checkbox" name="meat" /></label> <br />
<input type="reset" />
```

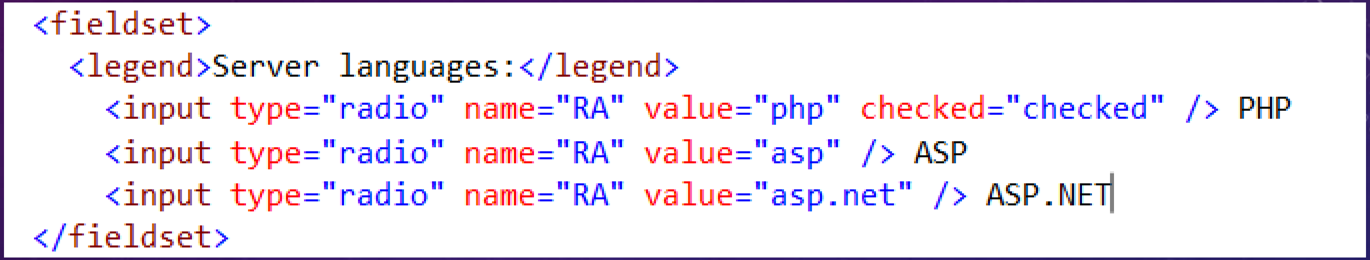
OUTPUT



- when clicked, returns all form controls to their initial values
- specify custom text on the button by setting its `value` attribute

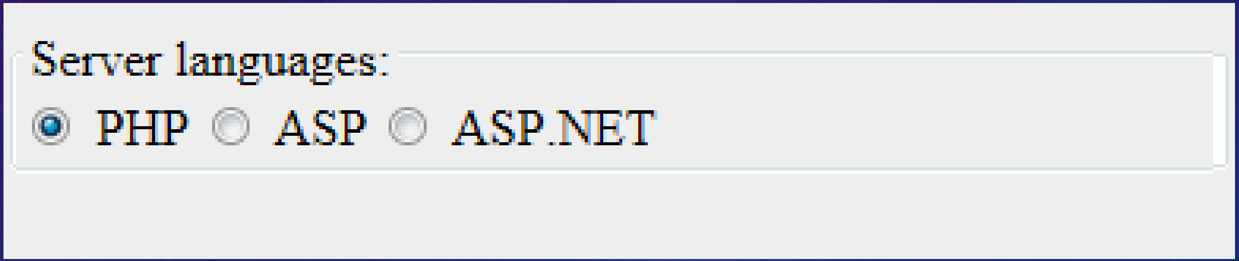
GROUPING INPUT: <FIELDSET>, <LEGEND>

HTML



```
<fieldset>
  <legend>Server languages:</legend>
    <input type="radio" name="RA" value="php" checked="checked" /> PHP
    <input type="radio" name="RA" value="asp" /> ASP
    <input type="radio" name="RA" value="asp.net" /> ASP.NET
</fieldset>
```

OUTPUT

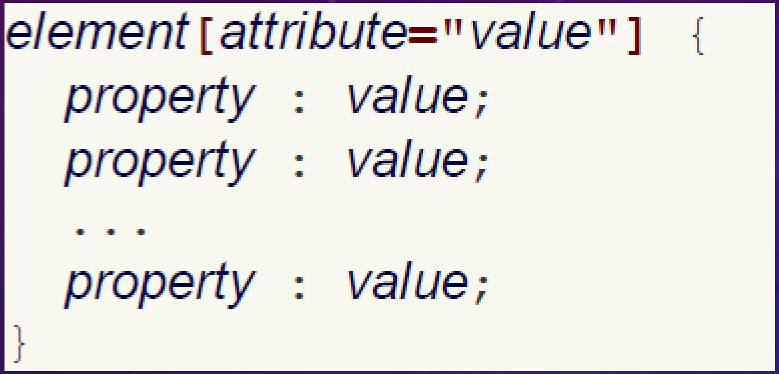


