PHP Forms Processing

Presented by Ulvi Bajarani, SID 20539914

FIGURE 5.1: Form Processing

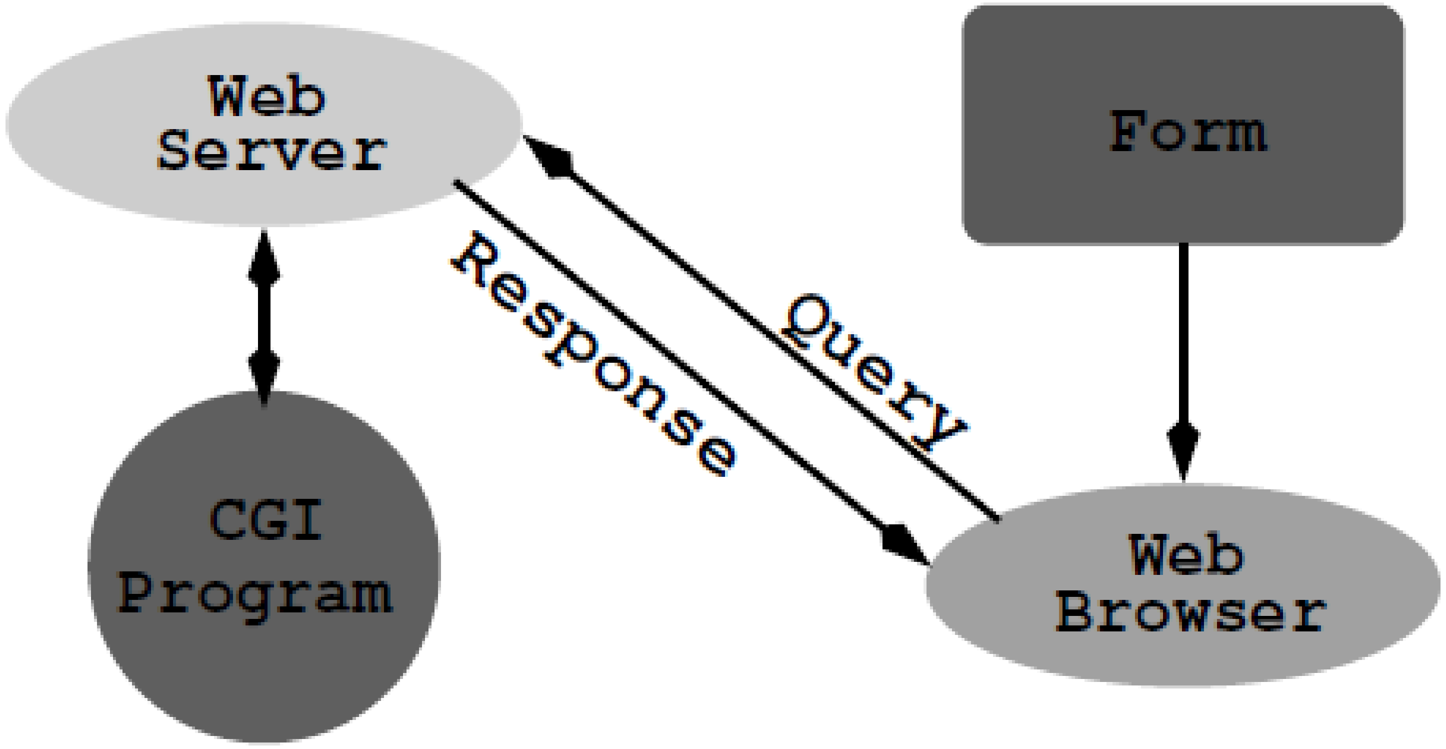
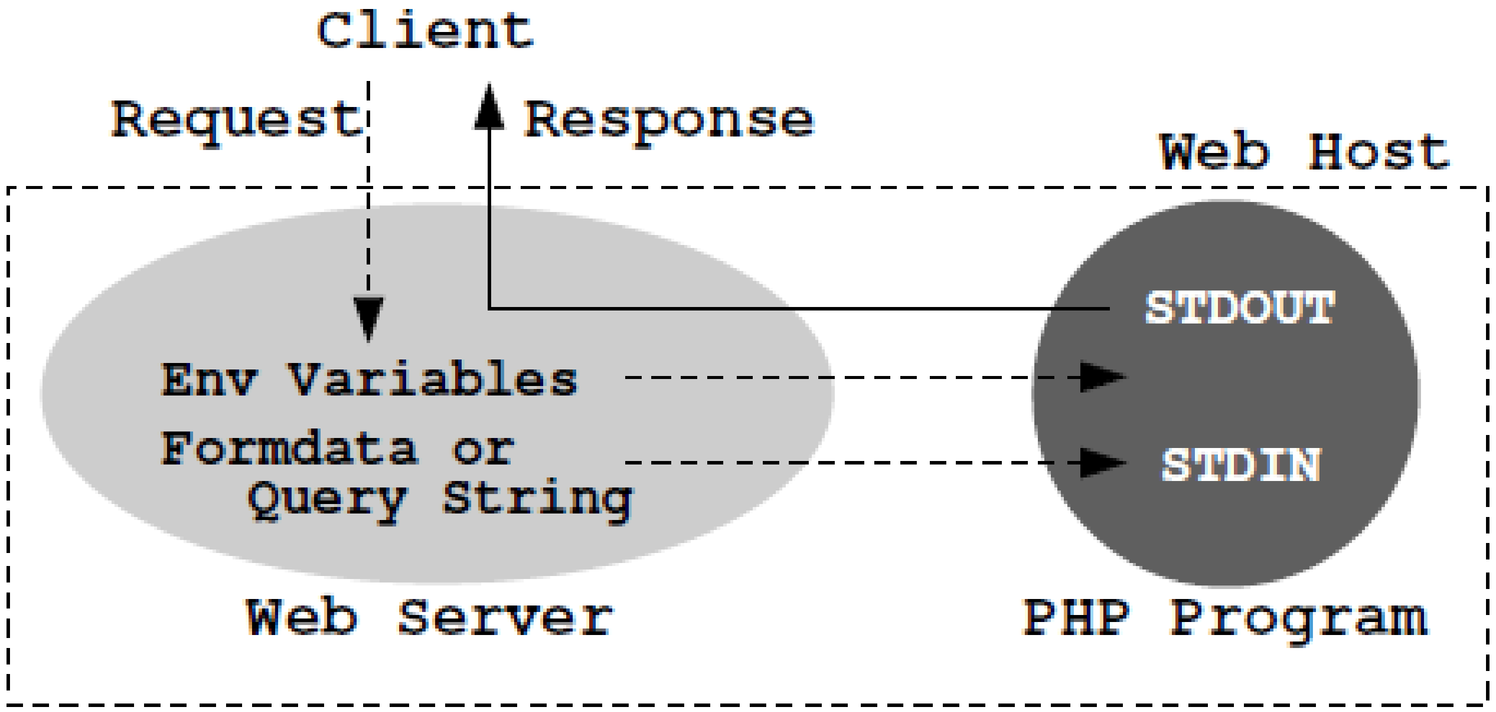


FIGURE 5.15: CGI Data Flow



HTML Forms

- Handled by *<form></form>* tags.
- Might have various input control element types.
- Might have either *get* or *post* methods.
- Might have 3 types of encoding (only in the *post* method).
- Might have user defined labels, which is important to define inputs.
- Might have own attributes, event attributes and global attributes.

HTML Forms

- Might contain one or more of the following form elements:
  - *<input>*
  - *<textarea>*
  - *<button>*
  - *<select>*
  - *<option>*
  - *<optgroup>*
  - *<fieldset>*
  - *<label>*
  - *<output>*
  - *<datalist>*

HTML <form> attributes (*value in the parentheses*)

- accept-charset (*character_set*) – Specifies the type of the character encoding to be used for the form submission. Common values are UTF-8 and ISO-8859-1.
- ***action*** (*URL*) – Specifies the place where the form is sent by the form submission.

HTML <form> attributes (*value in the parentheses*) [continued]

- autocomplete (*on* or *off*) – Specifies if the form could have autocomplete on or off.
- ***enctype*** (*application/x-www-form-urlencoded* or *multipart/form-data* or *text/plain*) – Specifies the place where the form is sent by the form submission.

HTML <form> attributes (*value in the parentheses*) [continued]

- ***method*** (*get* or *post*) – Specifies the HTTP method when sending the data in forms.
- rel (*external* or *help* or *license* or *next* or *nofollow* or *noopener* or *noreferrer* or *opener* or *prev* or *search*) – Specifies the relationship between a linked resource and the current document.

enctype attribute values

- *application/x-www-form-urlencoded* Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
- *multipart/form-data* No characters are encoded. This value is required when you are using forms that have a file upload control.
- *text/plain* Spaces are converted to "+" symbols, but no special characters are encoded.

The *get* method

- Appends form-data into the URL in name/value pairs: URL?name=value&name=value
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result, because might be cached.
- GET is better for non-secure data, like query strings in Google.

The *post* method

- Appends form-data inside the body of the HTTP request (data is not shown in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked
- Mostly used to send the sensitive data (password, etc.)

HTML form elements

<label>

<label>

- In the proper usage, beneficial
  - for screen reader users;
  - for users who have difficulty clicking on very small regions (such as checkboxes): because when a user clicks the text within the <label> element, it toggles the input (this increases the hit area).

<label>

- Defines the label for the next tags:
  - *<input type="checkbox">*
  - *<input type="color">*
  - *<input type="date">*
  - *<input type="datetime-local">*
  - *<input type="email">*
  - *<input type="file">*
  - *<input type="month">*
  - *<input type="number">*

<label> (continued)

- Defines the label for the next tags:
  - *<input type="password">*
  - *<input type="radio">*
  - *<input type="range">*
  - *<input type="search">*
  - *<input type="tel">*
  - *<input type="text">*
  - *<input type="time">*
  - *<input type="url">*
  - *<input type="week">*

<label> (continued)

- Defines the label for the next tags:
  - *<meter>*
  - *<progress>*
  - *<select>*
  - *<textarea>*

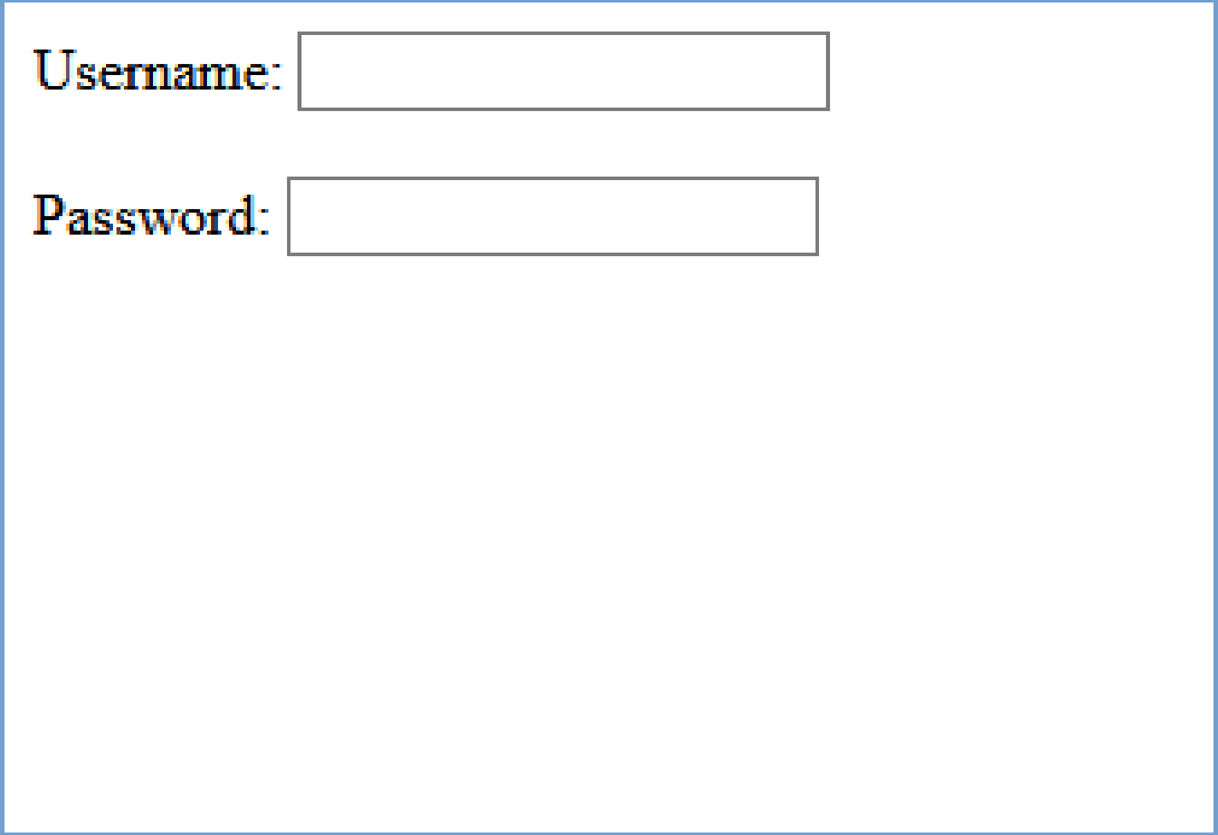
<label> (continued)

- Have next attributes (value in the parentheses):
  - for (*element_id*) – Specifies the id of the form where the label is bounded. Must be equal to the id of the related element. Also, the element might also be bounded by placing inside the <label></label>.
  - form (*form_id*) – Specifies the form where the label is bounded.

<input>

Input control types: *text*

```
<!DOCTYPE html>
<html>
<body>
<form>
  <label for="username">Username: </label>
  <input type="text" id="username" name="username">
</form>
<br>
<form>
  <label>Password:
      <input type="text" id="password" name="password">
  </label>
</form>
</body>
</html>
```

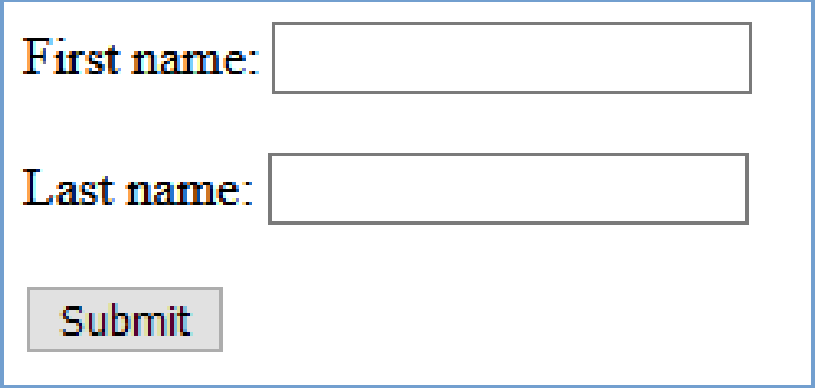
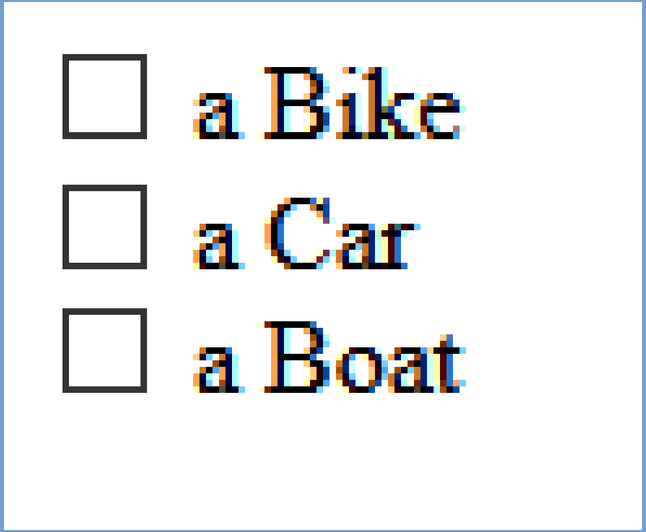


Username:

Password:

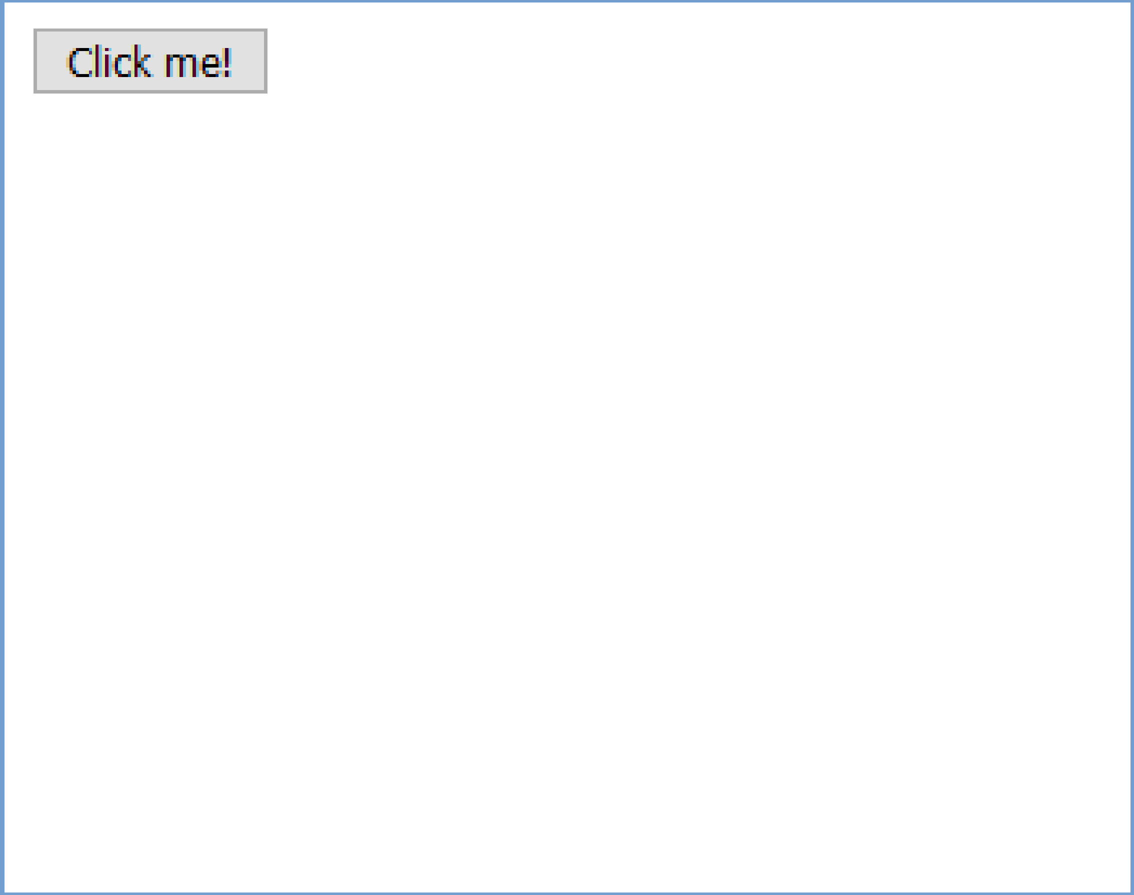
Input control types: *submit*

```
<!DOCTYPE html>
<html>
<body>

<form action="indexCheckbox.html">
  <label for="fname">First name: </label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name: </label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```


First name:
Last name:
Submit

☐ a Bike
☐ a Car
☐ a Boat

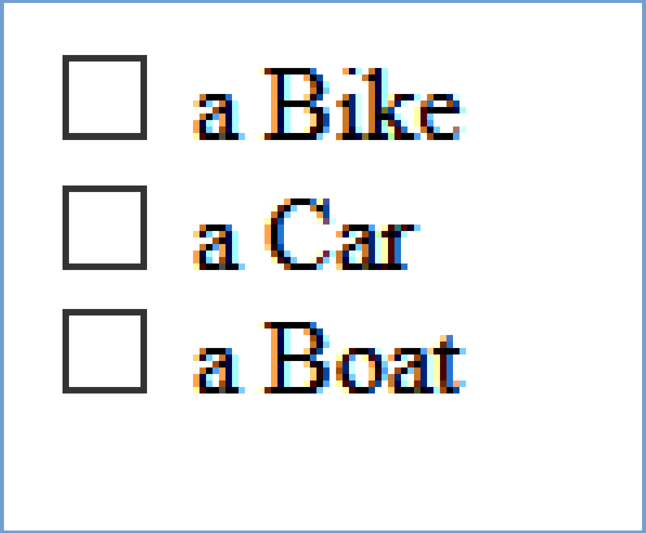
Input control types: *button*

```
<!DOCTYPE html>
<html>
<body>
<form>
<input type="button" value="Click me!">
</form>
</body>
</html>
```

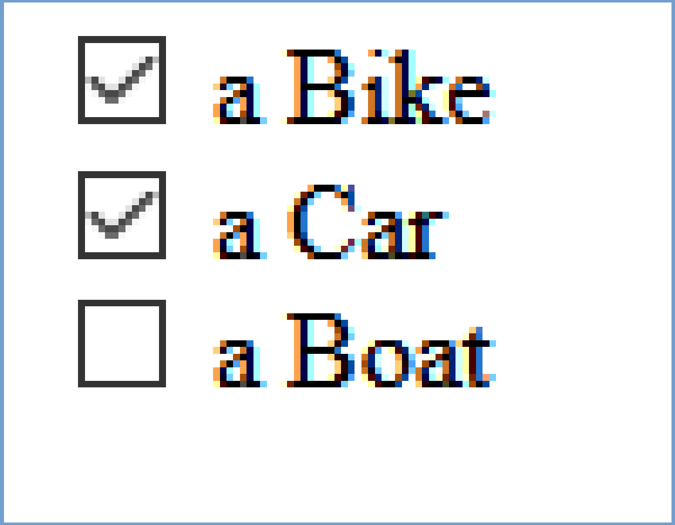

Click me!

Input control types: *checkbox*

```
<!DOCTYPE html>
<html>
<body>

<form>
<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
<label for="vehicle1">a Bike</label><br>
<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
<label for="vehicle2">a Car</label><br>
<input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
<label for="vehicle3">a Boat</label><br>
</form>

