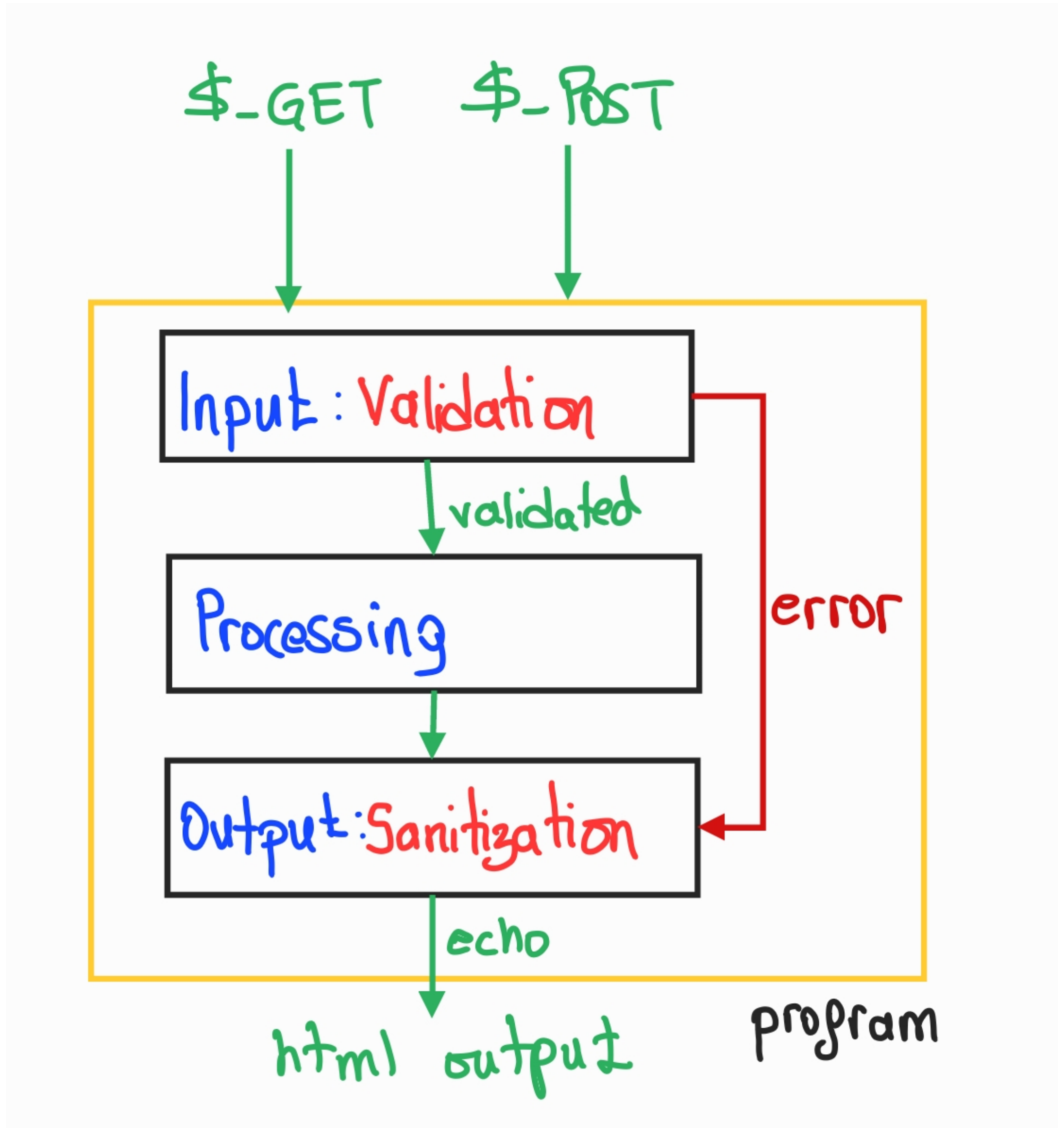
CTIS 256 Web Technologies II

Notes # 5

Serkan GENÇ

Secure Application

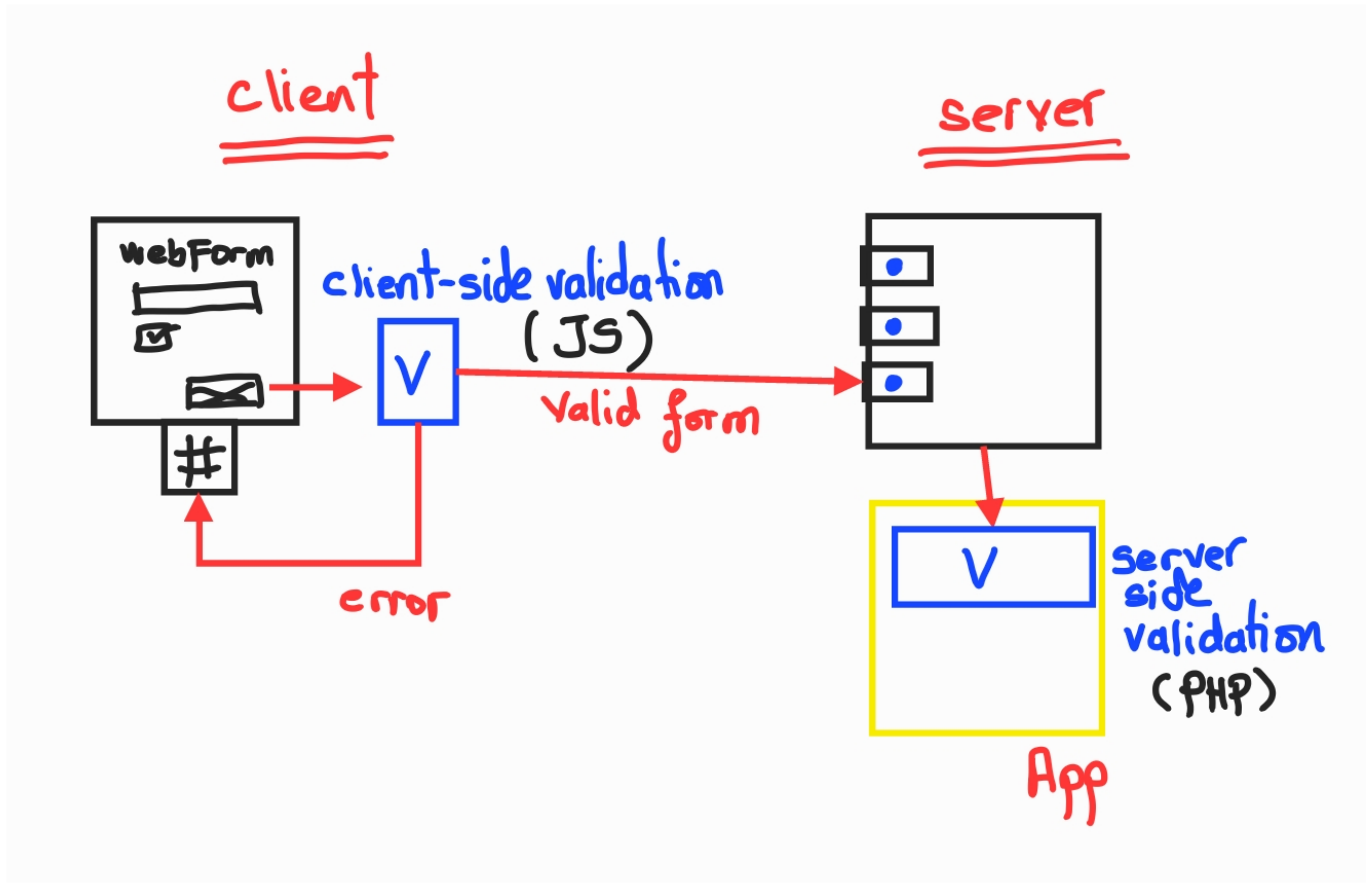


Input Validation

- **Input validation** is performed to make sure that only the properly formed data is accepted for further processing.
- Data from all potentially untrusted sources ($_GET, $_POST) should be subject to input validation.
- The aim is to **prevent security attacks** *(XSS, SQL Injection)* and **exploits** from hackers.