```
Server languages:
(●) PHP ( ) ASP ( ) ASP.NET
```

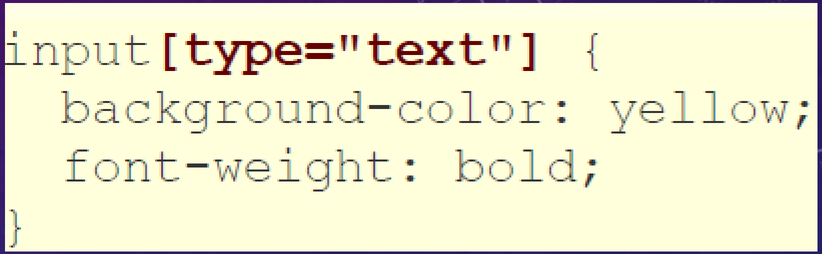
- `fieldset` groups related input fields; `legend` supplies an optional caption

STYLING FORM CONTROLS

CSS

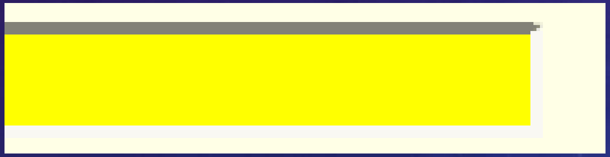


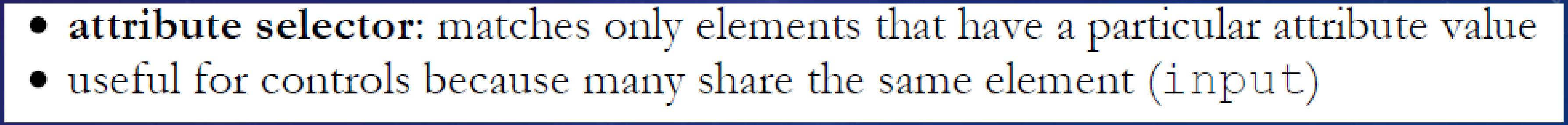
```
element[attribute="value"] {
   property : value;
   property : value;
   ...
   property : value;
}
```



```
input[type="text"] {
  background-color: yellow;
  font-weight: bold;
}
```

OUTPUT

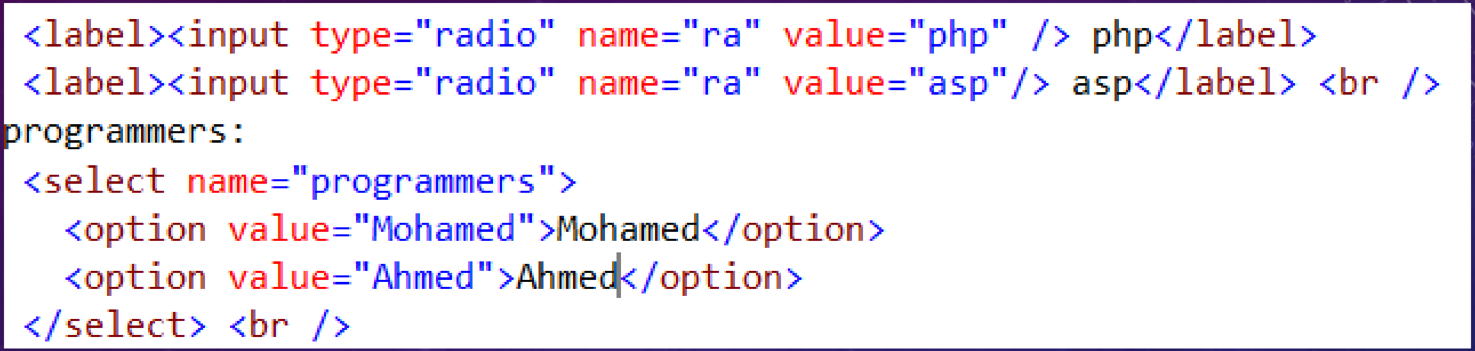




- **attribute selector:** matches only elements that have a particular attribute value
- useful for controls because many share the same element (`input`)

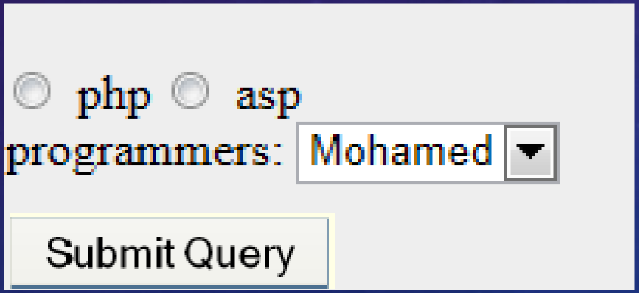
SUBMITTING DATA

HTML