</body>
</html>
```



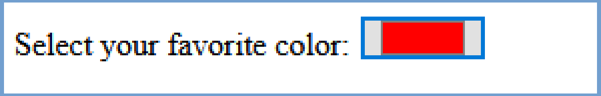
☐ a Bike
☐ a Car
☐ a Boat



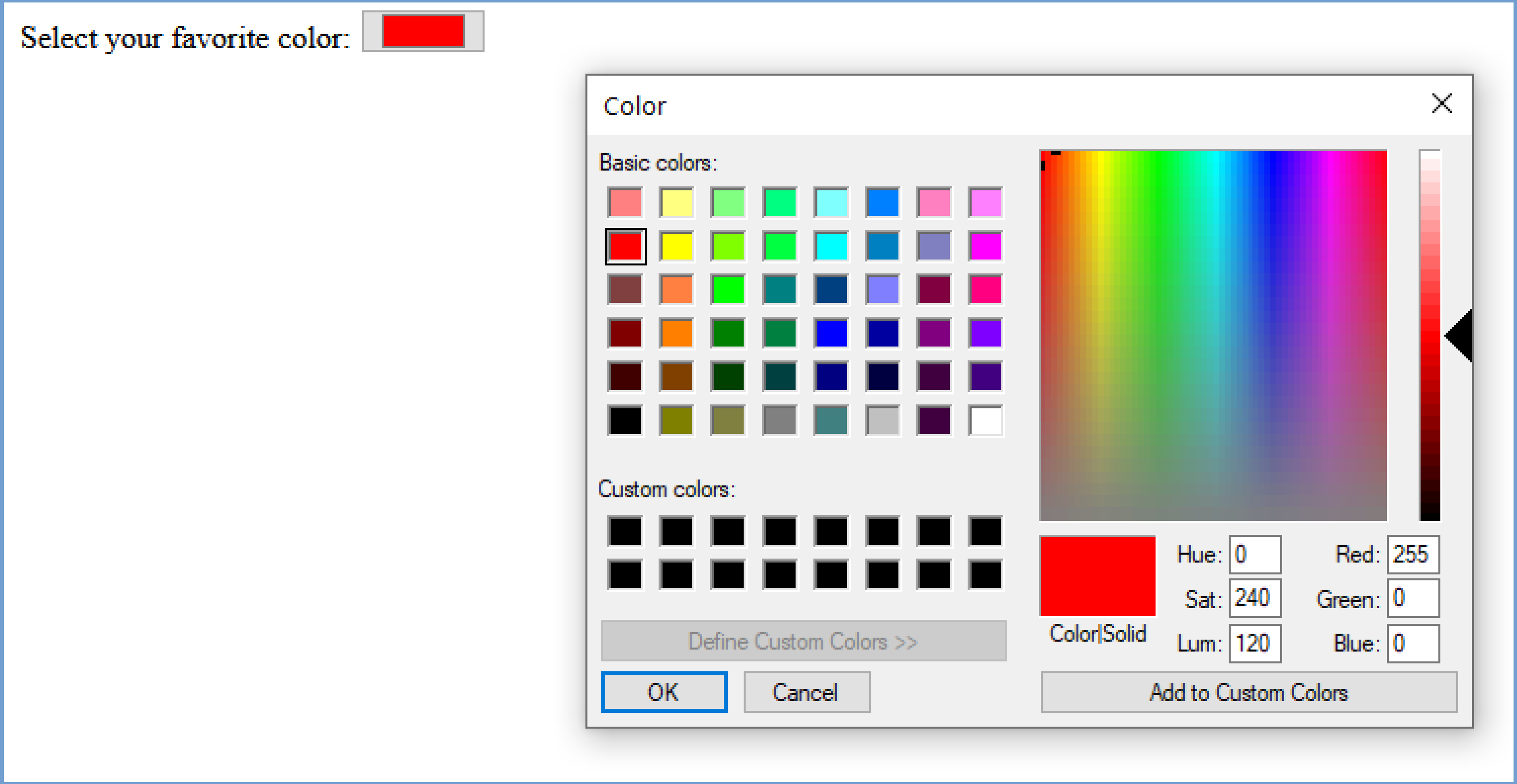
☑ a Bike
☑ a Car
☐ a Boat

Input control types: *color*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="favcolor">Select your favorite color:</label>
<input type="color" id="favcolor" name="favcolor" value="#ff0000">
</form>

</body>
</html>
```



Select your favorite color:

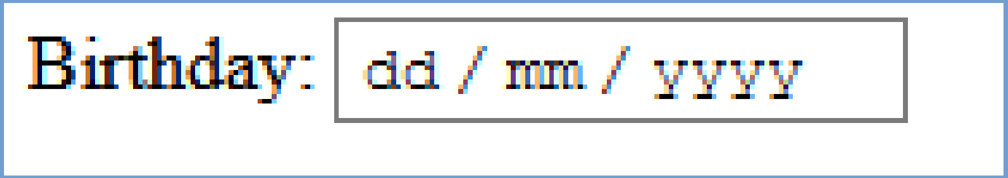


Select your favorite color:

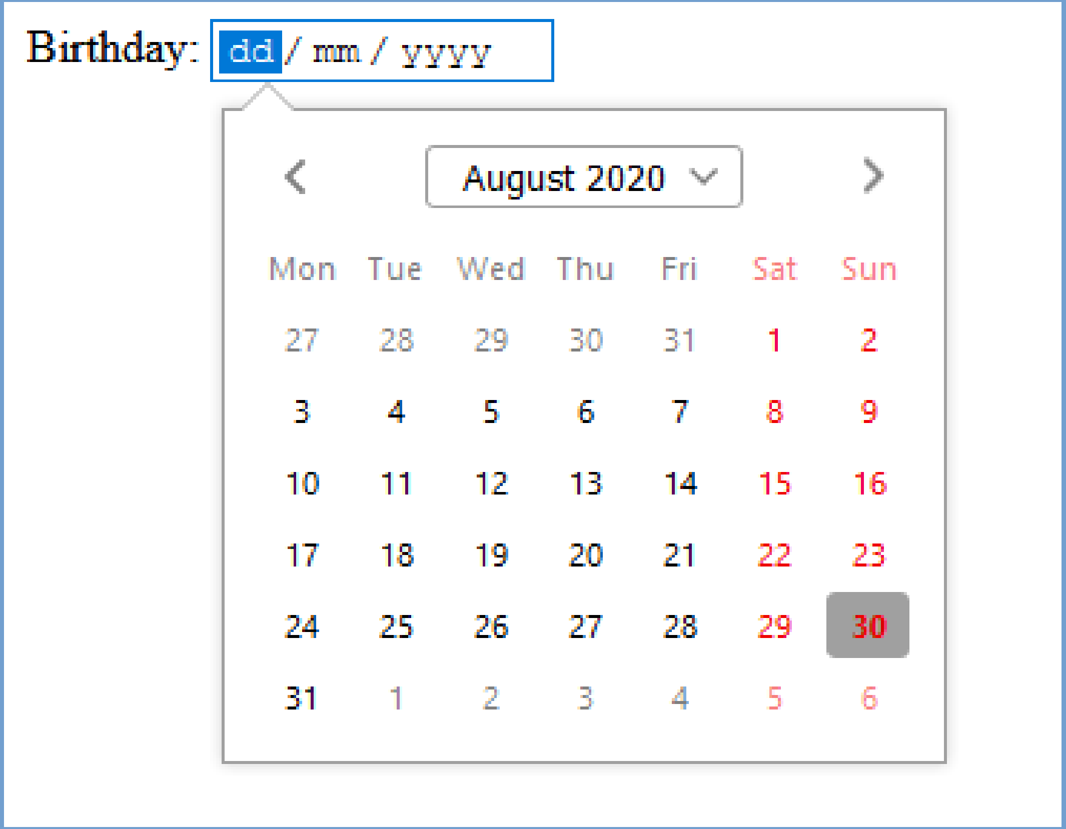
Hue: 0
Sat: 240
Lum: 120
Red: 255
Green: 0
Blue: 0

Input control types: *date*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="birthday">Birthday:</label>
<input type="date" id="birthday" name="birthday">
</form>

</body>
</html>
```



Birthday: dd / mm / yyyy



Birthday: dd / mm / yyyy

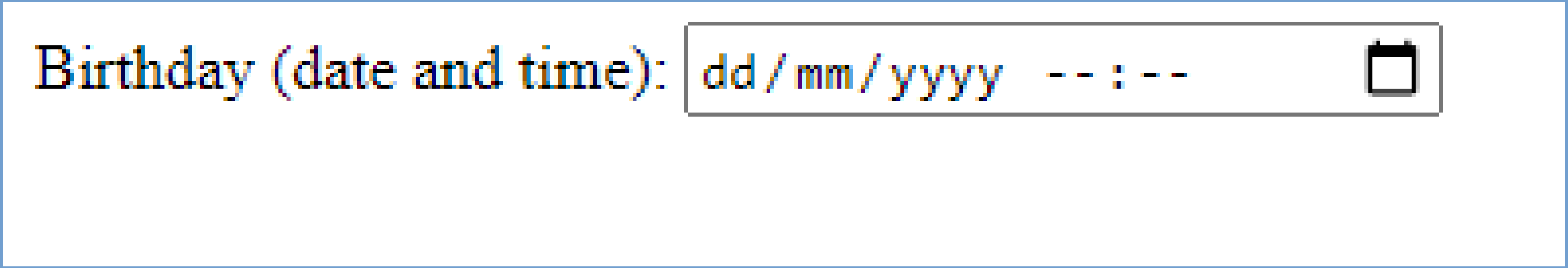
August 2020

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Input control types: *datetime-local*

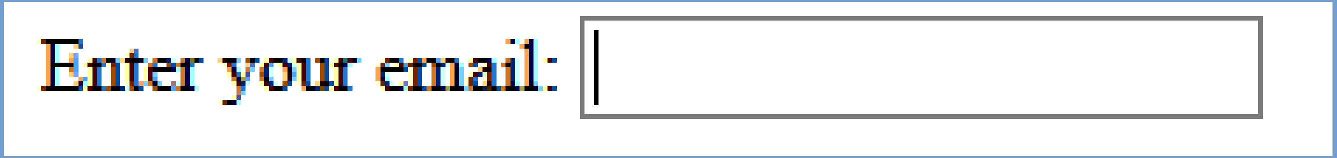
(supported in some browsers)

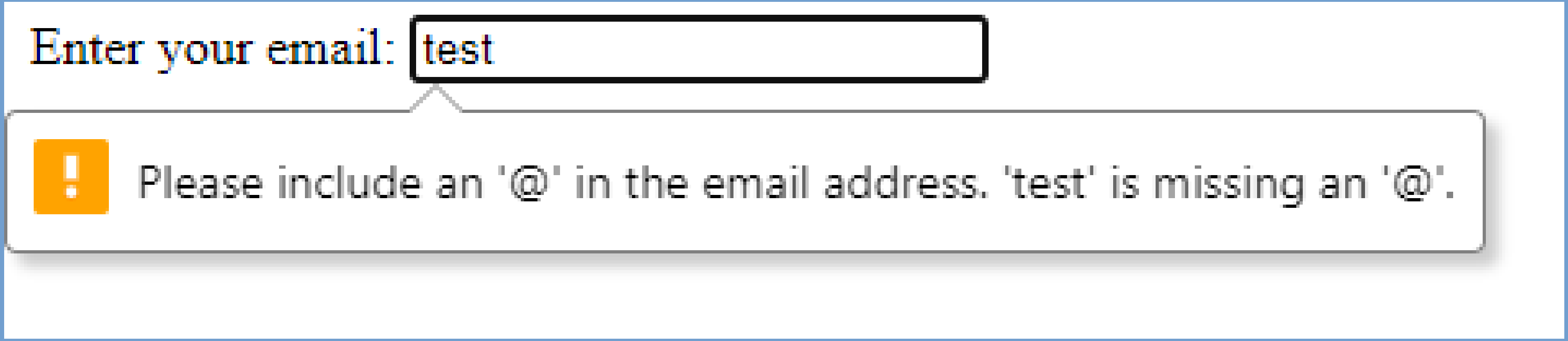
```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="birthdaytime">Birthday (date and time):</label>
<input type="datetime-local" id="birthdaytime"
name="birthdaytime">
</form>

</body>
</html>
```


Birthday (date and time): dd / mm / yyyy -- : --

Input control types: *email*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="email">Enter your email:</label>
<input type="email" id="email" name="email">
</form>

</body>
</html>
```


Enter your email:


Enter your email: test

Please include an '@' in the email address. 'test' is missing an '@'.

Input control types: *file*

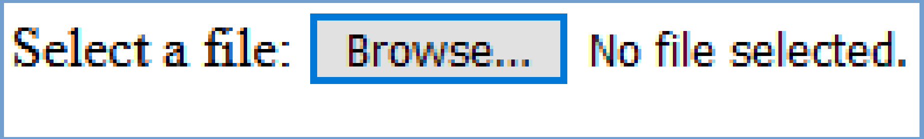
It is possible to define the type of the file by *accept* attribute, which takes one of these values (possible for other media types):

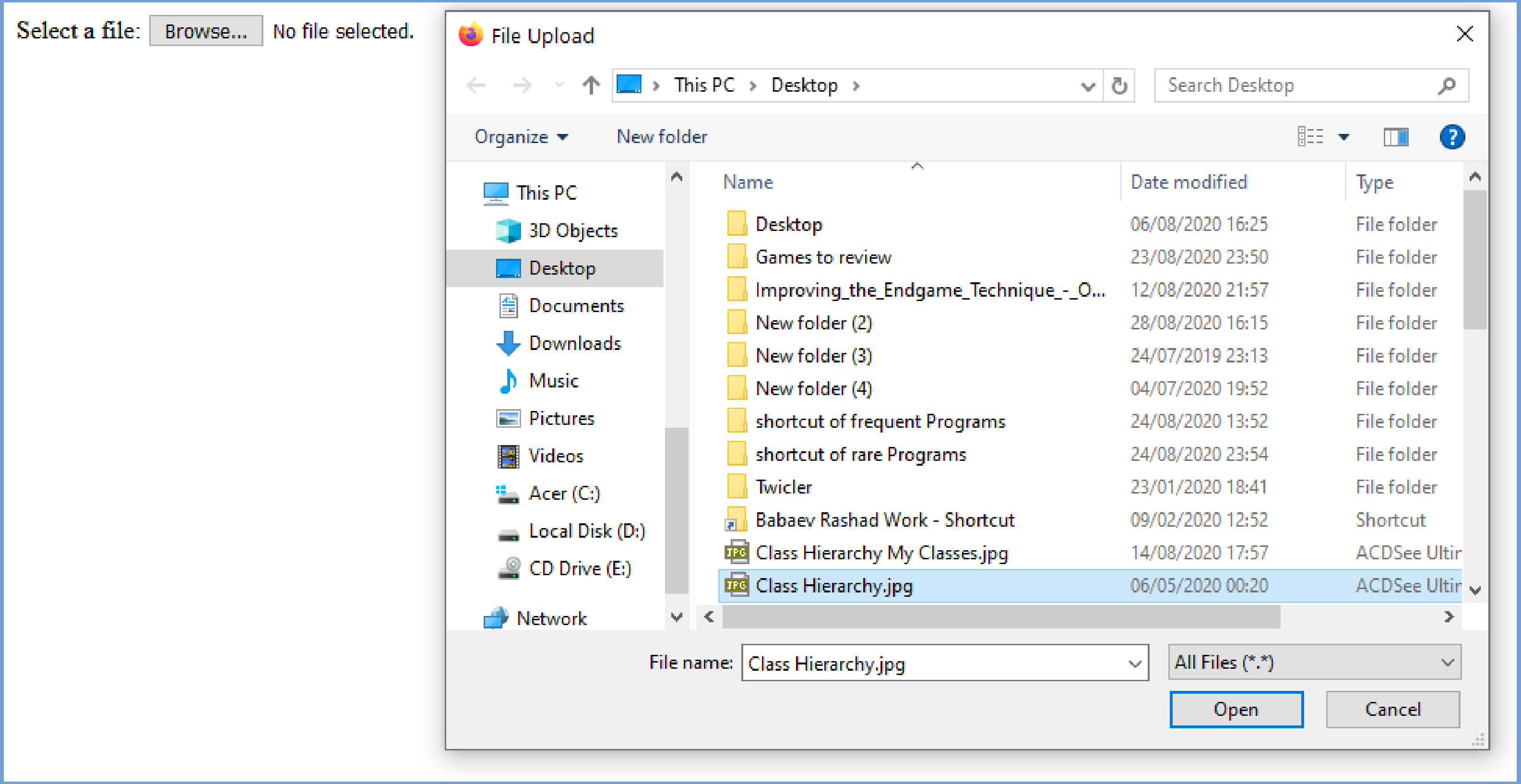
- *audio/**
- *video/**
- *image/**

Do not use this attribute as a validation tool. File uploads should be validated on the server.

Input control types: *file*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="myfile">Select a file:</label>
<input type="file" id="myfile" name="myfile">
</form>

</body>
</html>
```


Select a file: Browse... No file selected.


Select a file: Browse... No file selected.

File Upload

This PC › Desktop ›

Search Desktop

Organize ▾ New folder

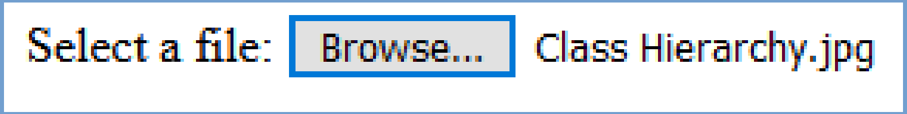
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Acer (C:)
Local Disk (D:)
CD Drive (E:)
Network

Name	Date modified	Type
Desktop	06/08/2020 16:25	File folder
Games to review	23/08/2020 23:50	File folder
Improving_the_Endgame_Technique_-_O...	12/08/2020 21:57	File folder
New folder (2)	28/08/2020 16:15	File folder
New folder (3)	24/07/2019 23:13	File folder
New folder (4)	04/07/2020 19:52	File folder
shortcut of frequent Programs	24/08/2020 13:52	File folder
shortcut of rare Programs	24/08/2020 23:54	File folder
Twicler	23/01/2020 18:41	File folder
Babaev Rashad Work - Shortcut	09/02/2020 12:52	Shortcut
Class Hierarchy My Classes.jpg	14/08/2020 17:57	ACDSee Ultir
Class Hierarchy.jpg	06/05/2020 00:20	ACDSee Ultir

File name: Class Hierarchy.jpg

All Files (*.*)

Open Cancel


Select a file: Browse... Class Hierarchy.jpg

Input control types: *hidden* (used for developers to know the column to update in the database)

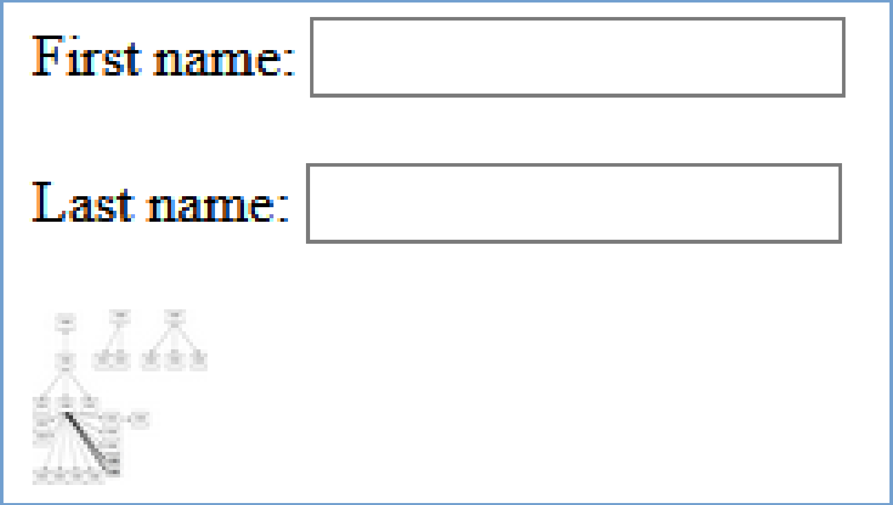
```
<!DOCTYPE html>
<html>
<body>

<form>
<input type="hidden" id="custId" name="custId" value="3487">
</form>

</body>
</html>
```

Input control types: *image*

- Might contain *alt* attribute for alternative text.
- Might contain *src* attribute for the URL of the image.
- As the *type="submit"* value, might has own encoding values by *formenctype* attribute.
- As the *type="submit"* value, might has own method by *formmethod* attribute.
- As the *type="submit"* , might has own action by *formaction* attribute.
- As the *type="submit"* , might has own target by *formtarget* attribute.
- Might define the image width and height pixels by *width* and *height* attributes, respectively.

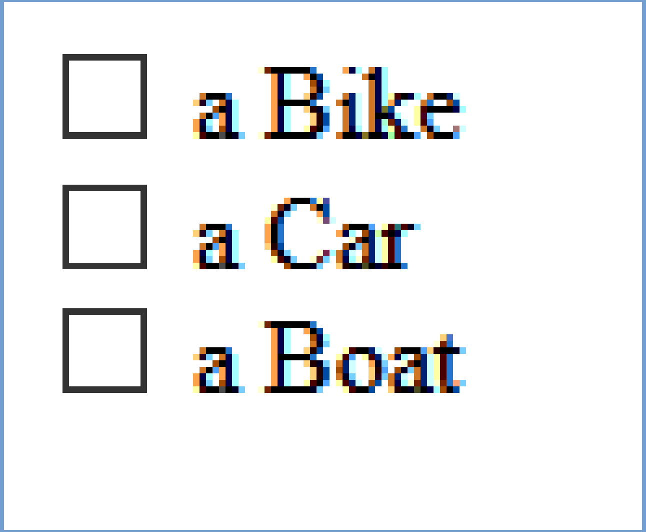
Input control types: *image*

```
<!DOCTYPE html>
<html>
<body>

<form action="indexCheckbox.html">
  <label for="fname">First name: </label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name: </label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="image" src="test.jpg" alt="Submit" width="48" height="48">
</form>

</body>
</html>
```



First name:

Last name:



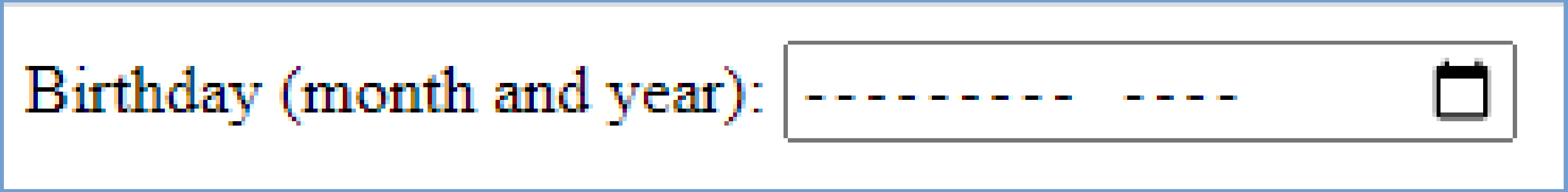
☐ a Bike

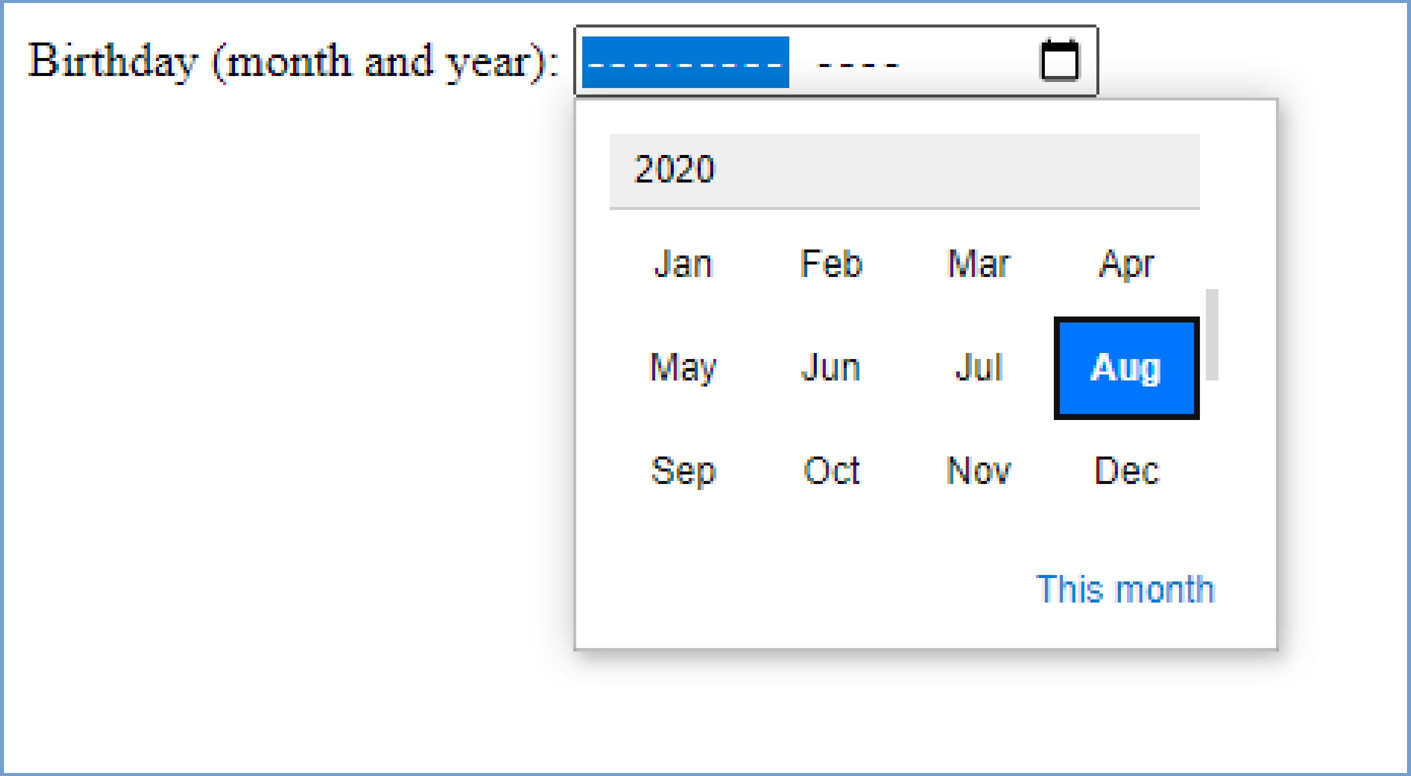
☐ a Car

☐ a Boat

Input control types: *month*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="bdaymonth">Birthday (month and year):</label>
<input type="month" id="bdaymonth" name="bdaymonth">
</form>

</body>
</html>
```


Birthday (month and year): --------- ----


Birthday (month and year): --------- ----

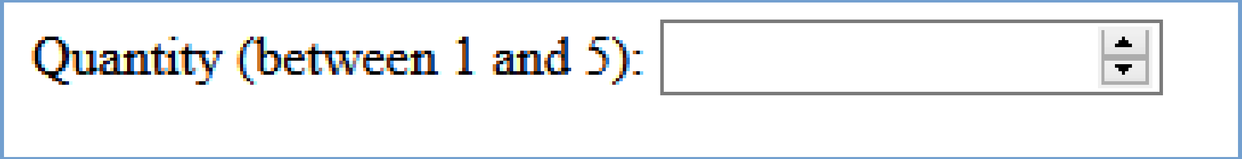
Input control types: *number*

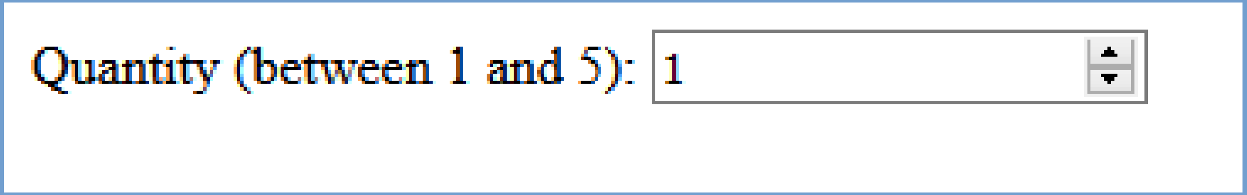
It is possible to set restrictions on what numbers are accepted with the attributes below:

- *max* - specifies the maximum value allowed;
- *min* - specifies the minimum value allowed;
- *step* - specifies the legal number intervals;
- *value* - Specifies the default value.