```
 <label><input type="radio" name="ra" value="php" /> php</label>
 <label><input type="radio" name="ra" value="asp"/> asp</label> <br />
programmers:
 <select name="programmers">
   <option value="Mohamed">Mohamed</option>
   <option value="Ahmed">Ahmed</option>
 </select> <br />
```

OUTPUT



```
○ php ○ asp
programmers: Mohamed ▾
Submit Query
```

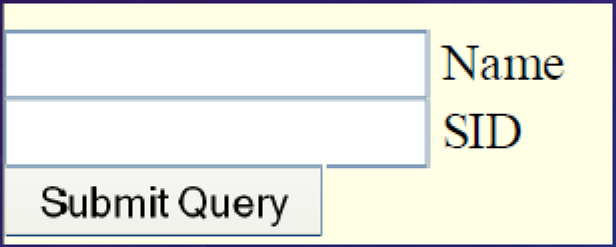
[ra] => php, [programmers] => Ahmed

HIDDEN INPUT PARAMETERS

HTML

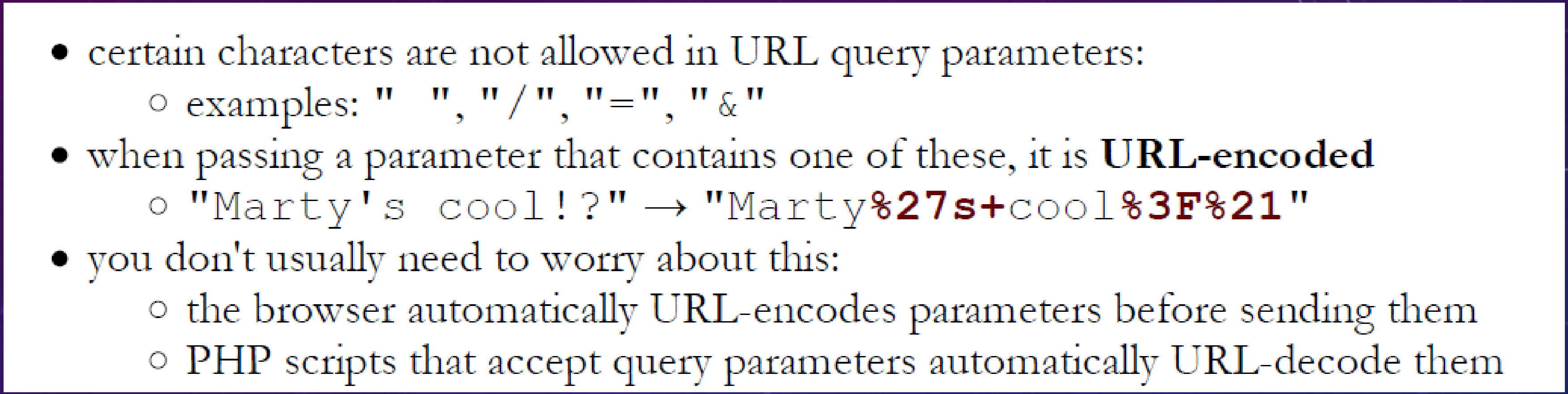
```
<input type="text" name="username" /> Name <br />
<input type="text" name="sid" /> SID <br />
<input type="hidden" name="school" value="UW" />
<input type="hidden" name="quarter" value="48sp" />
```

OUTPUT



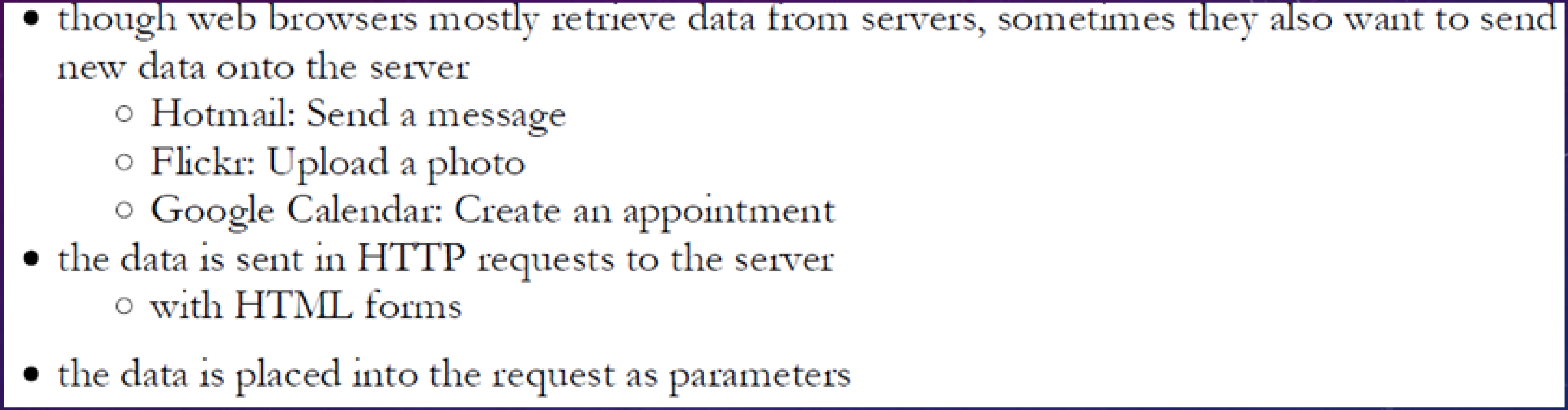
- an invisible parameter that is still passed to the server when form is submitted
- useful for passing on additional state that isn't modified by the user

URL-ENCODING



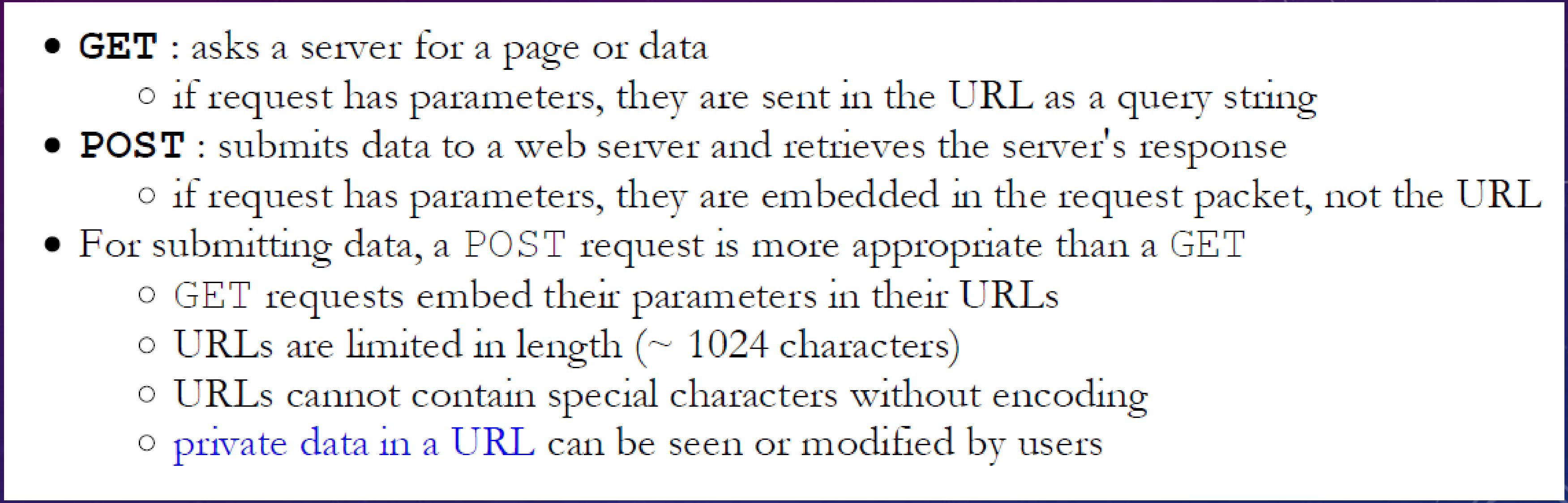
- certain characters are not allowed in URL query parameters:
  - examples: `" "`, `"/"`, `"="`, `"&"`
- when passing a parameter that contains one of these, it is **URL-encoded**
  - `"Marty's cool!?"` → `"Marty%27s+cool%3F%21"`
- you don't usually need to worry about this:
  - the browser automatically URL-encodes parameters before sending them
  - PHP scripts that accept query parameters automatically URL-decode them

SUBMITTING DATA TO A WEB SERVER



- though web browsers mostly retrieve data from servers, sometimes they also want to send new data onto the server
  - Hotmail: Send a message
  - Flickr: Upload a photo
  - Google Calendar: Create an appointment
- the data is sent in HTTP requests to the server
  - with HTML forms
- the data is placed into the request as parameters

HTTP GET VS. POST REQUESTS



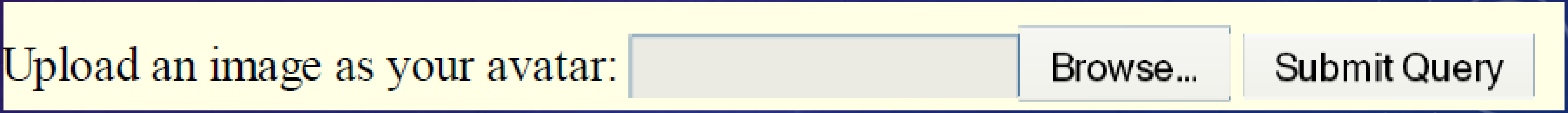
- **GET** : asks a server for a page or data
  - if request has parameters, they are sent in the URL as a query string
- **POST** : submits data to a web server and retrieves the server's response
  - if request has parameters, they are embedded in the request packet, not the URL
- For submitting data, a POST request is more appropriate than a GET
  - GET requests embed their parameters in their URLs
  - URLs are limited in length (~ 1024 characters)
  - URLs cannot contain special characters without encoding
  - private data in a URL can be seen or modified by users

UPLOADING FILES

HTML

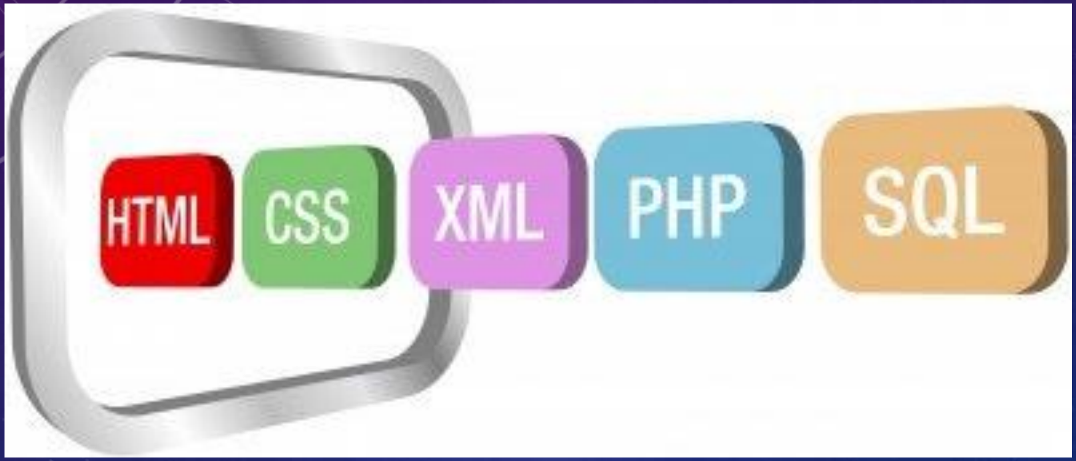
```
<form action="http://webster.cs.washington.edu/params.php"
      method="post" enctype="multipart/form-data">
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```