Input control types: *number*

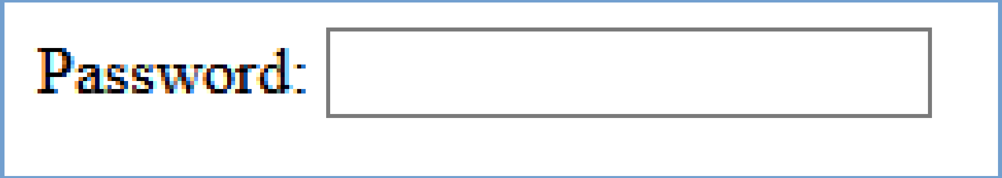
```
<!DOCTYPE html>
<html>
<body>

<form action="indexCheckbox.html">
 <label for="quantity">Quantity (between 1 and
5):</label>
<input type="number" id="quantity" name="quantity"
min="1" max="5">
</form>

</body>
</html>
```

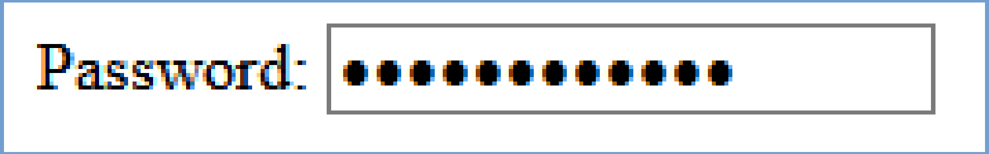

Quantity (between 1 and 5):


Quantity (between 1 and 5): 1

Input control types: *password* (should be used over HTTPS)

```
<!DOCTYPE html>
<html>
<body>

<form>
 <label for="pwd">Password:</label>
<input type="password" id="pwd" name="pwd">
</form>

</body>
</html>
```


Password:

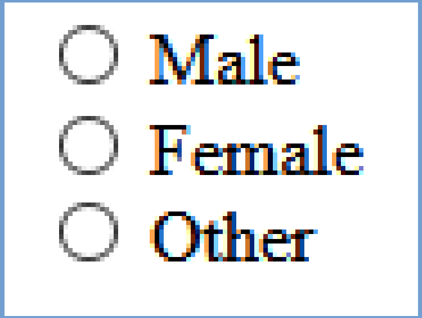

Password: ●●●●●●●●●●●●

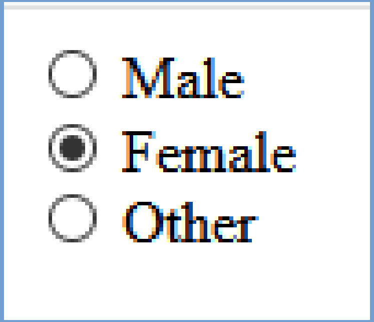
Input control types: *radio*

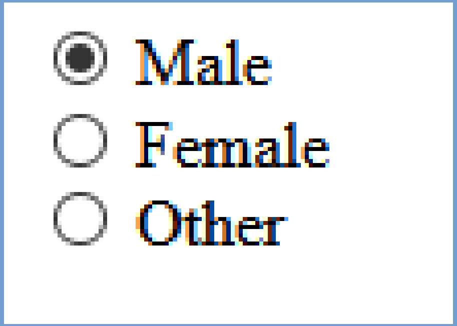
As the *type="checkbox"* value, the element might be defined as pre-selected by *checked* attribute (doesn’t require any values)

Input control types: *radio*

```
<!DOCTYPE html>
<html>
<body>

<form>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label><br>
<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label><br>
<input type="radio" id="other" name="gender" value="other">
<label for="other">Other</label>
</form>

</body>
</html>
```


○ Male
○ Female
○ Other


○ Male
[✓ Female]
○ Other


[✓ Male]
○ Female
○ Other

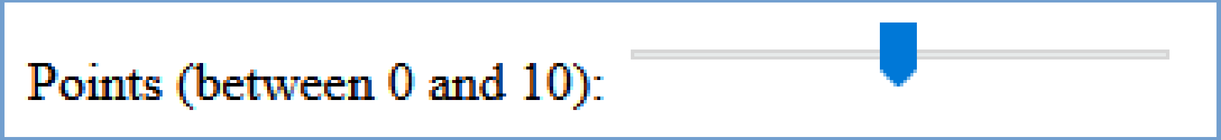
Input control types: *range*

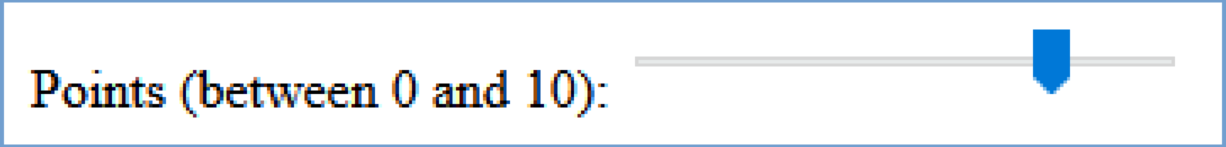
Default range is 0 to 100. However, it is possible to set restrictions on what numbers are accepted with the attributes below:

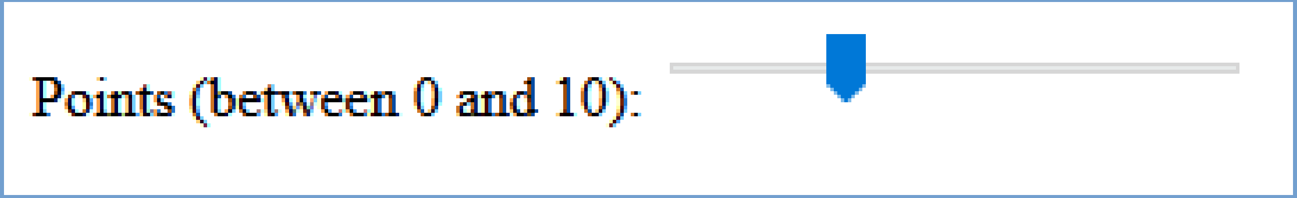
- *max* - specifies the maximum value allowed;
- *min* - specifies the minimum value allowed;
- *step* - specifies the legal number intervals;
- *value* - Specifies the default value.

Input control types: *range*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="points">Points (between 0 and 10):</label>
<input type="range" id="points" name="points" min="0"
max="10">
</form>

</body>
</html>
```

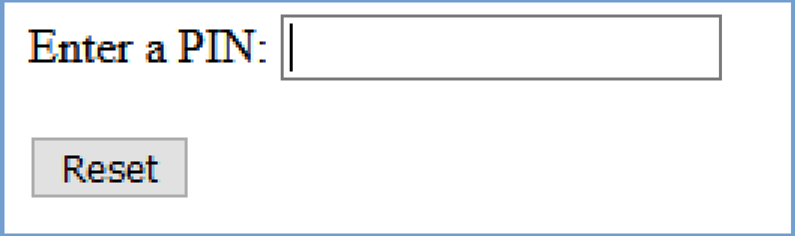

Points (between 0 and 10):


Points (between 0 and 10):

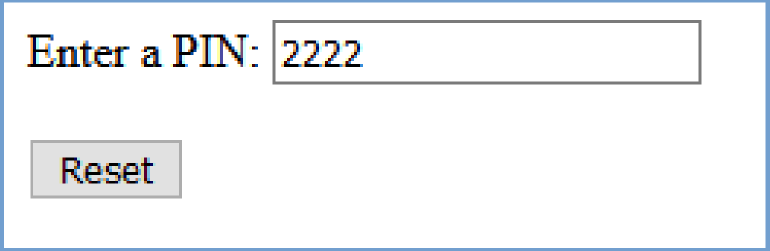

Points (between 0 and 10):

Input control types: *reset*

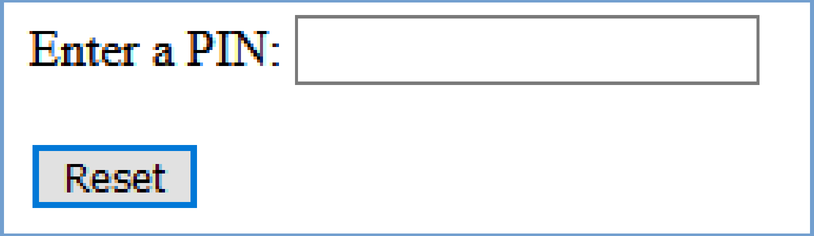
```
<!DOCTYPE html>
<html>
<body>

<form>
  <label for="pin">Enter a PIN:</label>
  <input type="text" id="pin" name="pin" maxlength="4"><br><br>

  <input type="reset" value="Reset">
</form>

</body>
</html>
```


Enter a PIN:

Reset


Enter a PIN: 2222

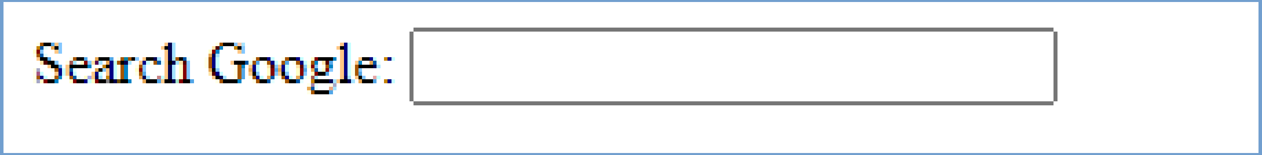
Reset


Enter a PIN:

Reset

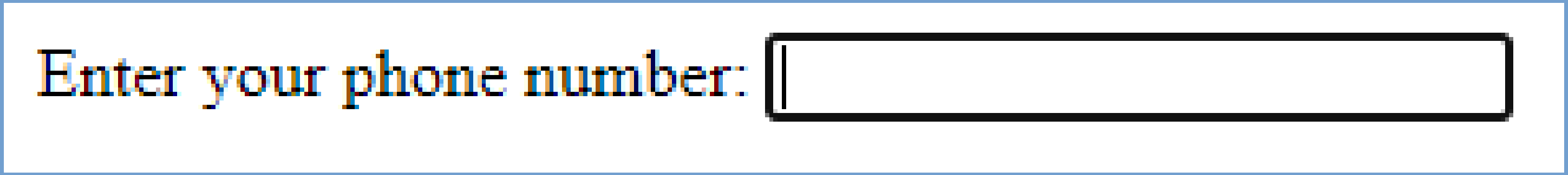
Input control types: *search* *(must have the name [commonly "q"])*

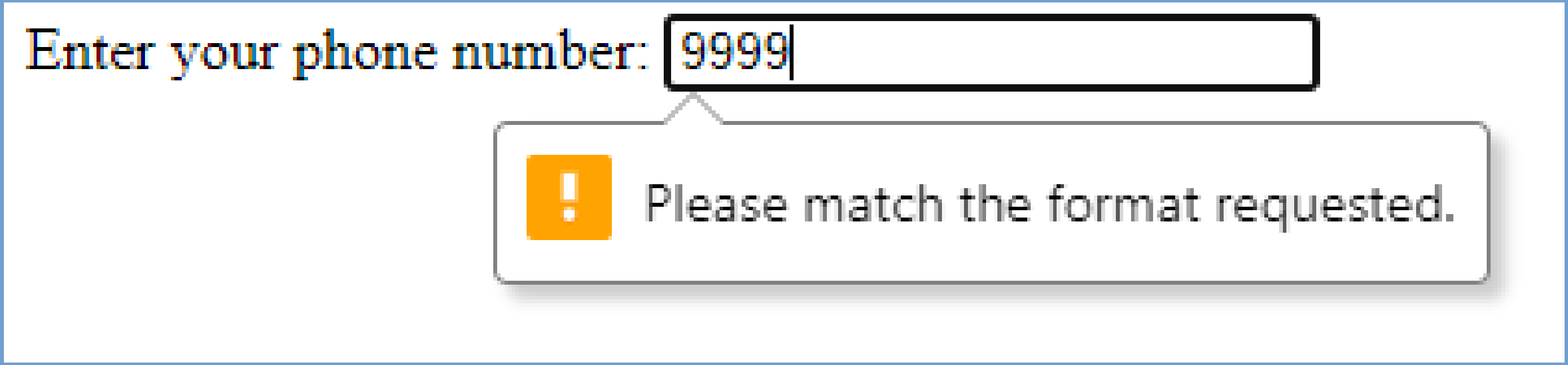
```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="gsearch">Search Google:</label>
<input type="search" id="gsearch"
name="gsearch">
</form>

</body>
</html>
```


Search Google:

Input control types: *tel*

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="phone">Enter your phone number:</label>
<input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>

</body>
</html>
```

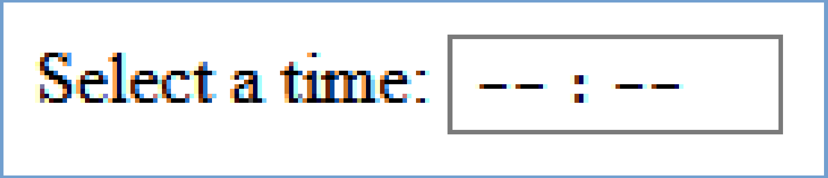

Enter your phone number:

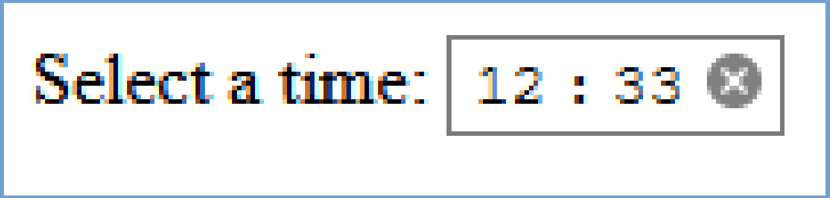

Enter your phone number: 9999

Please match the format requested.

Input control types: *time*

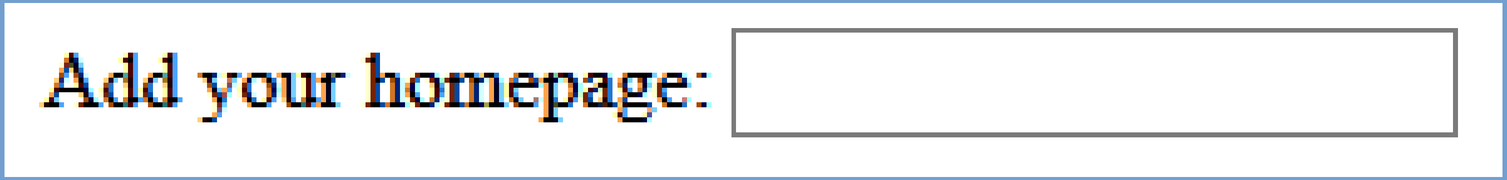
```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="appt">Select a time:</label>
<input type="time" id="appt" name="appt">
</form>

</body>
</html>
```

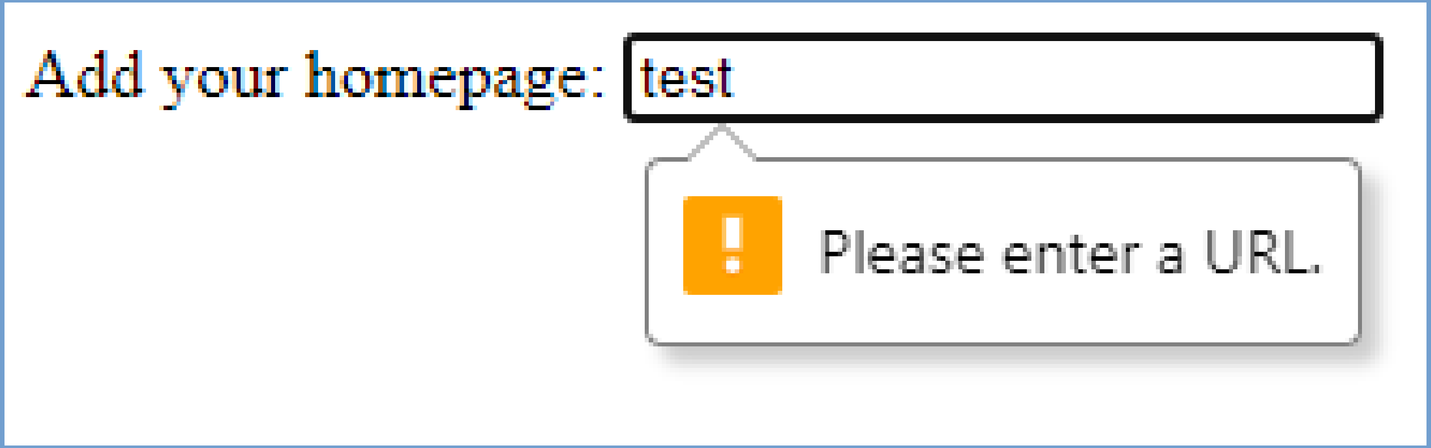

Select a time: -- : --


Select a time: 12 : 33

Input control types: *url*

```
<!DOCTYPE html>
<html>
<body>

<form>
  <label for="homepage">Add your homepage:</label>
  <input type="url" id="homepage"
name="homepage"><br><br>
</form>

</body>
</html>
```

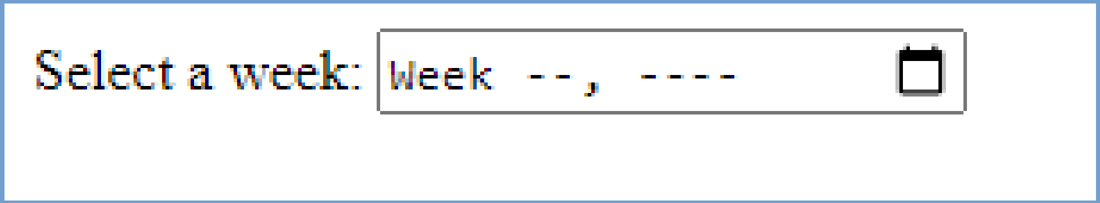

Add your homepage:


Add your homepage: test

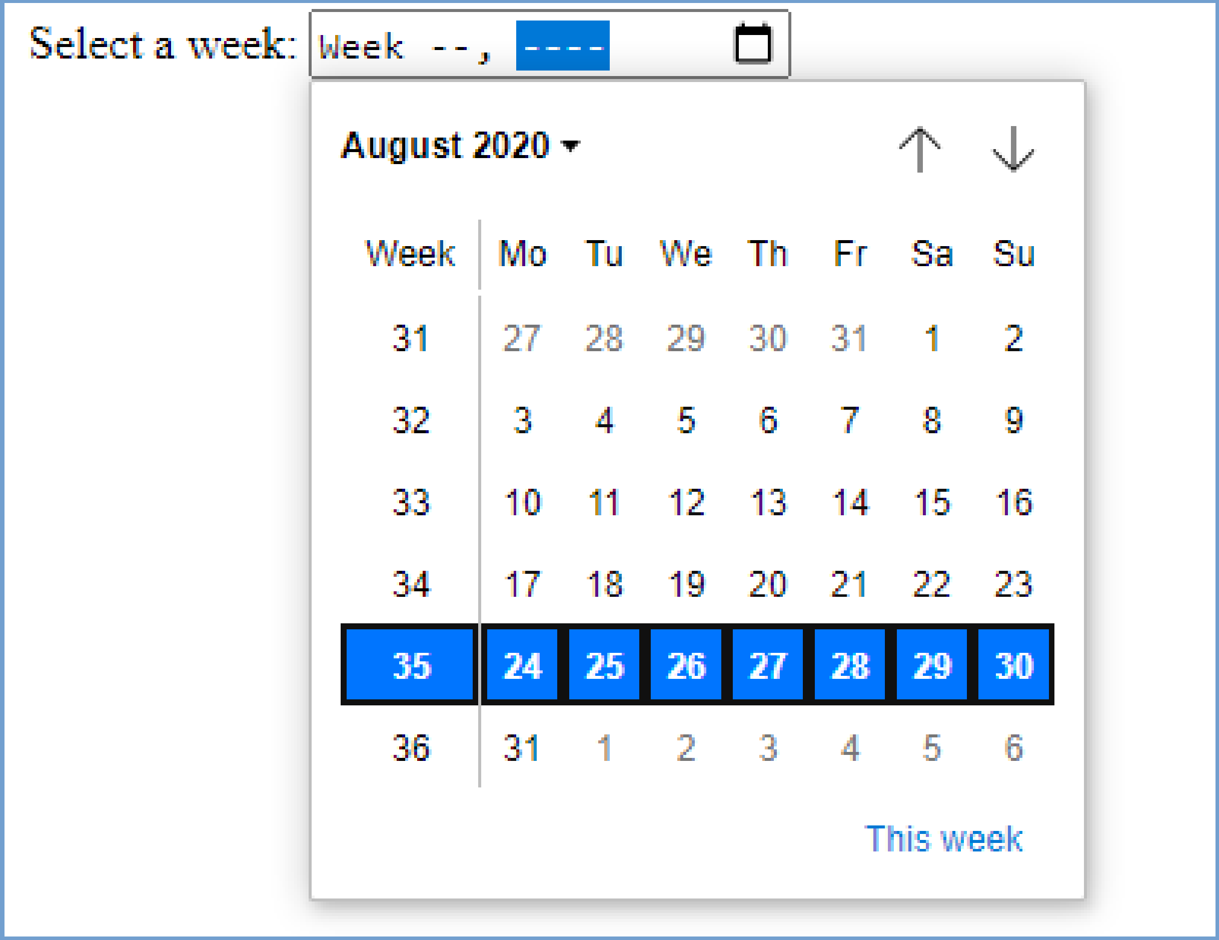
Please enter a URL.