OUTPUT


Upload an image as your avatar:

LECTURE 05 (02): PHP

Web Programming



INSTRUCTOR: DR. HOSSAM ZAWBAA

HISTORY OF PHP

- ✓ **PHP (Personal Home Page)** (PHP: Hypertext Preprocessor) was created by Rasmus Lerdorf in **1994**. It was initially developed as a server-side form generation in Unix.
- ✓ **PHP 2 (1995)** transformed the language into a Server-side embedded scripting language. Added database support, file uploads, variables, arrays, recursive functions, conditionals, iteration, regular expressions, etc.
- ✓ **PHP 3 (1998)** added support for ODBC data sources, multiple platform support, email protocols (SNMP,IMAP), and new parser written by Zeev Suraski and Andi Gutmans .

PHP

- ✓ **PHP 4 (2000)** became an independent component of the web server for added efficiency. Many security features were added.
- ✓ **PHP 5 (2004)** adds object oriented programming, robust XML support using the libxml2 library, SOAP extension for interoperability with Web Services, SQLite has been bundled with PHP

PHP

- PHP is *a Server-side Scripting Language* designed specifically for the Web.
- An **open source** language
- PHP code can be embedded within an **HTML page**, which will be executed each time that page is visited.
- Filenames end with **.php** by convention

PHP

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
- ‘View Source’ in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

PHP

- Open source / free software
- Cross platform to develop and deploy and to use
- Powerful, robust , scalable
- Web development specific
- Can be object oriented especially version 5
- Large active developer community (20 millions websites)
- Great documentation in many language

www.php.net/docs.php

PHP

- **Installation**

1. Web server (Apache)
2. PHP
3. Database (MySQL)
4. Text editor (Notepad)
5. Web browser (Firefox)
6. www.php.net/manual/en/install.php

- **EasyPHP** is recommended.

WHAT DOES PHP CODE LOOK LIKE?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
 …
?>
```

SYNTAX PHP CODE

- Standard Style :

```
<?php  …… ?>
```

- Short Style:

```
<? … ?>
```

- Script Style:

```
<SCRIPT LANGUAGE=‘php’> </SCRIPT>
```

ECHO

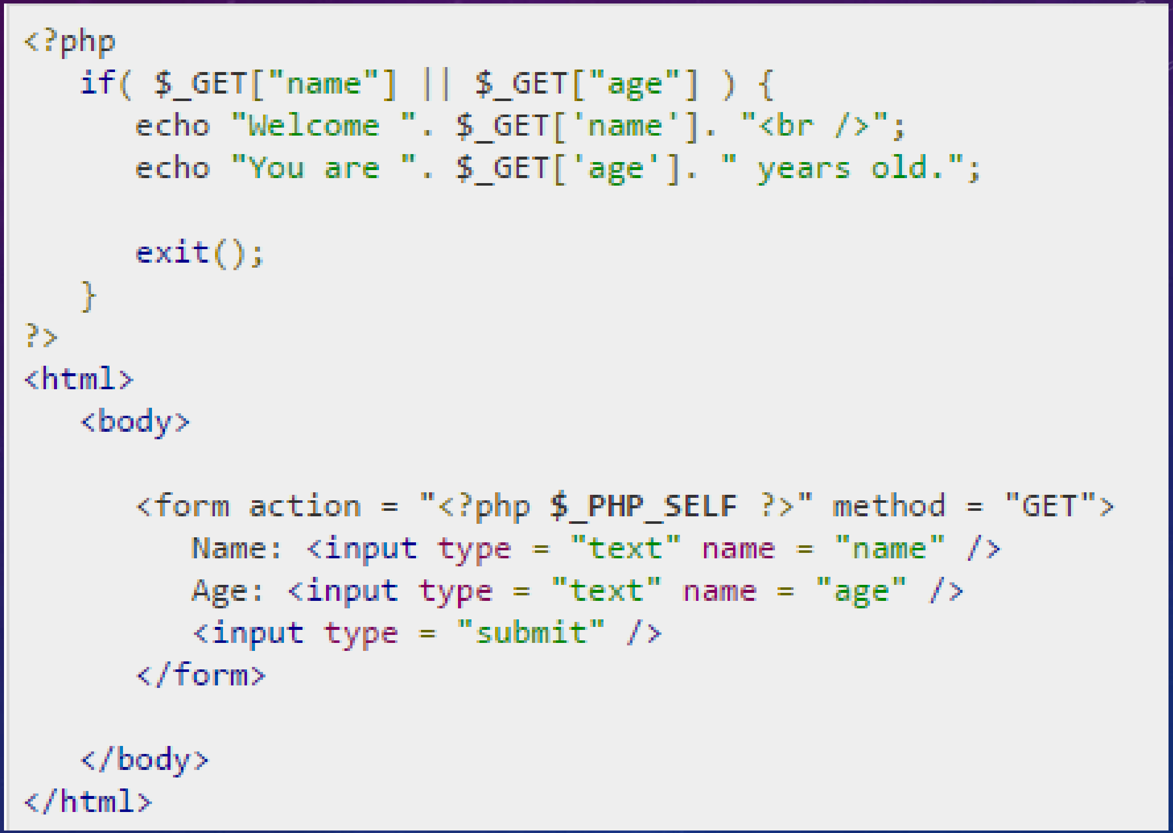
- The PHP command ‘echo’ is used to output the parameters passed to it .
- The typical usage for this is to send data to the client’s web-browser

ECHO - EXAMPLE