Input control types: *week* (supported in some browsers)

```
<!DOCTYPE html>
<html>
<body>

<form>
<label for="week">Select a week:</label>
<input type="week" id="week" name="week">
</form>

</body>
</html>
```



Select a week: Week --, ----



Select a week: Week --, ----

August 2020 ▾ ↑ ↓

Week	Mo	Tu	We	Th	Fr	Sa	Su
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31	1	2	3	4	5	6

This week

Other <input> attributes (values are in the parentheses)

- autocomplete (*on* or *off*) Specifies whether an <input> element should have autocomplete enabled
- autofocus Specifies that an <input> element should automatically get focus when the page loads
- dirname (*inputname.dir*) Specifies that the text direction will be submitted
- disabled Specifies that an <input> element should be disabled
- form (*form_id*) Specifies the form the <input> element belongs to
- formnovalidate   Defines that form elements should not be validated when submitted
- list (*datalist_id*) Refers to a <datalist> element that contains pre-defined options for an <input> element

Other <input> attributes (values are in the parentheses)

- maxlength (*number*) Specifies the maximum number of characters allowed in an <input> element
- minlength (*number*) Specifies the minimum number of characters required in an <input> element
- multiple   Specifies that a user can enter more than one value in an <input> element
- name (*text*) Specifies the name of an <input> element
- pattern (*regexp*) Specifies a regular expression that an <input> element's value is checked against
- placeholder (*text*) Specifies a short hint that describes the expected value of an <input> element
- readonly   Specifies that an input field is read-only
- required Specifies that an input field must be filled out before submitting the form
- size (*number*) Specifies the width, in characters, of an <input> element
- value (*text*) Specifies the value of an <input> element

<button>

<button> tag

- Unlike *<input type="button">*, might accept the text and other tags (<i>, <strong>, <img> etc.).
- Has three values for the *type* attribute: button, submit, reset.

<button> tag

- Unlike *<input type="button">*, might accept the text and other tags (<i>, <strong>, <img> etc.).
- Has three values for the *type* attribute: button, submit, reset.

Click Me!

Other <button> attributes (values in the parentheses)

- autofocus Specifies that a button should automatically get focus when the page loads
- disabled Specifies that a button should be disabled
- form (*form_id*) Specifies which form the button belongs to
- formaction (*URL*) Specifies where to send the form-data when a form is submitted. Only for type="submit"
- formenctype (*application/x-www-form-urlencoded* or *multipart/form-data* or *text/plain*) Specifies how form-data should be encoded before sending it to a server. Only for type="submit"

Other <button> attributes (values in the parentheses) (continued)

- formmethod (*get* or *post*) Specifies how to send the form-data (which HTTP method to use). Only for type="submit"
- formnovalidate Specifies that the form-data should not be validated on submission. Only for type="submit"
- formtarget (*_blank* or *_self* or *_parent* or *_top* or *framename*) Specifies where to display the response after submitting the form. Only for type="submit"
- name (*name*) Specifies a name for the button
- type (*button* or *reset* or *submit*) Specifies the type of button
- value (*text*) Specifies an initial value for the button

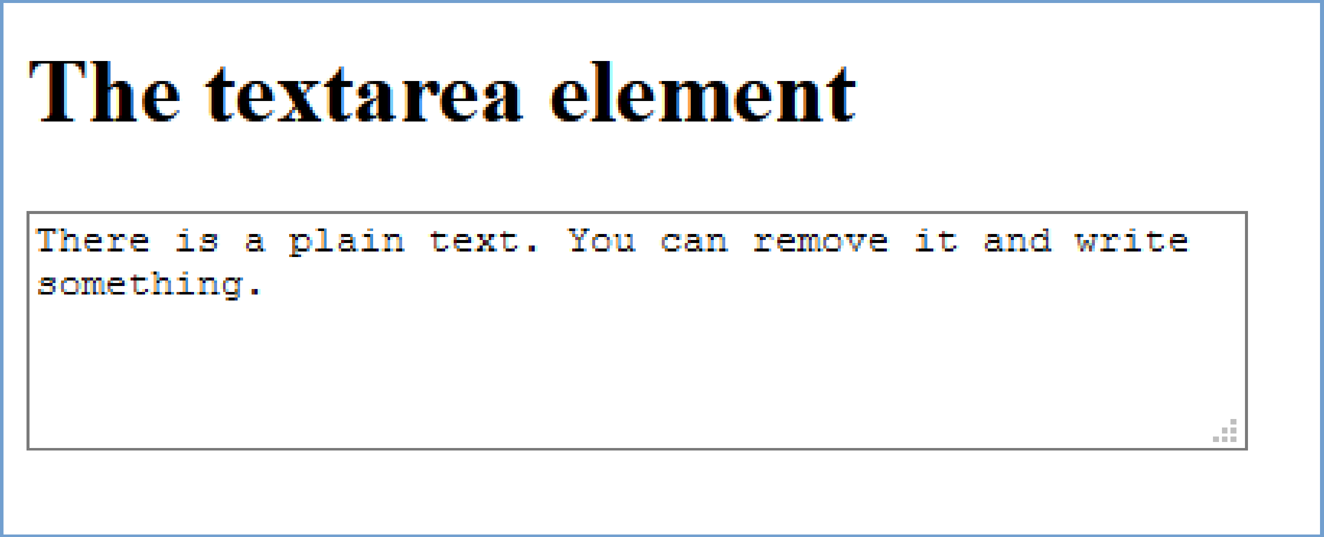
<textarea>

<textarea> tag

- Creates the multi-line text area.
- Might also be adjusted after being created.
- The number of visible rows in the area might be adjusted by *rows* attribute.
- The number of visible width in the area might be adjusted by *cols* attribute.

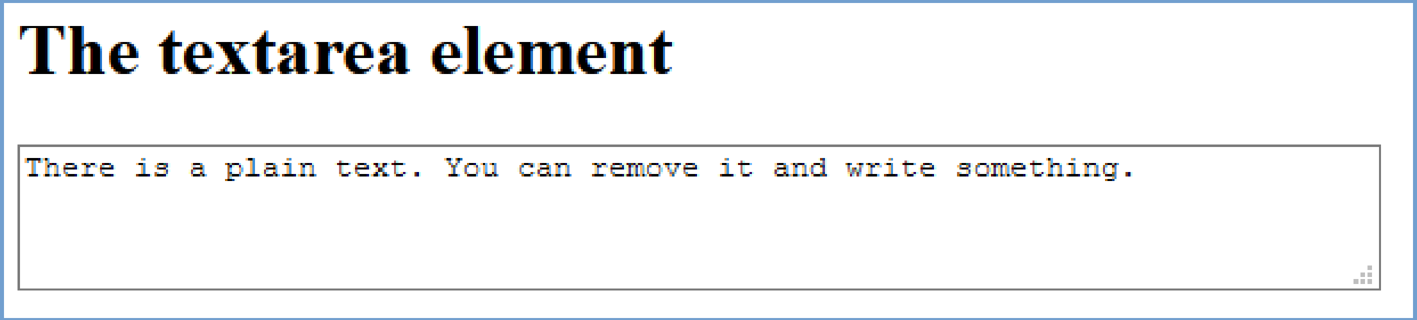
<textarea> tag

```
<!DOCTYPE html>
<html>
<body>

<h1>The textarea element</h1>

<form>
<textarea id="presentation" name="presentation" rows="4" cols="50">
There is a plain text. You can remove it and write something.</textarea>
</form>
</body>
</html>
```



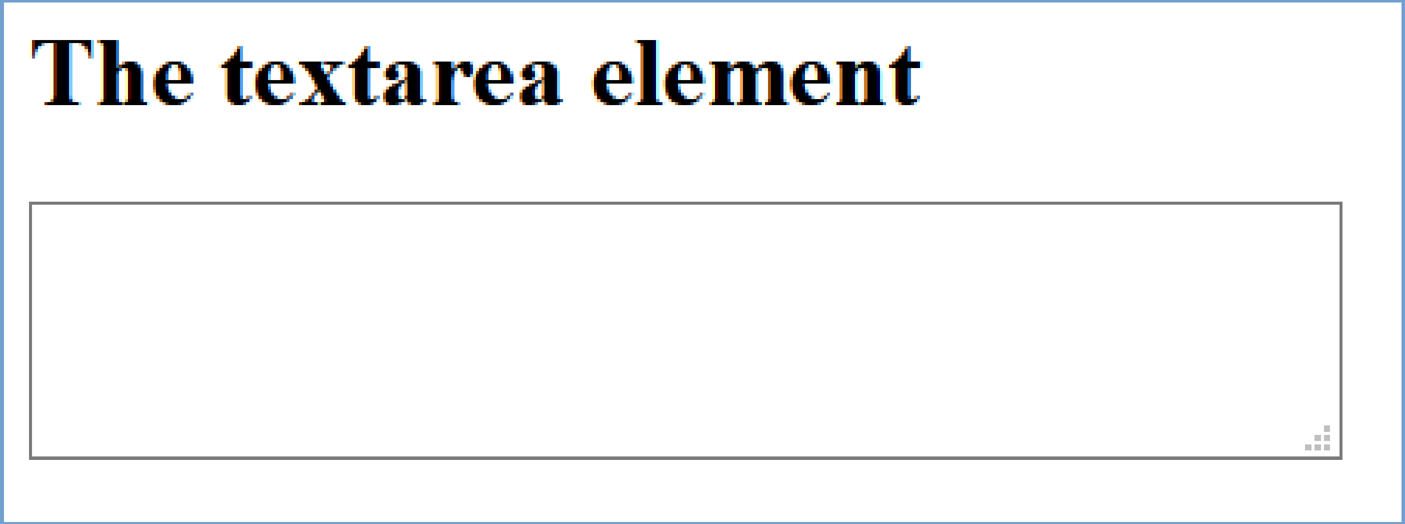
The textarea element

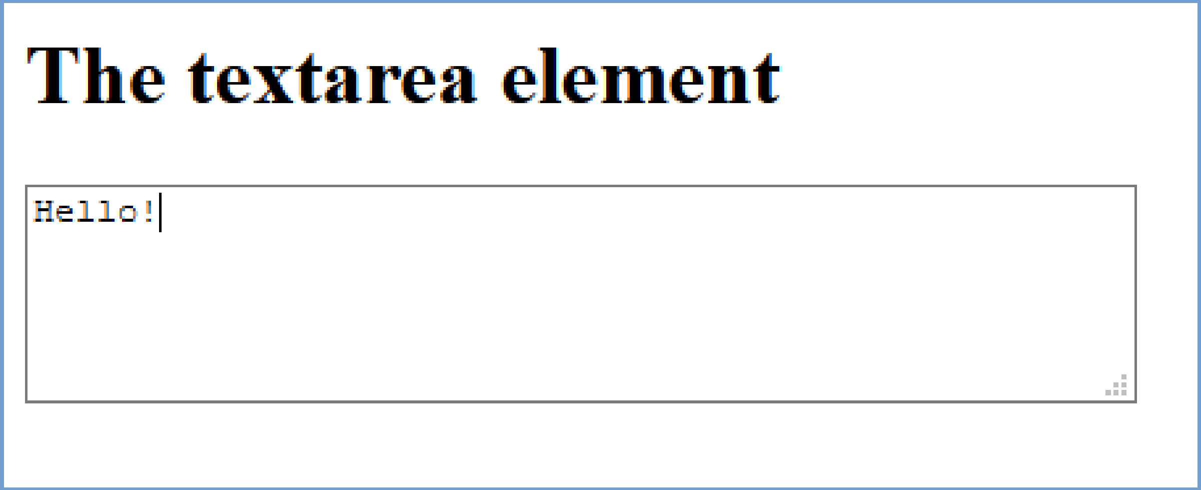
There is a plain text. You can remove it and write something.



The textarea element

There is a plain text. You can remove it and write something.


The textarea element


The textarea element

Hello!

Other <textarea> attributes (values in the parentheses)

- autofocus Specifies that a text area should automatically get focus when the page loads.
- dirname (*textareaname.dir*) Specifies that the text direction of the textarea will be submitted.
- disabled Specifies that a text area should be disabled.
- form (*form_id*) Specifies which form the text area belongs to.

Other <textarea> attributes (values in the parentheses) (continued)

- maxlength (*number*) Specifies the maximum number of characters allowed in the text area
- name (*text*) Specifies a name for a text area
- placeholder (*text*) Specifies a short hint that describes the expected value of a text area
- readonly Specifies that a text area should be read-only
- required Specifies that a text area is required/must be filled out
- wrap (*hard* or *soft*) Specifies how the text in a text area is to be wrapped when submitted in a form

<option>

<option> Tag

- Defines the options in the list.
- Usually goes inside <select>, <optgroup>, <datalist> tags.
- Should have *value* attribute if it is used lonely.

<option> attributes (values in the parentheses)

- disabled   Specifies that an option should be disabled
- label (*text*) Specifies a shorter label for an option
- selected   Specifies that an option should be pre-selected when the page loads
- value (*text*) Specifies the value to be sent to a server

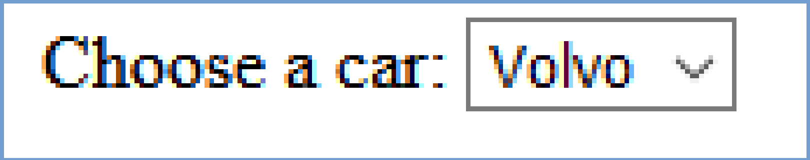
<select>

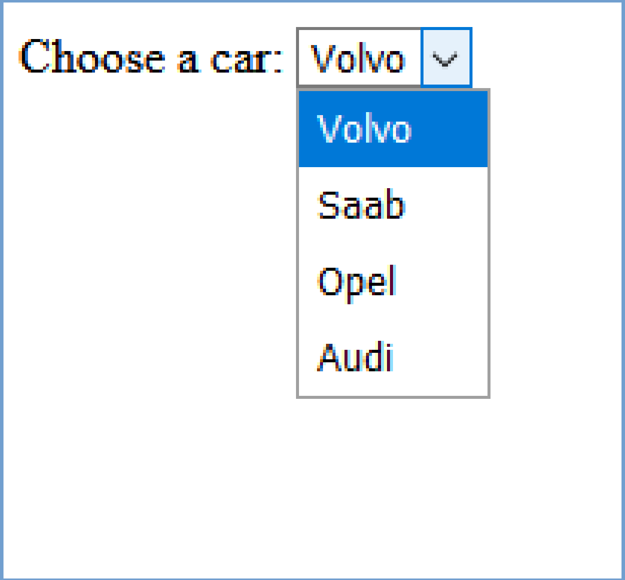
<select> tag

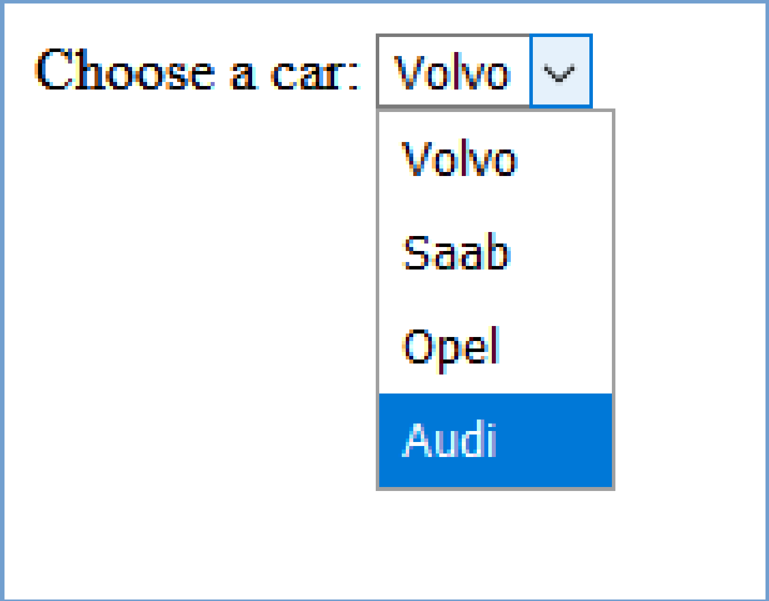
- Creates a drop-down list.
- Often used to collect data.
- To submit the data, it is important to reference the form data by *name* attribute.
- To associate with the label, *id* attribute is required.

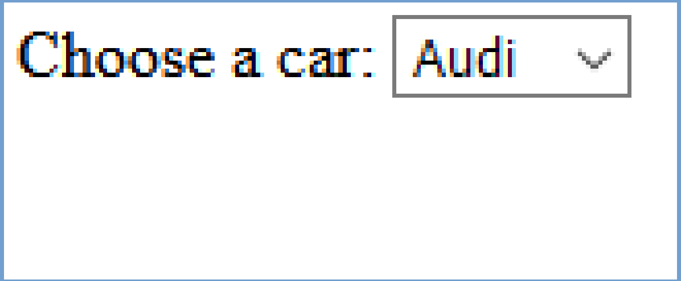
<select> tag

```
<!DOCTYPE html>
<html>
<body>
<label for="cars">Choose a car:</label>

<select id="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="opel">Opel</option>
<option value="audi">Audi</option>
</select>
</body>
</html>
```


Choose a car:
Volvo


Choose a car:
Volvo
Volvo
Saab
Opel
Audi


Choose a car:
Volvo
Volvo
Saab
Opel
Audi


Choose a car:
Audi

Other <select> attributes (values in the parentheses)

- autofocus Specifies that the drop-down list should automatically get focus when the page loads
- disabled Specifies that a drop-down list should be disabled
- form (*form_id*) Defines which form the drop-down list belongs to
- multiple Specifies that multiple options can be selected at once. To select it, Ctrl+Mouseclick might be used.
- name (*name*) Defines a name for the drop-down list
- required Specifies that the user is required to select a value before submitting the form
- size (*number*) Defines the number of visible options in a drop-down list

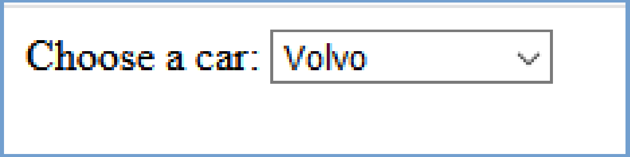
<optgroup>

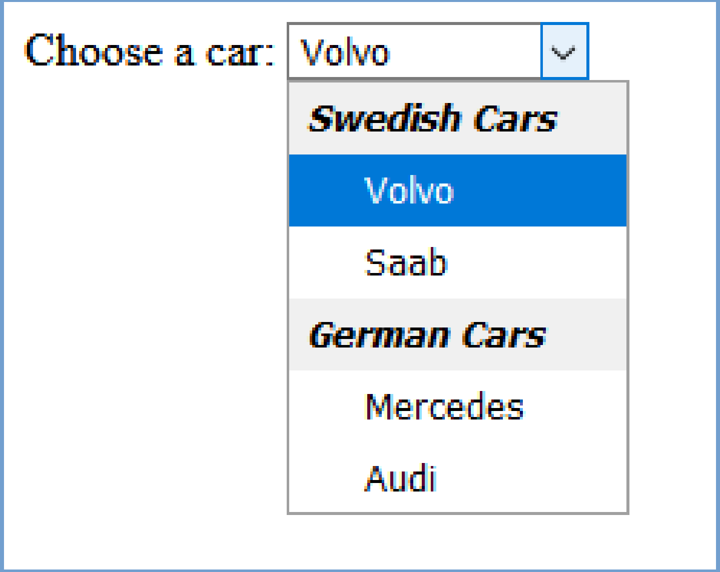
<optgroup> tag

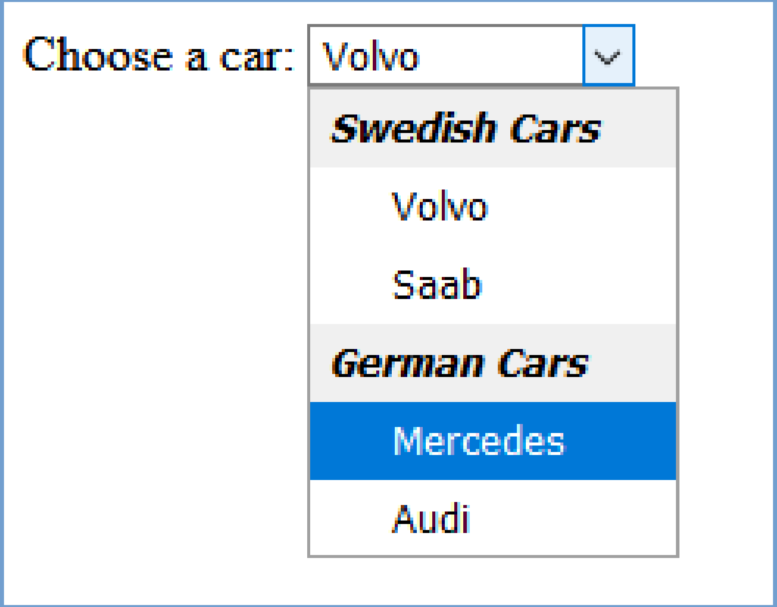
- Is used to group the tag-related options in the <select> tag.
- Makes the distinguishing the options easier.

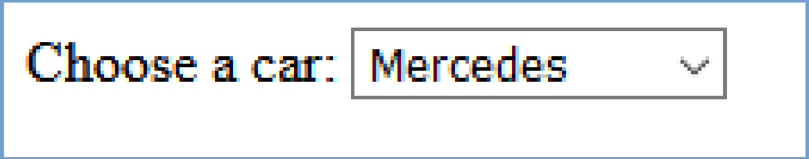
<optgroup> tag

```
<!DOCTYPE html>
<html>
<body>
 <label for="cars">Choose a car:</label>
<select  name="cars" id="cars">
  <optgroup label="Swedish Cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </optgroup>
  <optgroup label="German Cars">
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
</body>
</html>
```


Choose a car:
Volvo


Choose a car:
Volvo
Swedish Cars
Volvo
Saab
German Cars
Mercedes
Audi


Choose a car:
Volvo
Swedish Cars
Volvo
Saab
German Cars
Mercedes
Audi


Choose a car:
Mercedes

<optgroup> attributes (values in the parentheses)

- disabled Specifies that an option-group should be disabled
- label (text) Specifies a label for an option-group

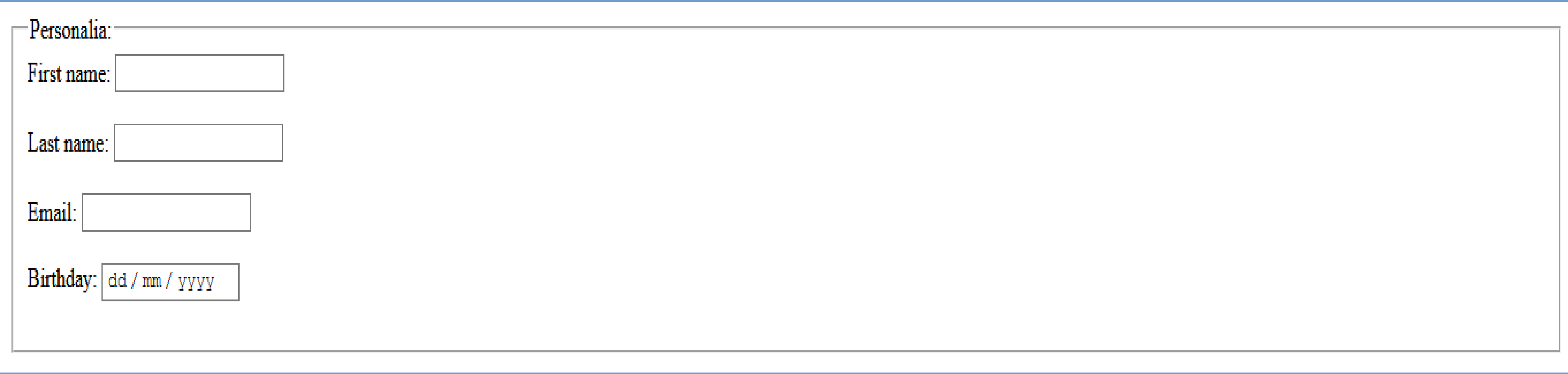
<fieldset>

<fieldset> tag

- The <fieldset> tag is used to group the elements in the <form>
- The <fieldset> tag draws the box around the tag.
- The <legend> tag is used to define a caption for the <fieldset> element.

<fieldset> tag

```
<!DOCTYPE html>
<html>
<body>
<form>
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname"><br><br>
    <label for="lname">Last name:</label>
    <input type="text" id="lname" name="lname"><br><br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email"><br><br>
    <label for="birthday">Birthday:</label>
    <input type="date" id="birthday" name="birthday"><br><br>
</fieldset>
</form>
</select>
</body>
</html>
```


Personalia:

First name:

Last name:

Email:

Birthday: dd / mm / yyyy

<fieldset> attributes (values in the parentheses)

- disabled Specifies that a group of related form elements should be disabled
- form (*form_id*) Specifies which form the fieldset belongs to
- name (*text*) Specifies a name for the fieldset

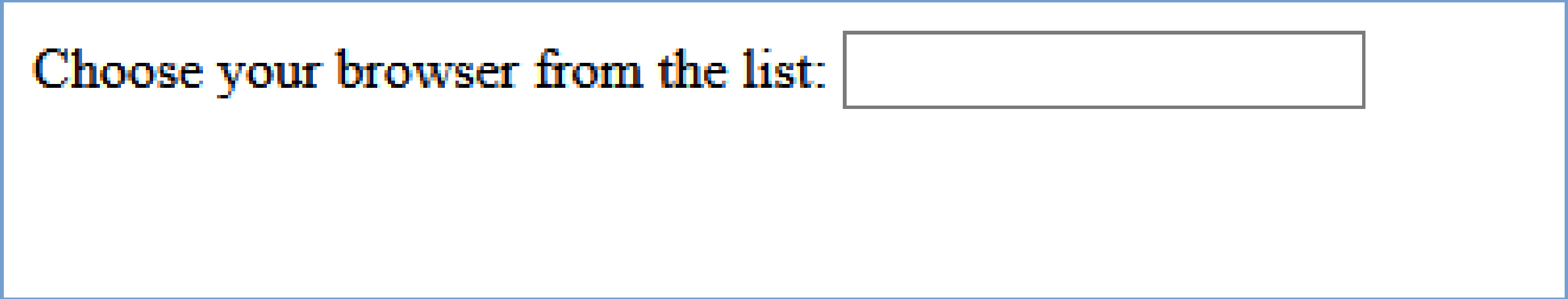
<datalist>

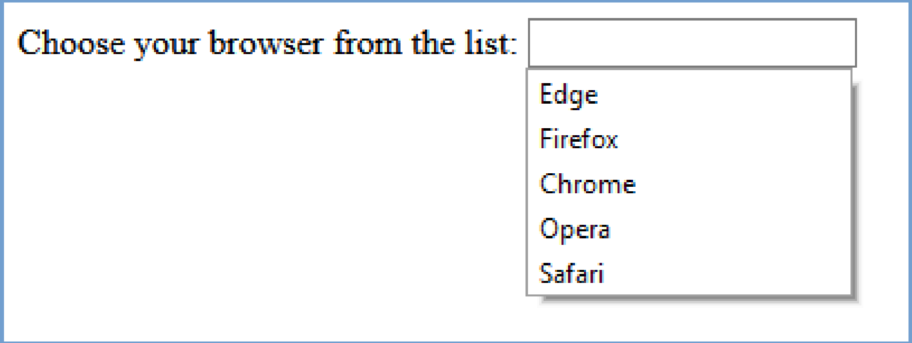
<datalist> tag

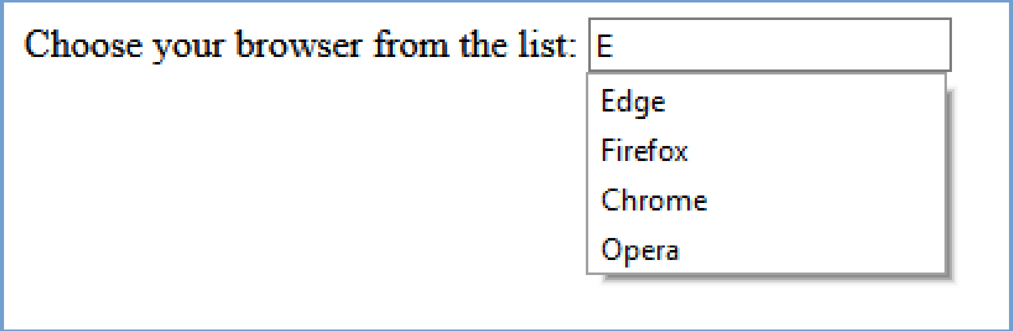
- Specifies a list of pre-defined options for an <input> element. It is done by "autocomplete" feature for <input> elements. Users will see a drop-down list of pre-defined options as they input data.
- The <datalist> element's id attribute must be equal to the <input> element's list attribute (this binds them together).

<datalist> tag

```
<!DOCTYPE html>
<html>
<body>
<label for="browser">Choose your browser from the list:</label>
<input list="browsers" name="browser" id="browser">

<datalist id="browsers">
  <option value="Edge">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
</select>
</body>
</html>
```


Choose your browser from the list:


Choose your browser from the list:


Choose your browser from the list: E

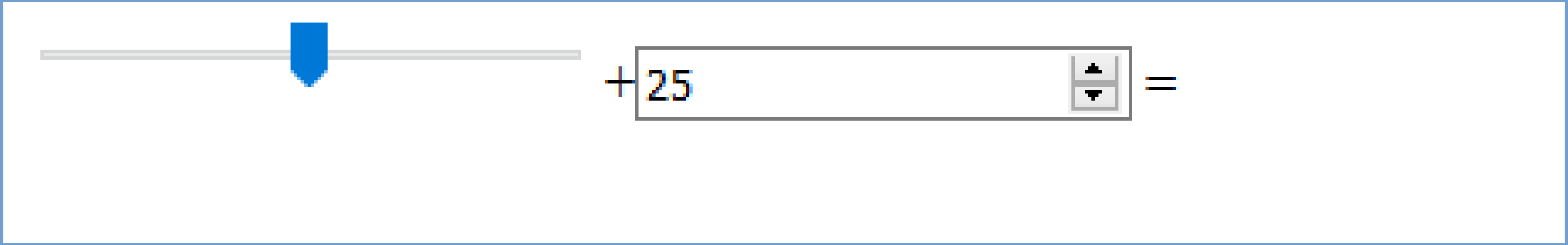
<output>

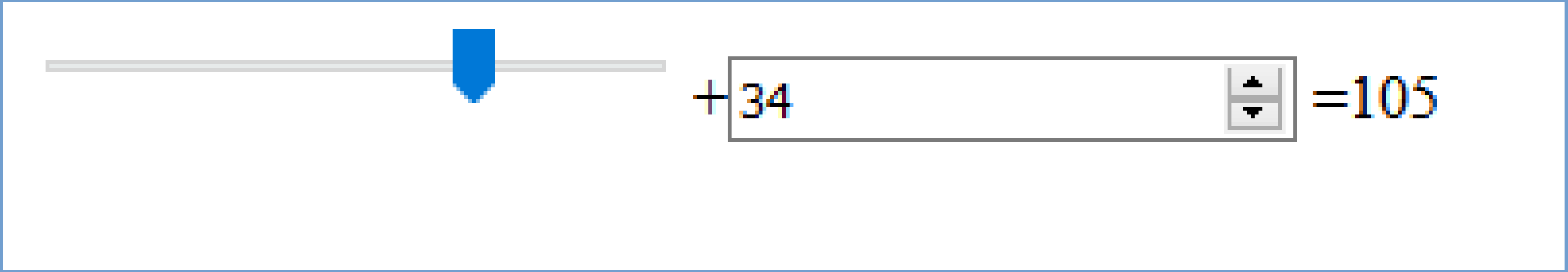
<output> tag

- Performs a calculation and describes the result.

<output> tag

```
<!DOCTYPE html>
<html>
<body>
 <form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="a" value="50">
  +<input type="number" id="b" value="25">
  =<output name="x" for="a b"></output>
</form>
</select>
</body>
</html>
```


+25
=


+34
=105

<output> tag attributes

- for (element_id) Specifies the relationship between the result of the calculation, and the elements used in the calculation
- form (form_id) Specifies which form the output element belongs to
- name (name) Specifies a name for the output element

PHP

PHP

- High-level, interpreter-based programming language.
- The files for PHP code has .php suffix. Some servers might execute the PHP code from .html files.
- Free, widely used, and open source.
- The scripts are executed in the server.

PHP code structure

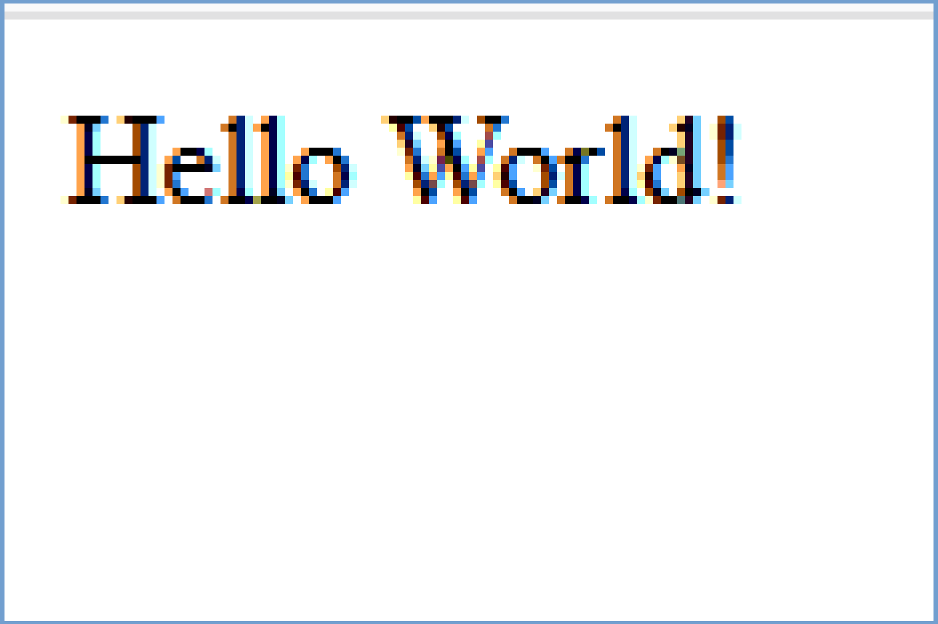
```
<?php

/* The commands are placed between brackets.
This is a multi-line comment */

// This is a single-line comment
# This is also a single-line comment

?>
```

PHP output

- The display is done by **echo** command (the alternative is **print**). For example,

```
<?php
echo "<p>Hello World!</p><br>";
?>
```

- The statement is terminated by semicolon **;**
- While the keywords are not case sensitive, **the variables are case sensitive**.

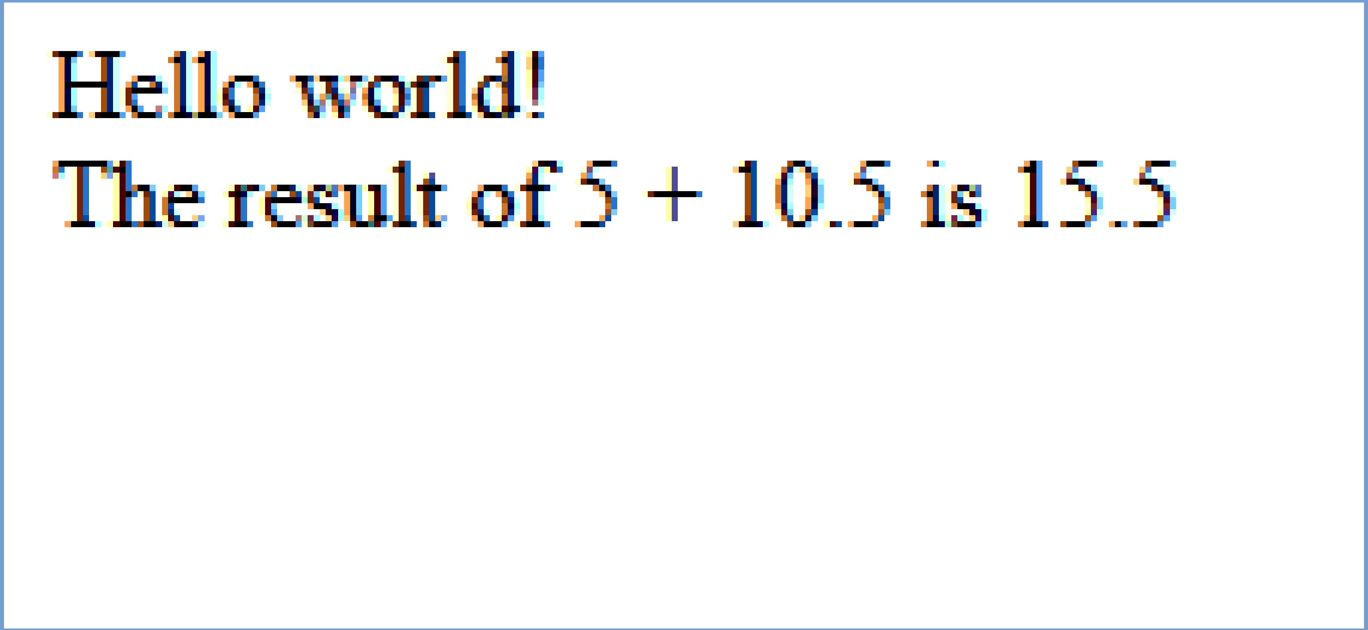


Hello World!

PHP variables

- The variables starting with **$** sign. For example, variable test is **$test**
- All variables have a dynamic data type. It means that PHP itself defines the type of data (String, Integer, Float/Double, Boolean, Array, Object, NULL, Resource)
- The variables can be used inside and outside of string in echo

-
  ```
  <?php

  $txt = "Hello world!";

  $x = 5;

  $y = 10.5;

  echo $txt;

  echo "The result of $x + $y is" . $coLOR . "<br>";

  ?>
  ```



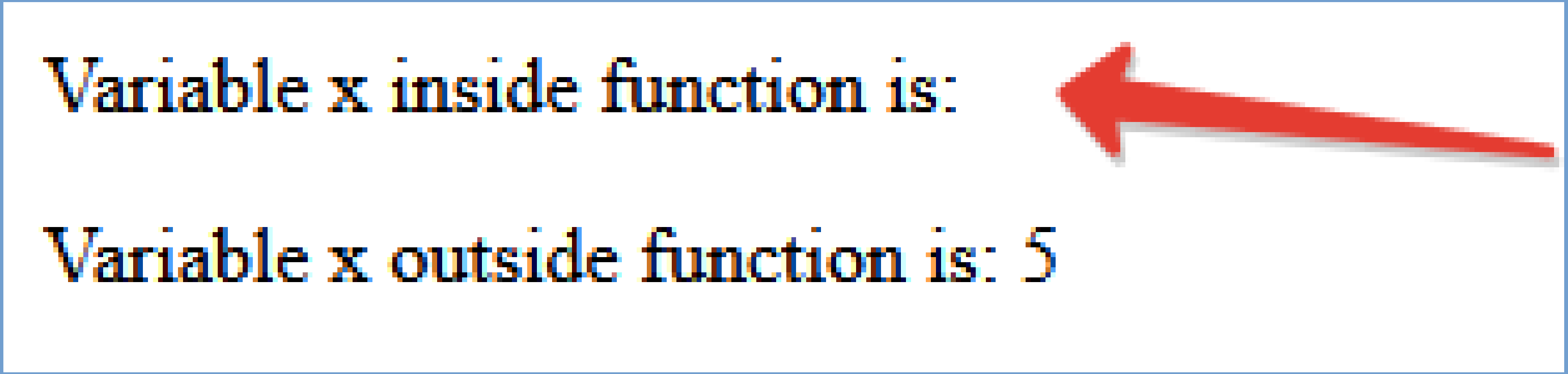
```
Hello world!
The result of 5 + 10.5 is 15.5
```

PHP scopes

- There are 3 scopes of variables:
  - *Local*
  - *Global*
  - *Static*
- If the variable defined **inside** the function, its scope is **local**. Local variables can be accessed only **inside of the function.**
- If the variable defined **outside** the function, its scope is **global**. The global variables can be accessed only **outside of the function**. If such an access should be done, ***global*** **keyword should be used**. All global variables are stored in an array called $**GLOBALS[*index*]**, where *index* is the name of the variable.
- Static variables are local variables kept after the function execution. **static** word should be added before such a variable.

PHP global scope

```
 <?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```



```
Variable x inside function is:

Variable x outside function is: 5
```

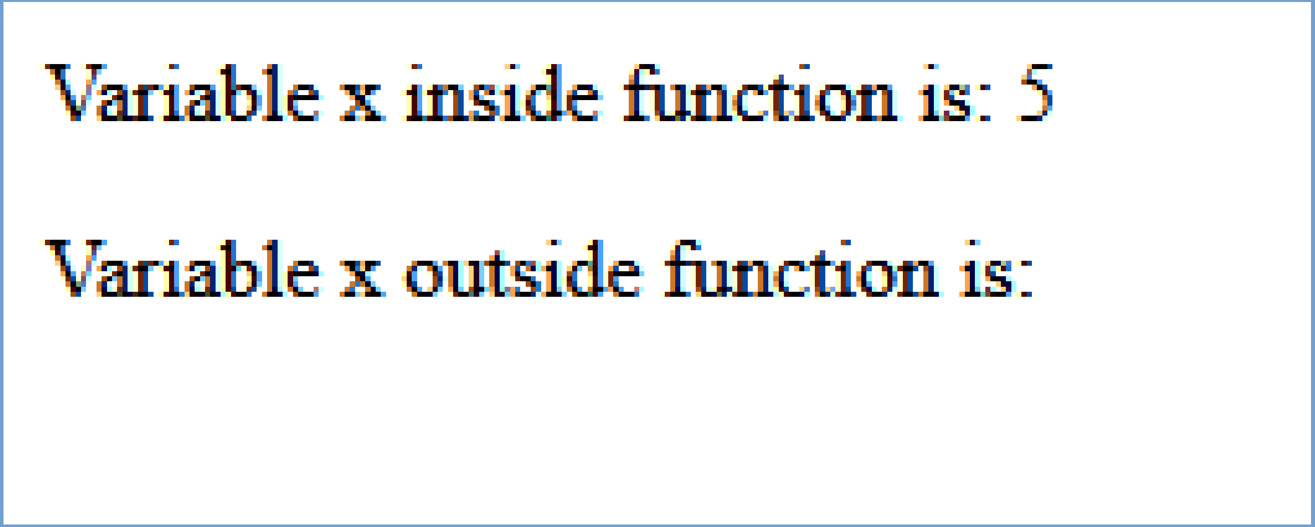
PHP global scope (global keyword)

```
 <?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

15

PHP local scope

```
<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```



```
Variable x inside function is: 5

Variable x outside function is:
```

PHP static scope

```
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

0

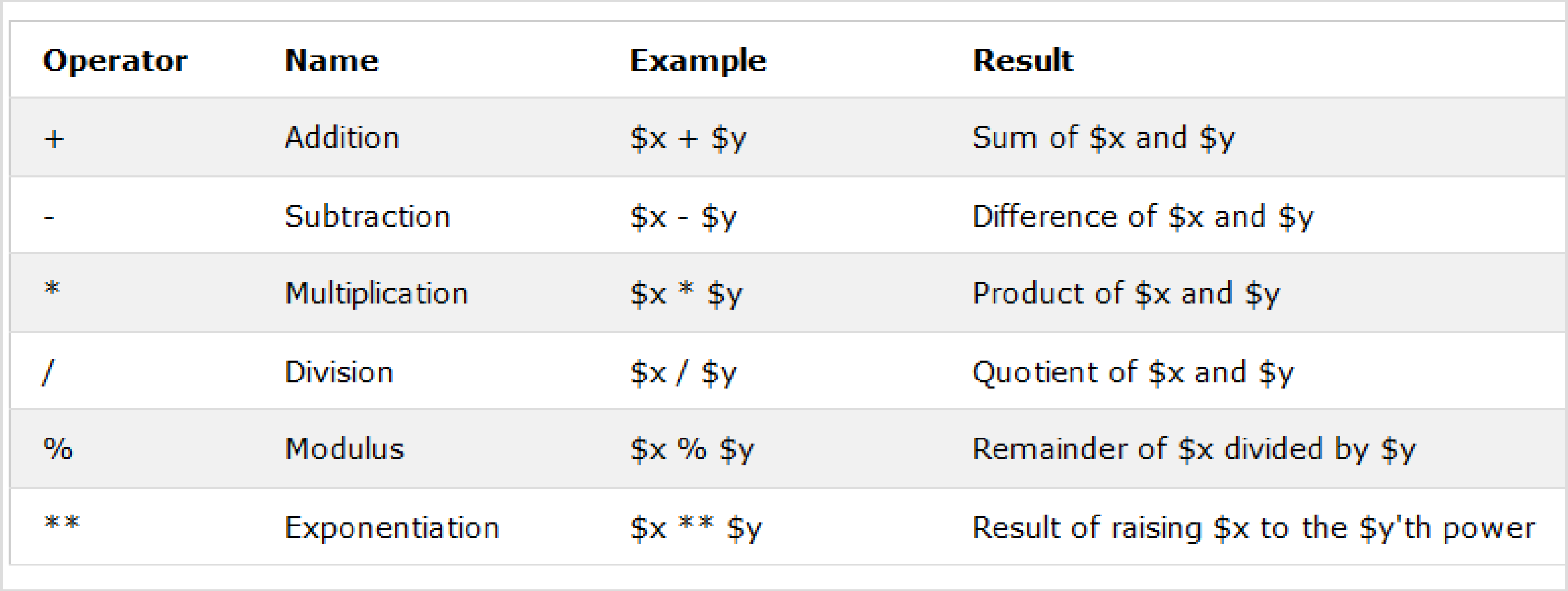
1

2

PHP Operators

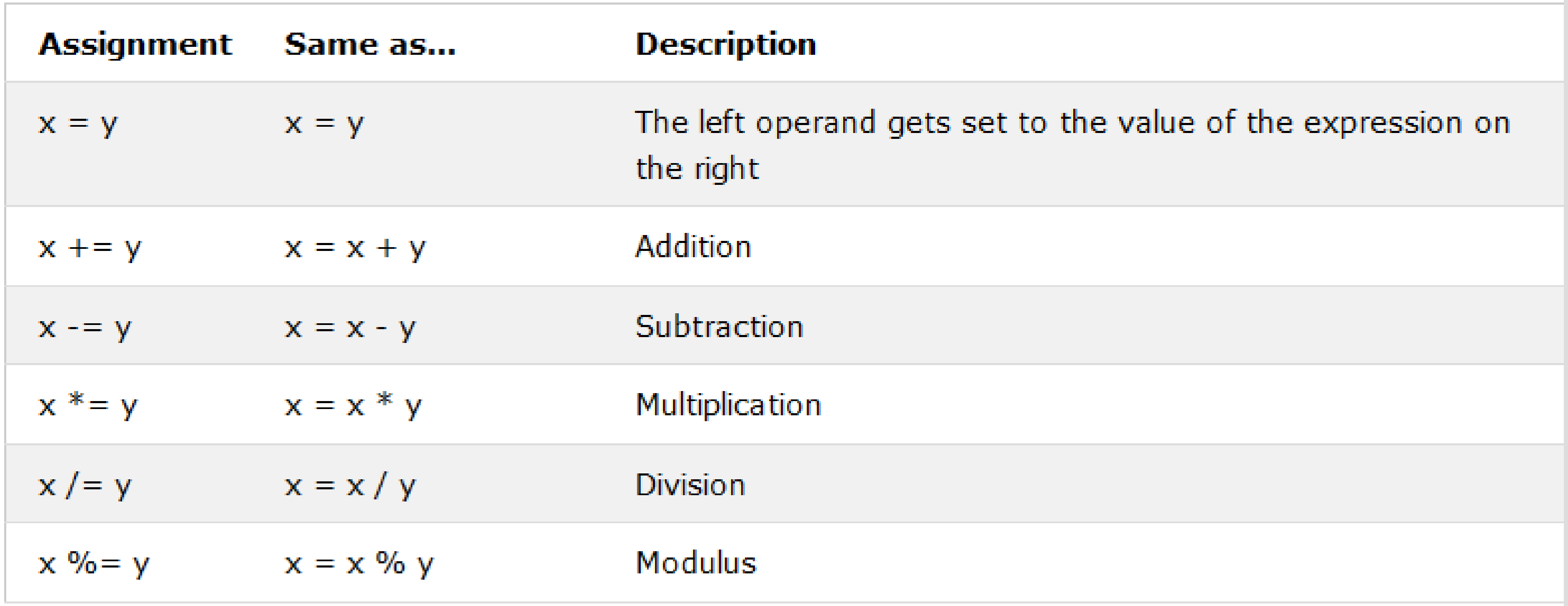
- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - *Arithmetic operators*
  - *Assignment operators*
  - *Comparison operators*
  - *Increment/Decrement operators*
  - *Logical operators*
  - *String operators*
  - *Array operators*
  - *Conditional assignment operators*