```
<?php
    echo “ This my first statement in PHP language“;
?>
```

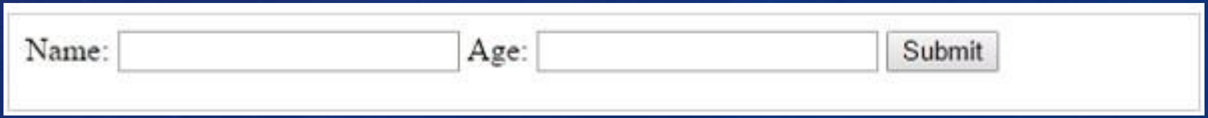
FORM GET EXAMPLE

HTML



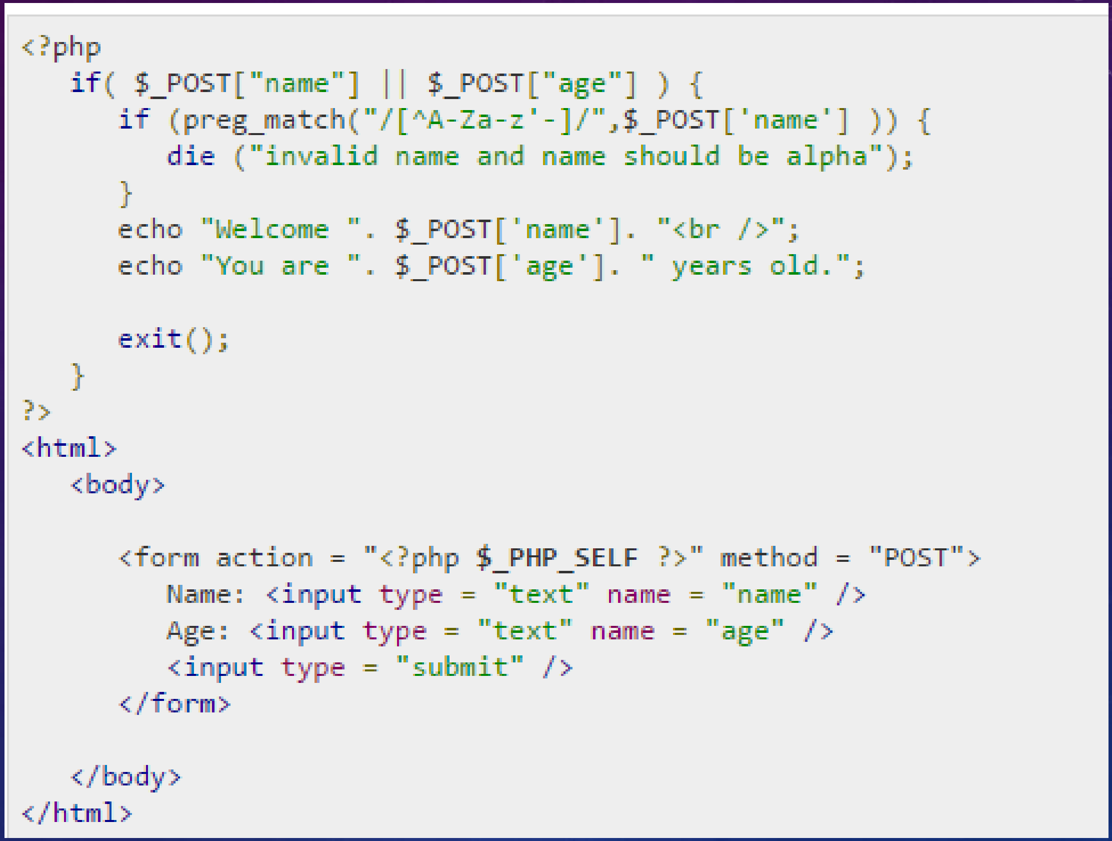
```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "GET">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

OUTPUT



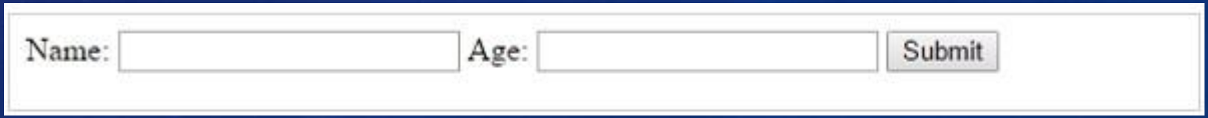
FORM POST EXAMPLE

HTML



```
<?php
   if( $_POST["name"] || $_POST["age"] ) {
      if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
         die ("invalid name and name should be alpha");
      }
      echo "Welcome ". $_POST['name']. "<br />";
      echo "You are ". $_POST['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "POST">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

OUTPUT


Name:
Age:
Submit