PHP Arithmetic Operators



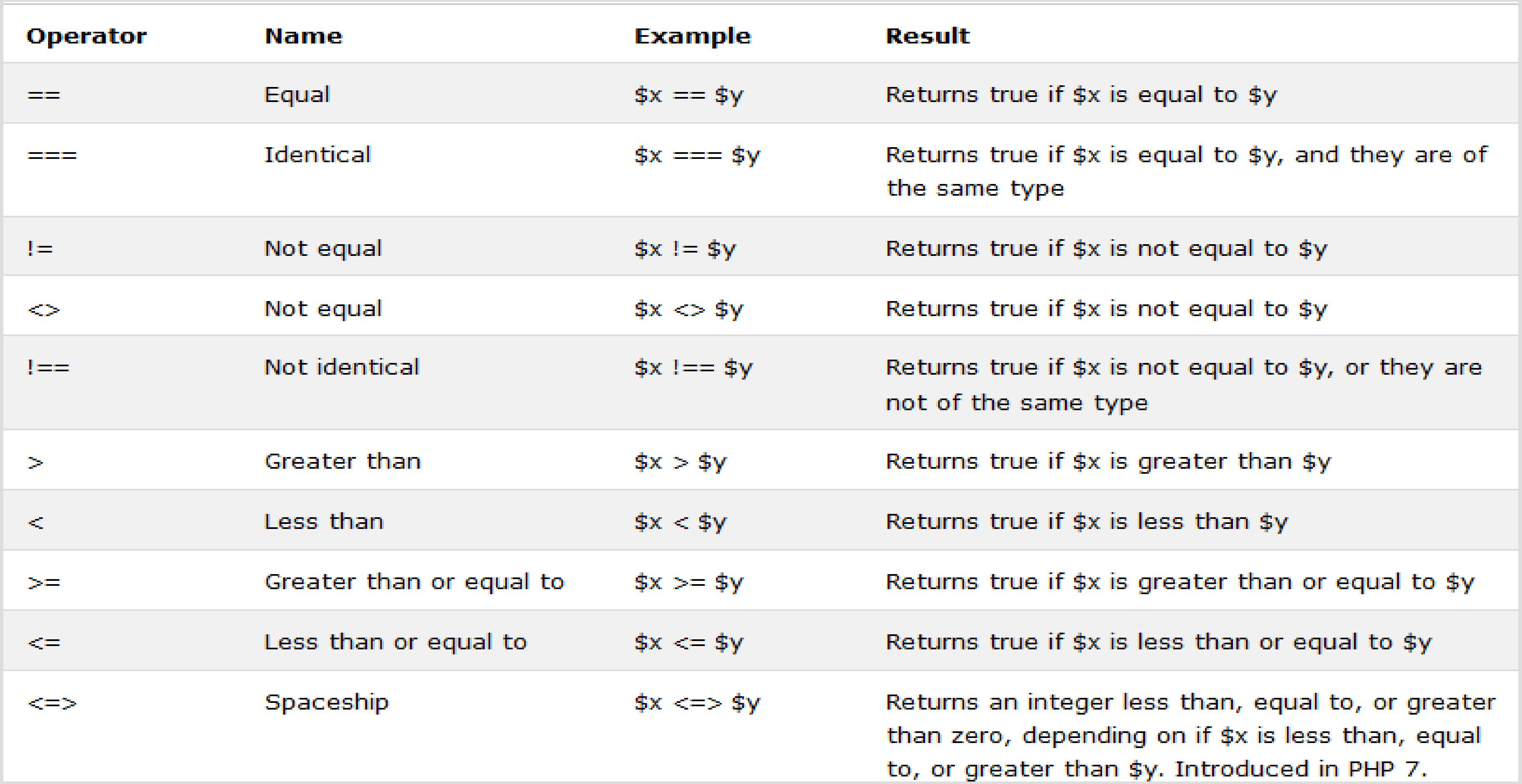
Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators



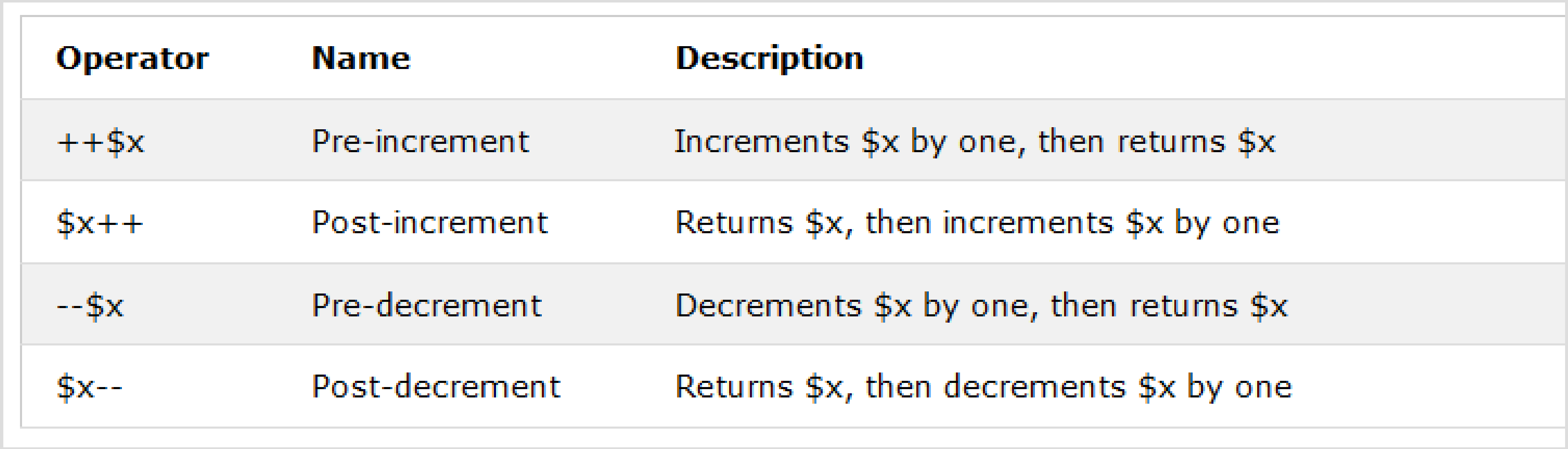
Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators



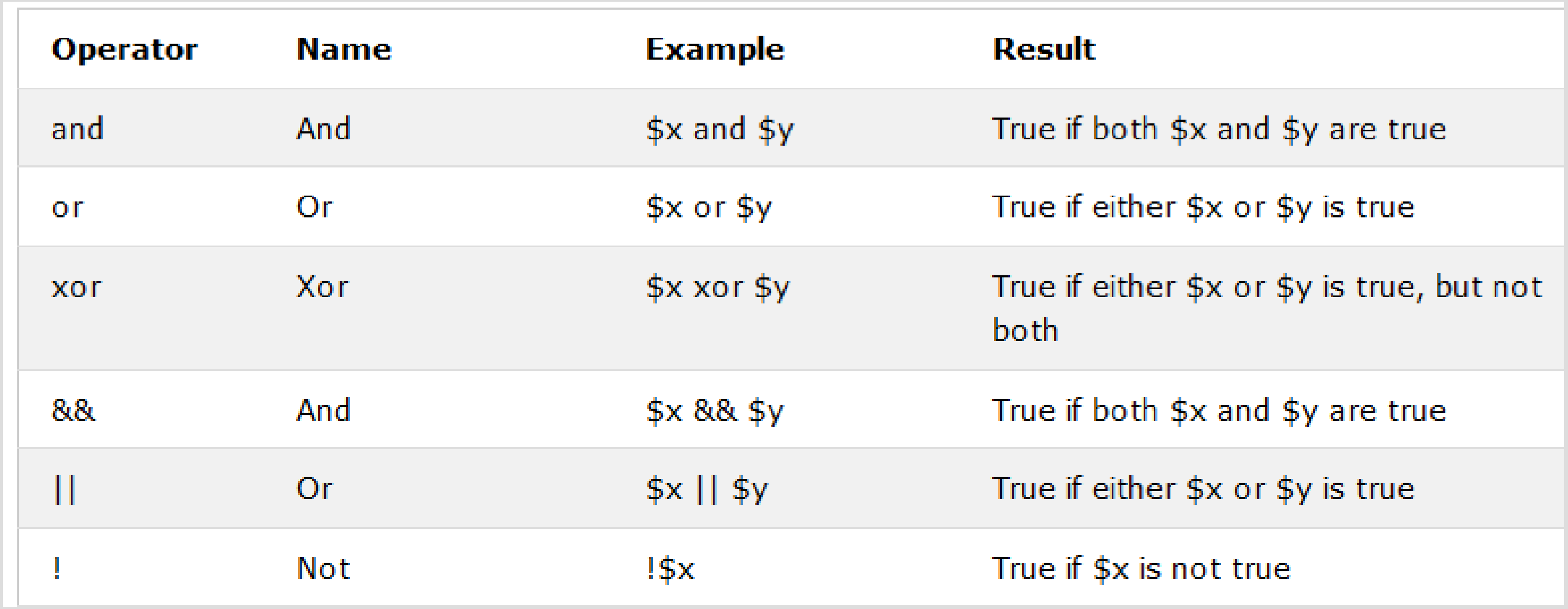
Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators



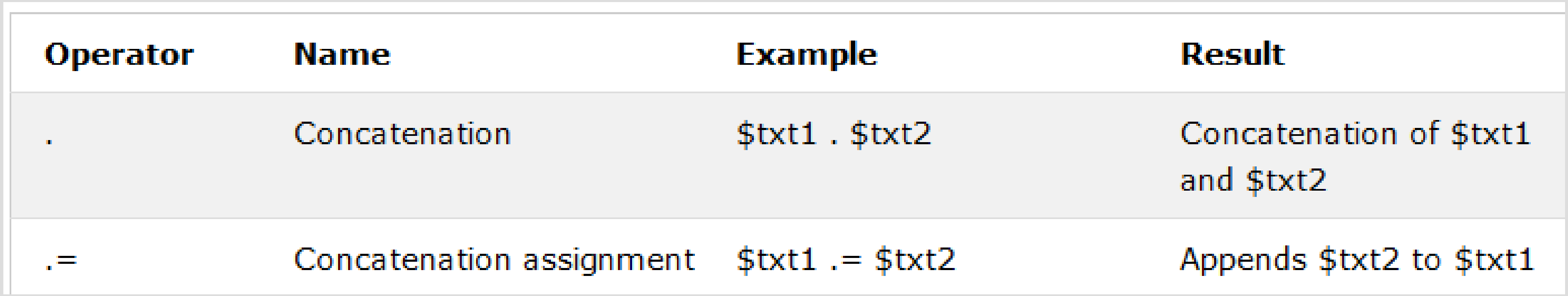
Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators



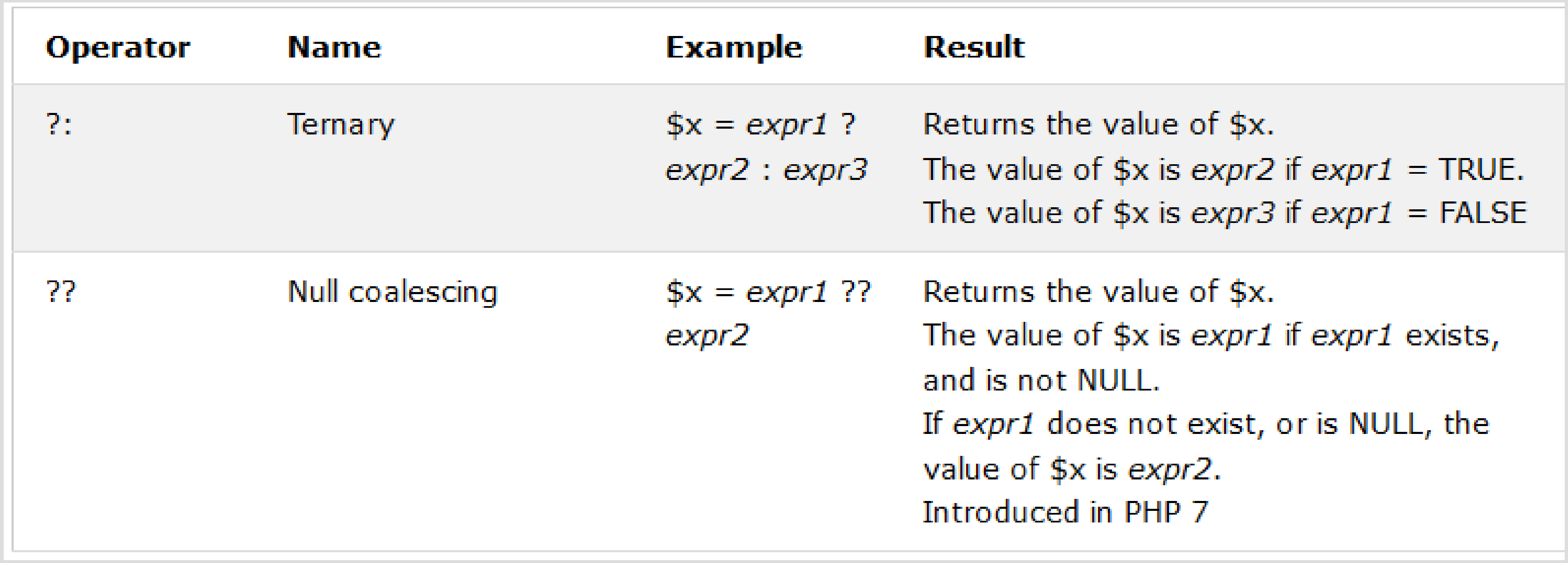
Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators



Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Conditional Assignment Operators



Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

PHP Conditional Statements

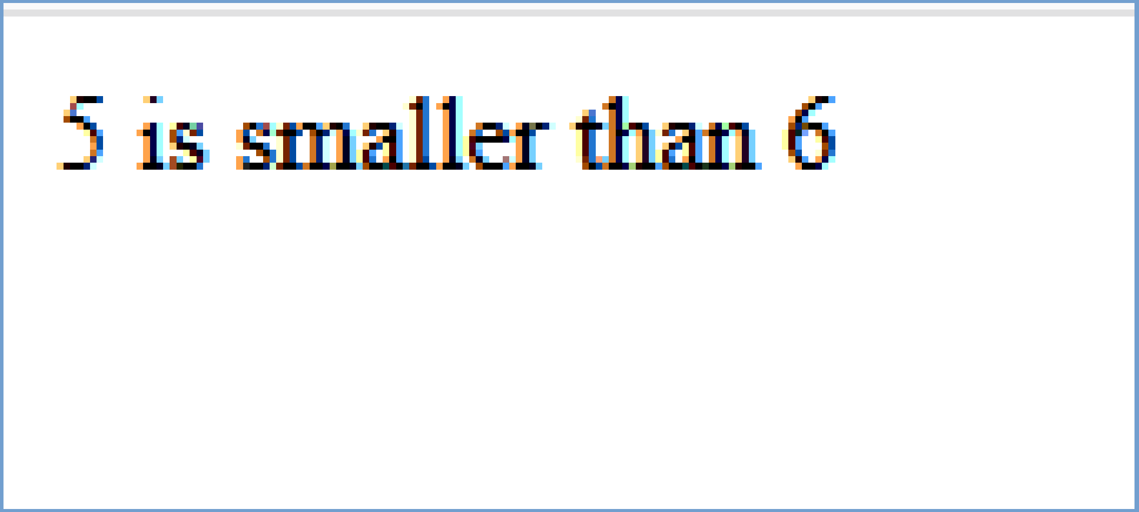
- **if** statement – executes some code if a condition is **true**
- **if...else** statement – executes some code if a condition is true and another code if that condition is false
- **if...elseif...else** statement – executes different codes for more than two conditions.
- **switch** statement – selects one of many blocks of code to be executed.

PHP if Statement

```
if (condition) {
  code to be executed if condition is true;
}
```

For example,

```
<?php
$t = 5;

if ($t < 6) {
  echo "$t is smaller than 6";
}
?>
```



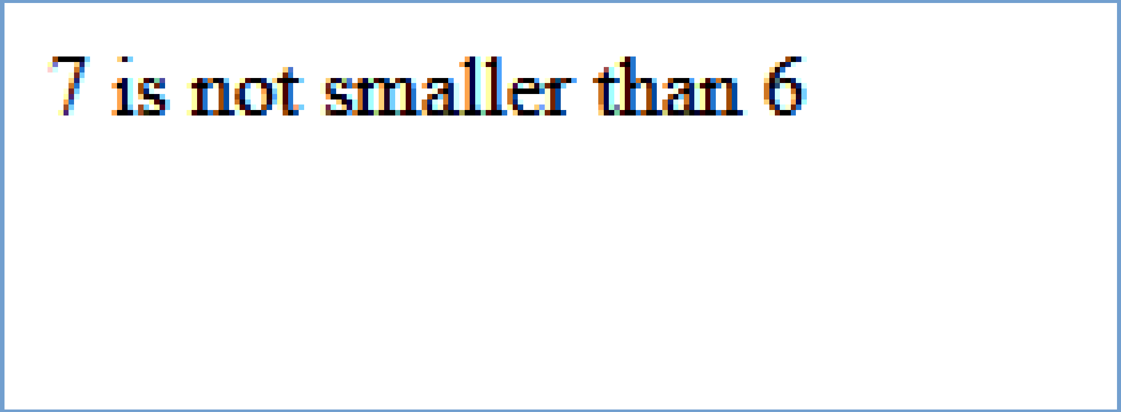
```
5 is smaller than 6
```

PHP if...else Statement

```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

For example,

```
<?php
$t = 7;

if ($t < 6) {echo "$t is smaller than 6";}
else
{echo "$t is not smaller than 6";}
?>
```



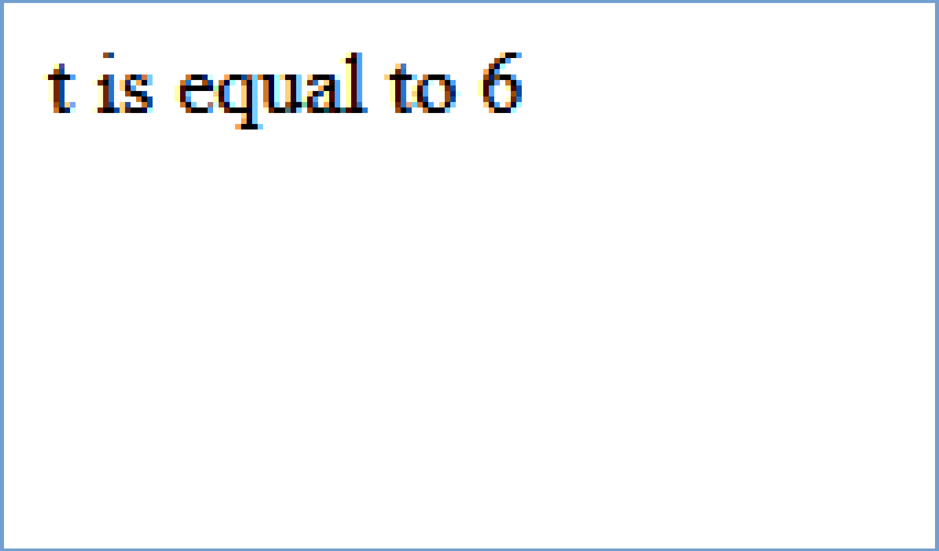
```
7 is not smaller than 6
```

PHP if...elseif...else Statement

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

For example,

```
<?php
$t = 6;

if ($t < 6) {echo "$t is smaller than 6";}
elseif ($t === 6) {echo "t is equal to 6";}
else {echo "$t is not smaller than 6";}
?>
```



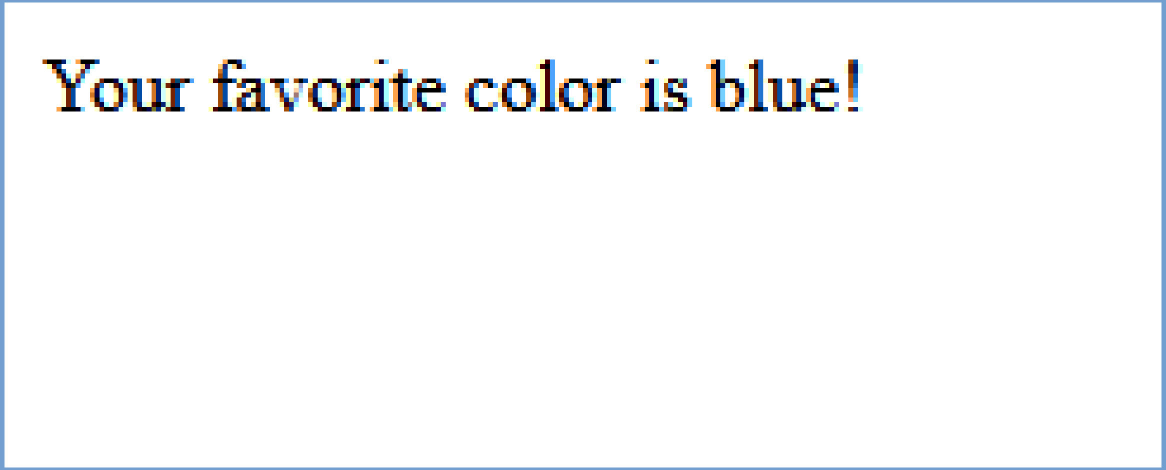
```
t is equal to 6
```

PHP switch Statements

```
switch (n) {
case label1:
code to be executed if n=label1;
break; /* if the break is not included, all cases till the first break/end is executed  */
case label2:
code to be executed if n=label2;
break;
case label3:
code to be executed if n=label3;
break;
...
default:
code to be executed if n is different from all labels;
}
```

PHP switch Statements (continued)

```
 <?php
$favcolor = "blue";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```



Your favorite color is blue!

PHP Loops

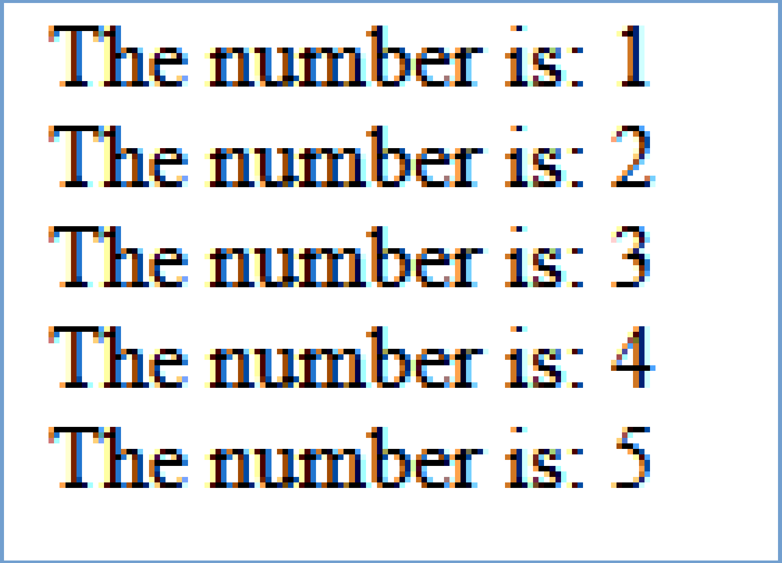
- Loops are used to execute the same block of code again and again, as long as a certain condition is true.
- In PHP, we have the following loop types:
  - ***while*** *– loops through a block of code as long as the specified condition is true*
  - ***do...while*** *– loops through a block of code once, and then repeats the loop as long as the specified condition is true*
  - ***for*** *– loops through a block of code a specified number of times*
  - ***foreach*** *– loops through a block of code for each element in an array*

PHP while loops

```
while (condition is true) {
  code to be executed;
}
```

For example,

```
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```



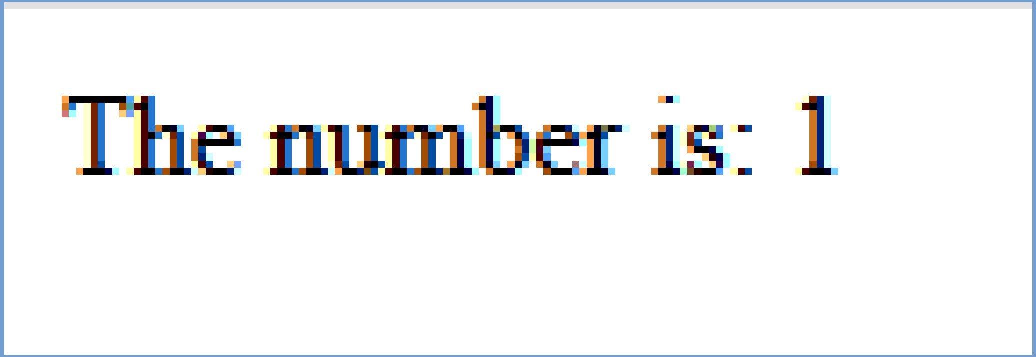
```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

PHP do...while loops

```
do {
  code to be executed;
} while (condition is true);
```

For example,

```
 <?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x < 1);
?>
```



```
The number is: 1
```

PHP for loops

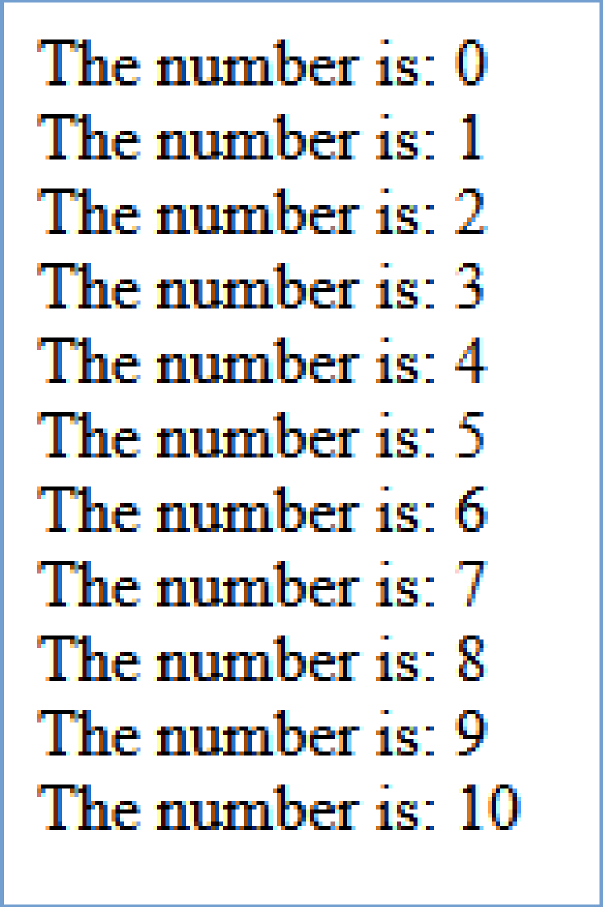
for (init counter; test counter; increment counter) {

code to be executed for each iteration;

}

For example,

```
 <?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

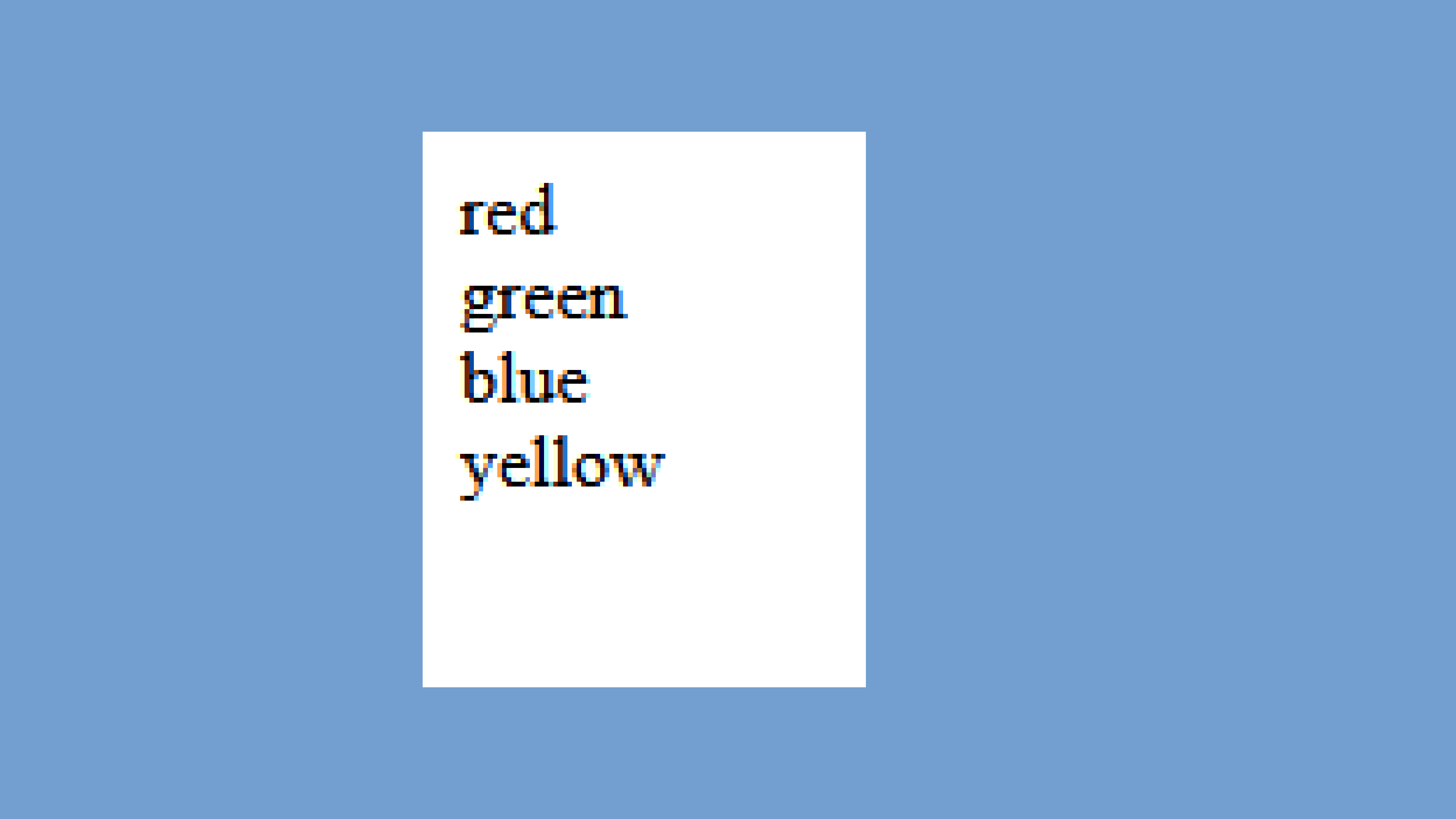

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

PHP foreach loops

foreach ($array as $value) {
code to be executed;
}

For example,

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```



red

green

blue

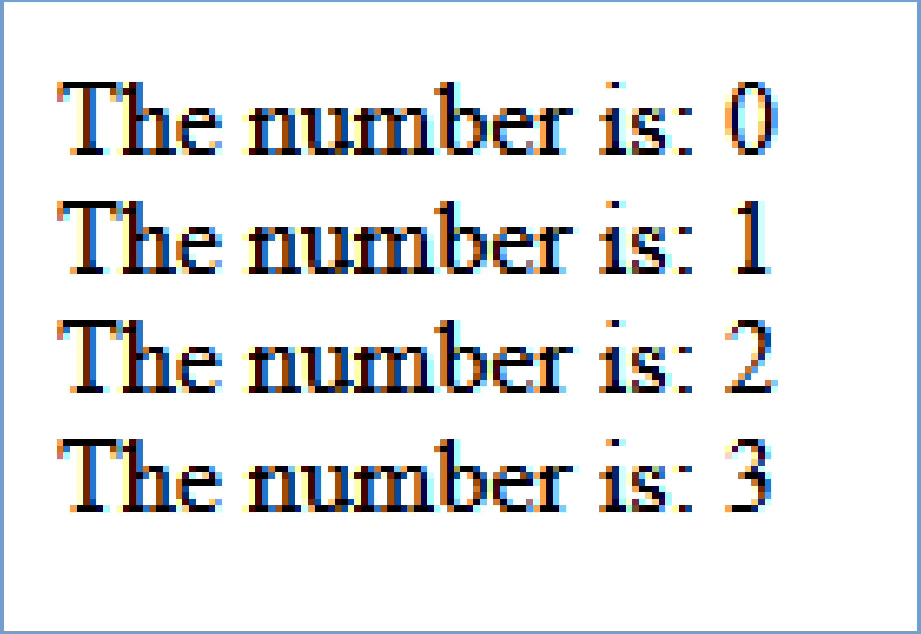
yellow

break

To jump out of the loop. Also, could be used for jumping out of the switch, as it described before.

For example,

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
?>
```

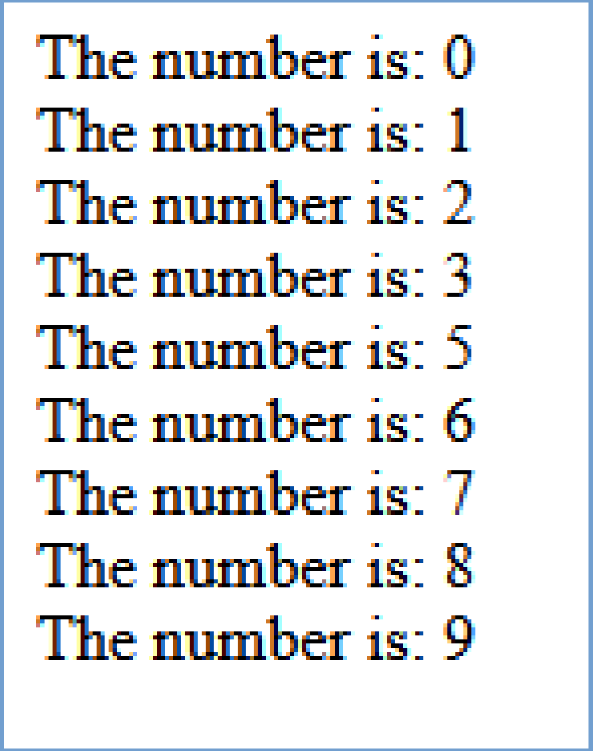

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
```

continus

To break one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

For example,

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
?>
```



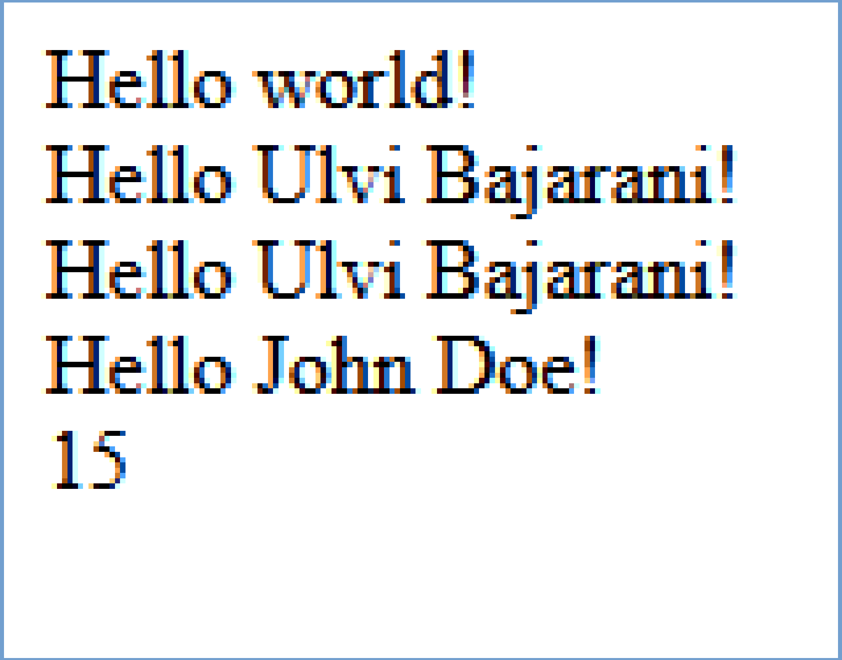
```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
```

PHP functions

```
function nameOfTheFunction (variables)
{
/* Definition */
}
nameOfTheFunction (variableValues);
```

PHP functions (continue)

```
 <?php
function writeMsg() {
  echo "Hello world!<br>";
}
writeMsg(); // call the function
function Test($name, $surname) {
  echo "Hello $name $surname!<br>";
}
Test("Ulvi", "Bajarani");
function TestDefaultValue($name="John", $surname="Doe") {
  echo "Hello $name $surname!<br>";
}
TestDefaultValue("Ulvi", "Bajarani");
TestDefaultValue();
function TestAddition($a, $b) {
return $a + $b;
}
TestAddition(5, 10);
?>
```



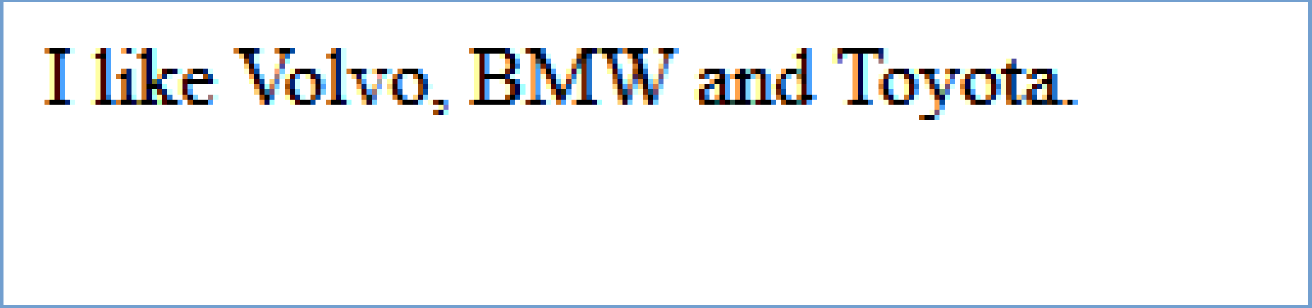
```
Hello world!
Hello Ulvi Bajarani!
Hello Ulvi Bajarani!
Hello John Doe!
15
```

PHP Array

Created by array(values);

For example,

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```



I like Volvo, BMW and Toyota.

PHP super global variables

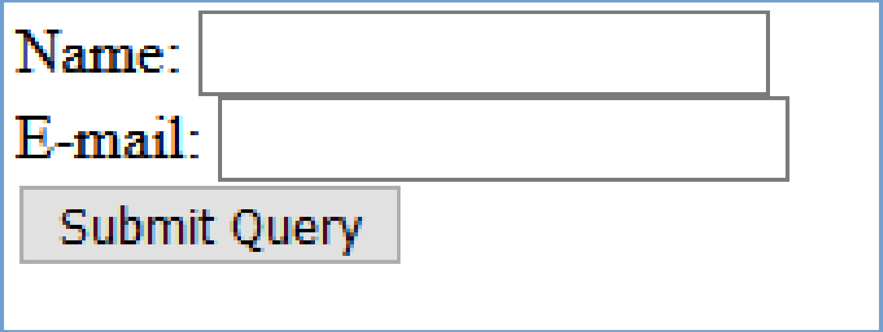
- The **super global variables** allow the usage of HTTP requests:
  - ***$_POST –*** *for post formdata*
  - ***$_GET*** *– for get formdata or query string.*
  - ***$_REQUEST –*** *for either get or post request data.*
  - ***$_SERVER*** *– for information related to the Web server, HTTP request headers, and the PHP script itself*
  - ***$_ENV*** *– for CGI defined variables and any form the shell used on particular operating systems*
  - ***$_COOKIES*** *– for stored cookies*
  - ***$_SESSION*** *– for session information*

PHP Form Processing

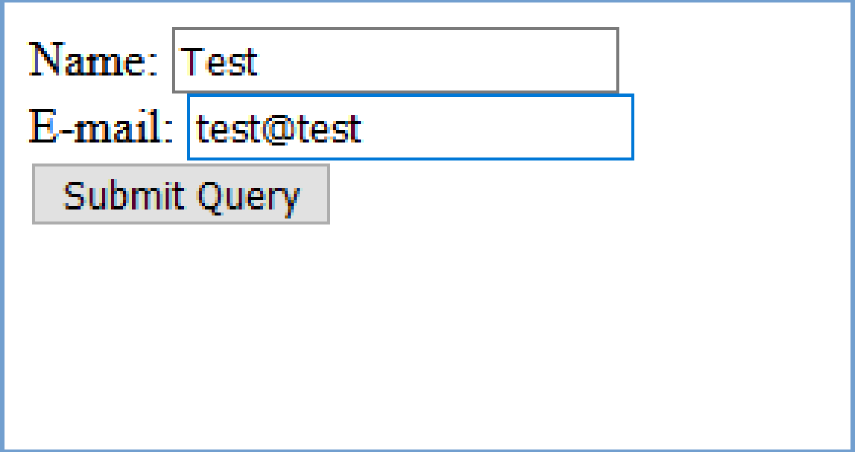
- Is done by $_GET["name"] and $_POST["name"] methods (same as HTML’s get and post). For details please go to the proper slide.

PHP Form Processing (continued)

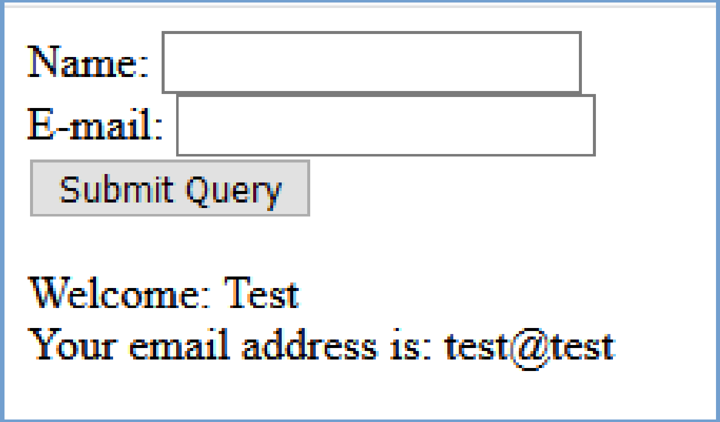
```
<html>
<body>
<form action="" method="post">
Name: <input type="text" name="name"><br>E-mail: <input type="text"
name="email"><br><input type="submit" name="SubmitButton">
</form>
<?php
if(isset($_POST['SubmitButton']))
{echo "Welcome: " . $_POST["name"] . "<br>" . "Your email address is: " .
$_POST["email"] ;}
?>
</body>
</html>
```


Name:

E-mail:


Name: Test

E-mail: test@test


Name:

E-mail:

Welcome: Test
Your email address is: test@test

PHP Form Validation

- $_SERVER["PHP_SELF"] is a super global variable returning the file name of the executing script. It submits the data in the same page instead of jumping to another page.
- Can be used by hackers.
- To avoid the interception by hackers:
- Use **htmlspecialchars()** function for $_SERVER["PHP_SELF"]. The function converts all characters to HTML characters, which allows to avoid JavaScript code execution (Cross-site Scripting attacks)
- Remove extra spaces/tabs by **trim()** function
- Remove \ - Backslashes by stripslashes() function

PHP Form Validation (continued)

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}
```

PHP Form Validation (continued) the continue of the code

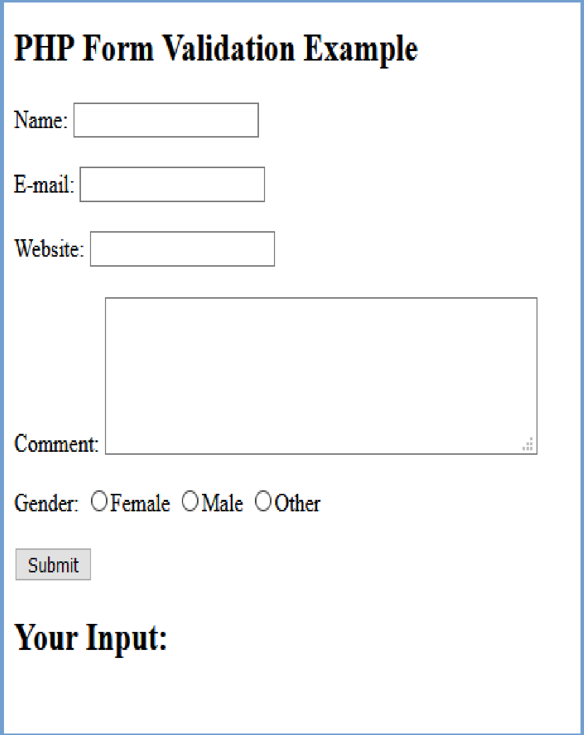
```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
```

PHP Form Validation (continued) the continue of the code

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
```

PHP Form Validation (continued) the continue of the code

```
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```



PHP Form Validation Example

Name:

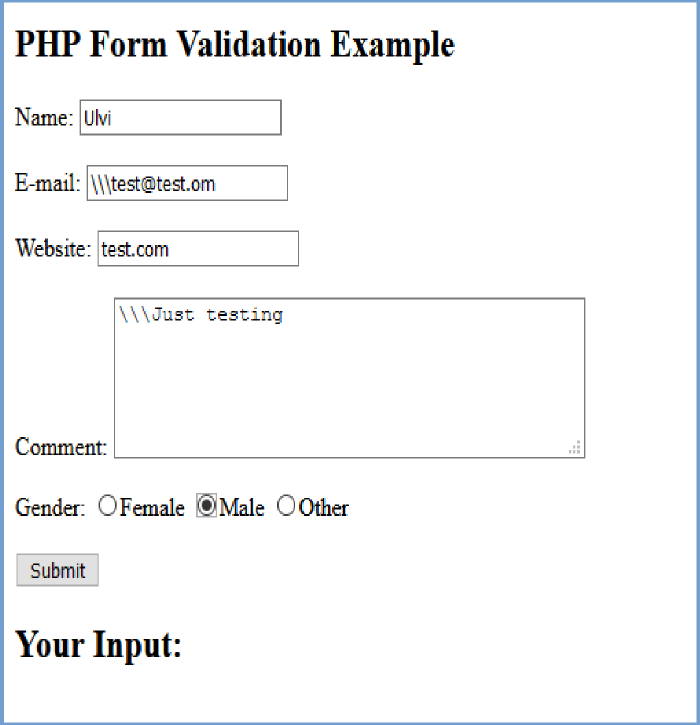
E-mail:

Website:

Comment:

Gender: ○ Female ○ Male ○ Other

Your Input:



PHP Form Validation Example

Name: Ulvi

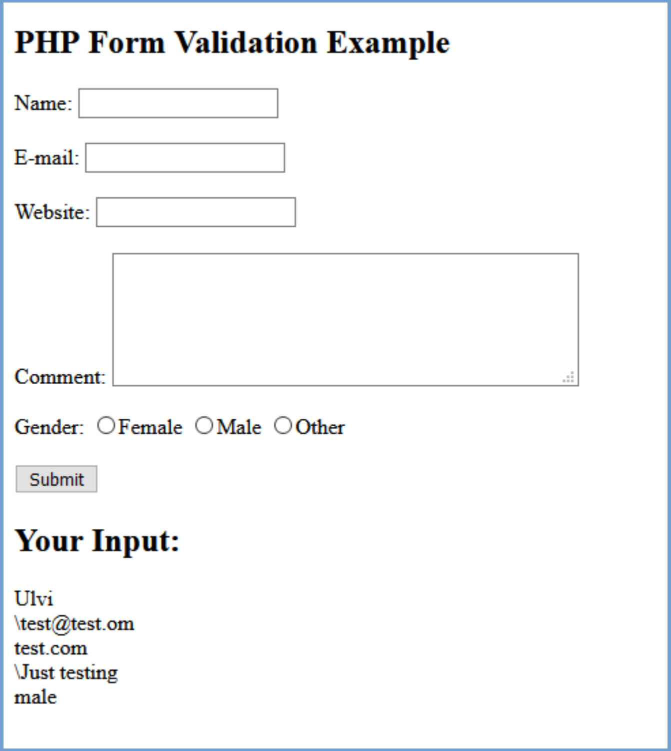
E-mail: \\\test@test.om

Website: test.com

Comment: \\\Just testing

Gender: ○ Female [✓ Male] ○ Other

Your Input:



PHP Form Validation Example

Name:

E-mail:

Website:

Comment:

Gender: ○ Female ○ Male ○ Other

Your Input:

```
Ulvi
\test@test.om
test.com
\Just testing
male
```

PHP Form Validation

- To check if the variable is empty, we use **empty()** function.

PHP Form Validation (continued)

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
```

PHP Form Validation (continued) the continue of the code

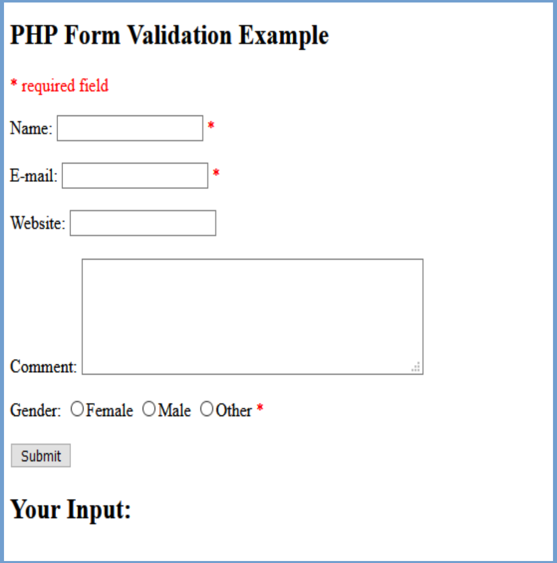
```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {$nameErr = "Name is required";} else {$name = test_input($_POST["name"]);}

  if (empty($_POST["email"])) {$emailErr = "Email is required";} else {$email = test_input($_POST["email"]);}

  if (empty($_POST["website"])) {$website = "";} else {$website = test_input($_POST["website"]);}

  if (empty($_POST["comment"])) {$comment = "";} else {$comment = test_input($_POST["comment"]);}

  if (empty($_POST["gender"])) {$genderErr = "Gender is required";} else {$gender =
test_input($_POST["gender"]);}
}
```

PHP Form Validation (continued) the continue of the code

```
<h2>PHP Form Validation Example</h2><p><span class="error">* required
field</span></p><form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">Name: <input type="text"
name="name"><span class="error">* <?php echo
$nameErr;?></span><br><br>E-mail: <input type="text" name="email"><span
class="error">* <?php echo $emailErr;?></span><br><br>Website: <input
type="text" name="website"><span class="error"><?php echo
$websiteErr;?></span><br><br>Comment: <textarea name="comment" rows="5"
cols="40"></textarea><br><br>Gender:<input type="radio" name="gender"
value="female">Female<input type="radio" name="gender"
value="male">Male<input type="radio" name="gender" value="other">Other<span
class="error">* <?php echo $genderErr;?></span><br><br><input type="submit"
name="submit" value="Submit"></form>
```

PHP Form Validation (continued) the continue of the code

```
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```



PHP Form Validation Example

* required field

Name: *

E-mail: *

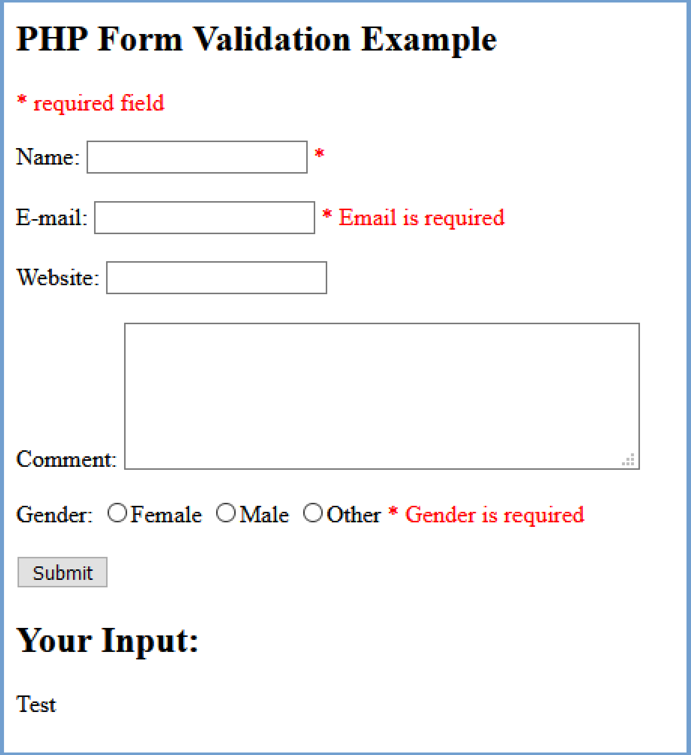
Website:

Comment:

Gender: ○Female ○Male ○Other *

Submit

Your Input:



PHP Form Validation Example

* required field

Name: *

E-mail: * Email is required

Website:

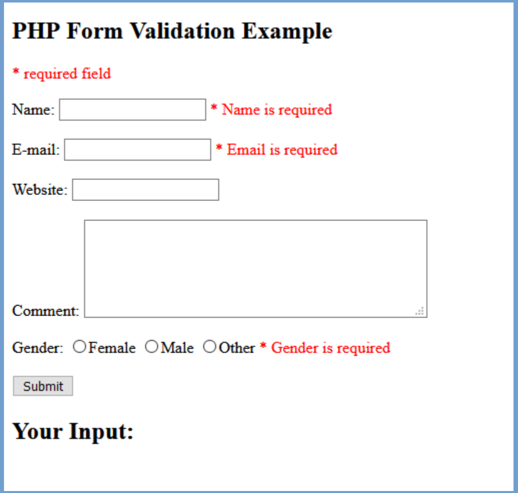
Comment:

Gender: ○Female ○Male ○Other * Gender is required

Submit

Your Input:

Test



PHP Form Validation Example

* required field

Name: * Name is required

E-mail: * Email is required

Website:

Comment:

Gender: ○Female ○Male ○Other * Gender is required

Submit

Your Input:

PHP Form Validation (continued)

- Name validation:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

- The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP Form Validation (continued)

- Email validation:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
$emailErr = "Invalid email format";
}
```

PHP Form Validation (continued)

- $website = test_input($_POST["website"]);

  if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {

  $websiteErr = "Invalid URL";

  }

Using MySQL with PHP

- For the database operations, either MySQLi or PDO might be used (the latter one is not going to be covered). MySQLi allows both procedural and OOP approach.

Using MySQL with PHP (continued)

- To connect the database, these values are used (OOP):

```
<?php
$servername = "localhost";$username = "username";$password = "password";$dbname = "dbname";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
?>
```

Using MySQL with PHP (continued)

- To connect the database, these codes are used (procedural):

```
<?php
$servername = "localhost";$username = "username";$password = "password";
$dbname = "dbname";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {die("Connection failed: " . mysqli_connect_error());}
?>
```

Using MySQL with PHP (continued)

- To close the connection, these codes are used:
  - *For OOP:*

    $conn→close();
  - *For Procedural:*

    mysqli_close($conn);

Using MySQL with PHP (continued)

- To execute the specific SQL statements, these codes are used:
  - *For OOP:*

    $conn->query($sql)
  - *For Procedural:*

    mysqli_query($conn, $sql);
- Remember that if you create a database, you shouldn't specify it while connecting (in other words, only the server name, the user name, and the password should be specified)

Using MySQL with PHP (continued)

- A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency. It is done by $conn→prepare($sql); usually executed by **$stmt = $conn→prepare($sql);**
- Prepared statements basically work like this:
  - *Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)*
  - *The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it*
  - *Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values*
- Compared to executing SQL statements directly, prepared statements have three main advantages:
- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query. It is done by **$stmt->bind_param("arguments", variables);** arguments are **i** – integer, **d** – double, **s** – string, **b** – BLOB and written in quotes. Be sure that arguments are equal to the number of variables.
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

Using MySQL with PHP (continued)

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
```

Using MySQL with PHP (continued)

(the continue of the code)

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
?>
```

Using MySQL with PHP (continued)

(the continue of the code)

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }
```

Using MySQL with PHP (continued)

(the continue of the code)

```
if (empty($_POST["website"])) {$website = "";} else {$website =
test_input($_POST["website"]);}


if (empty($_POST["comment"])) {$comment = "";} else {$comment =
test_input($_POST["comment"]);}


if (empty($_POST["gender"])) {$genderErr = "Gender is required";} else
{$gender = test_input($_POST["gender"]);}
}
```

Using MySQL with PHP (continued)

(the continue of the code)

```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Using MySQL with PHP (continued)

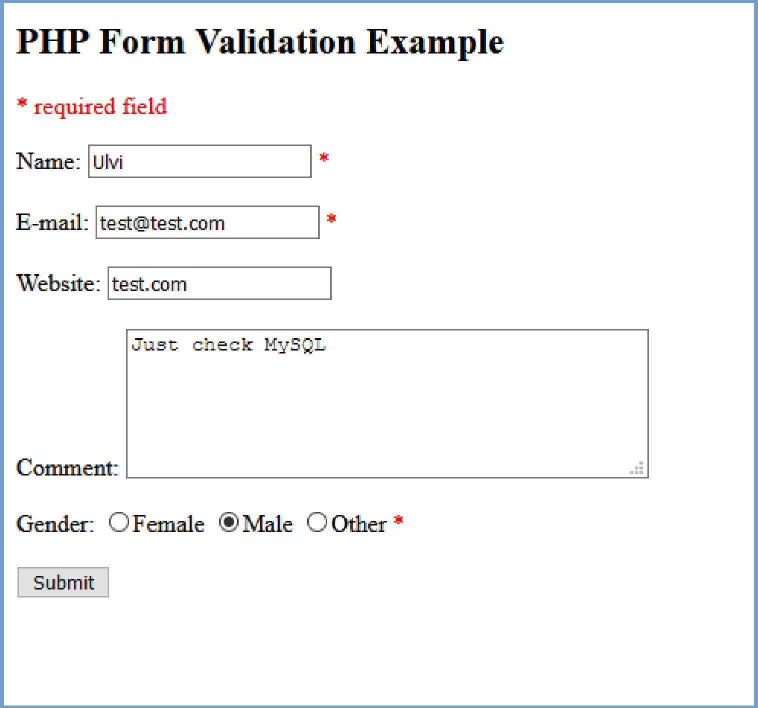
(the continue of the code)

```
<h2>PHP Form Validation Example</h2><p><span class="error">* required field</span></
p><form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?
>">Name: <input type="text" name="name"><span class="error">* <?php echo $nameErr;?
></span><br><br>E-mail: <input type="text" name="email"><span class="error">* <?php
echo $emailErr;?></span><br><br>Website: <input type="text" name="website"><span
class="error"><?php echo $websiteErr;?></span><br><br>Comment: <textarea
name="comment" rows="5" cols="40"></textarea><br><br>Gender:<input type="radio"
name="gender" value="female">Female<input type="radio" name="gender"
value="male">Male<input type="radio" name="gender" value="other">Other<span
class="error">* <?php echo $genderErr;?></span><br><br><input type="submit"
name="submit" value="Submit"></form>
```

Using MySQL with PHP (continued)

(the continue of the code)

```
<?php
if(isset($_POST["submit"]))
{
$sql = "INSERT INTO myguests (name, email, gender, comment, website) VALUES (?, ?, ?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("sssss", $name, $email, $gender, $comment, $website);
$stmt->execute();
$stmt->close();
echo "Data entered successfully!";
}
?>

</body>
</html>
```



PHP Form Validation Example

* required field

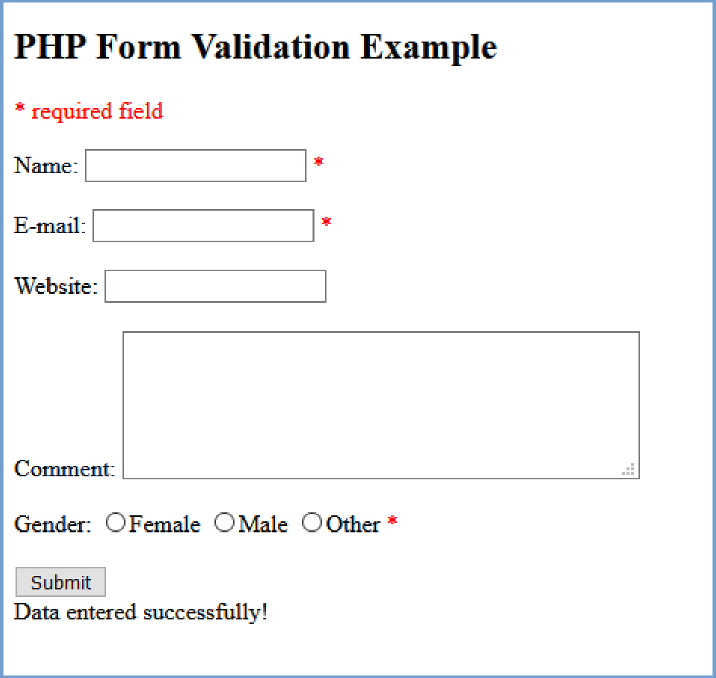
Name: Ulvi *

E-mail: test@test.com *

Website: test.com

Comment: Just check MySQL

Gender: ○Female [✓ Male] ○Other *



PHP Form Validation Example

* required field

Name: *

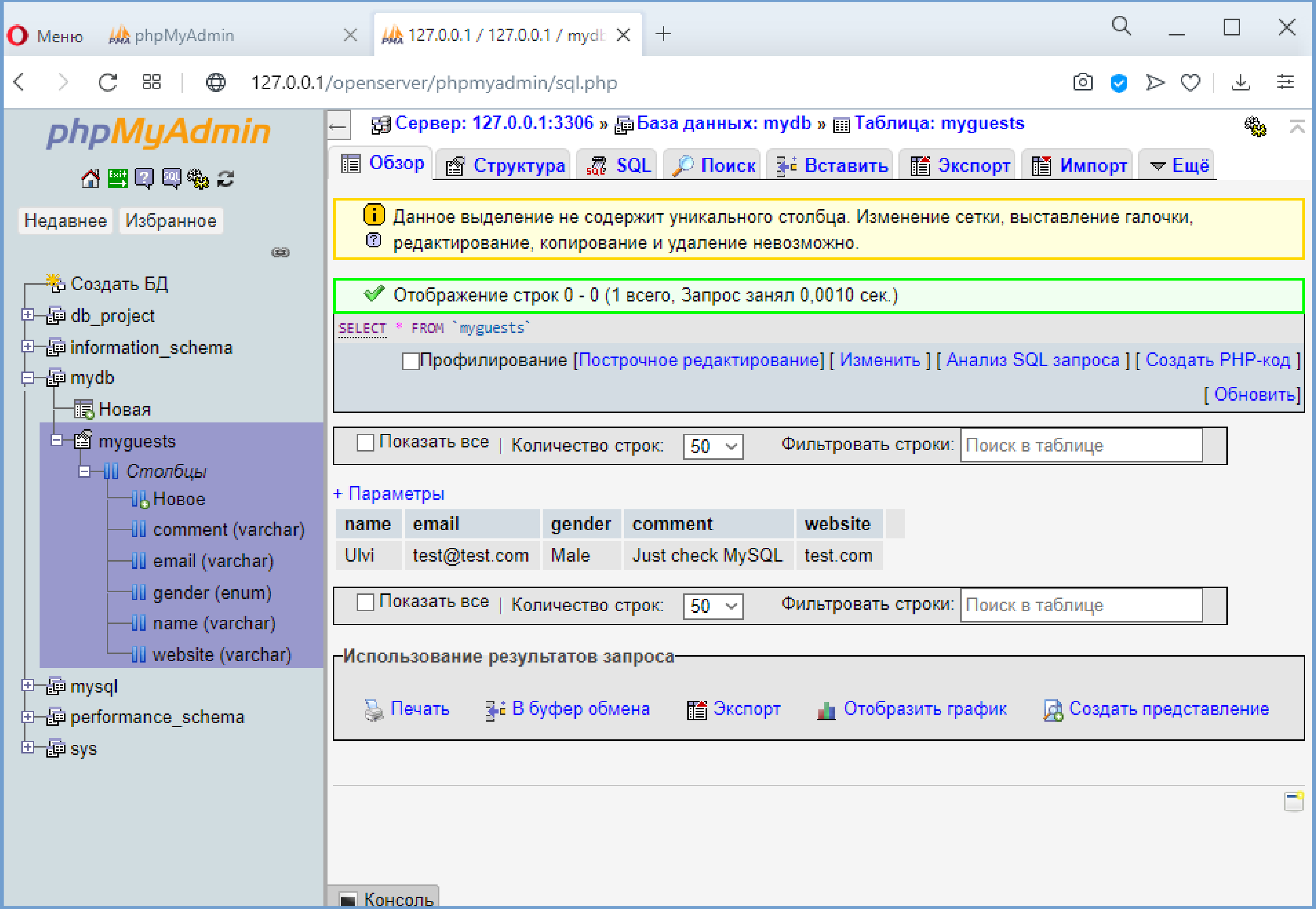
E-mail: *

Website:

Comment:

Gender: ○Female ○Male ○Other *

Data entered successfully!



Меню | phpMyAdmin | 127.0.0.1 / 127.0.0.1 / mydb

127.0.0.1/openserver/phpmyadmin/sql.php

phpMyAdmin

Недавнее Избранное

- Создать БД
- db_project
- information_schema
- mydb
  - Новая
  - myguests
    - *Столбцы*
      - Новое
      - comment (varchar)
      - email (varchar)
      - gender (enum)
      - name (varchar)
      - website (varchar)
- mysql
- performance_schema
- sys

Сервер: 127.0.0.1:3306 » База данных: mydb » Таблица: myguests

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Ещё

Данное выделение не содержит уникального столбца. Изменение сетки, выставление галочки, редактирование, копирование и удаление невозможно.

✓ Отображение строк 0 - 0 (1 всего, Запрос занял 0,0010 сек.)

```
SELECT * FROM `myguests`
```

Профилирование [Построчное редактирование] [Изменить] [Анализ SQL запроса] [Создать PHP-код] [Обновить]

Показать все | Количество строк: 50 Фильтровать строки: Поиск в таблице

+ Параметры

name	email	gender	comment	website
Ulvi	test@test.com	Male	Just check MySQL	test.com

Показать все | Количество строк: 50 Фильтровать строки: Поиск в таблице

Использование результатов запроса

Печать | В буфер обмена | Экспорт | Отобразить график | Создать представление

Консоль

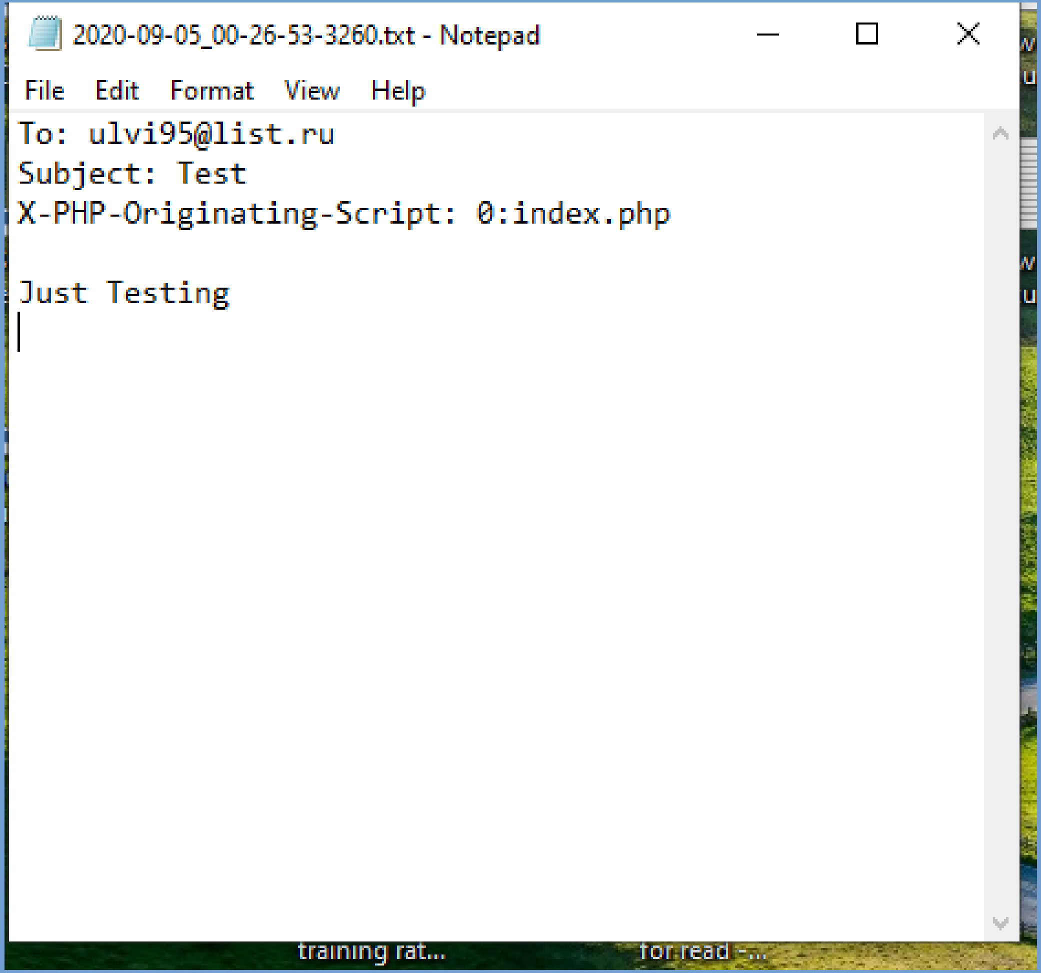
PHP Mails

- Is sent by **mail()** function;
- Has limited functionality;
- PHP mail function does not support SMTP authentication and doesn’t allow sending messages via external servers.
- The syntax is **(headers and parameters are optional)**:

```
<?php
mail($to_email_address,$subject,$message,[$headers],[$parameters]);
?>
```

PHP Mails (continued)

```
if(isset($_POST["submit"]))
{
mail($email, "Test", $comment);
echo "mail sent successfully!";
}
?>
```


2020-09-05_00-26-53-3260.txt - Notepad
File Edit Format View Help

```
To: ulvi95@list.ru
Subject: Test
X-PHP-Originating-Script: 0:index.php

Just Testing
```

training rat...
for read -...