- **Syntactic Validation** should enforce syntax of the data (e.g. date, email, phone, security id, integer, float)
- **Semantic Validation (Verification)** should enforce the correctness of their values. (age withing expected range, start date is before end date, email address belongs to the user, given date is valid, 29 Feb 2021 invalid for instance)

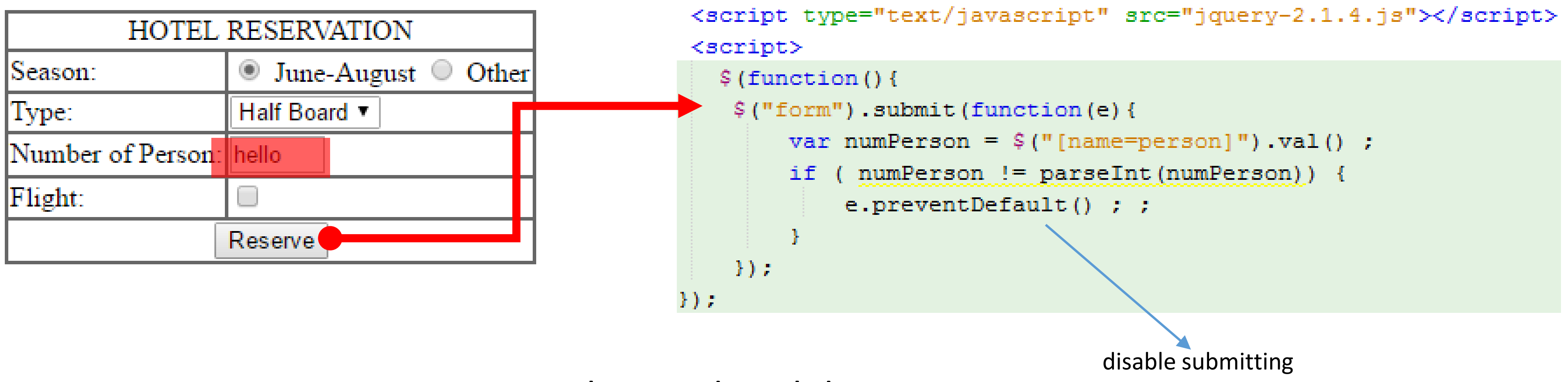
Field	Validation Rules	Invalid Samples
Name	required letters, dot and spaces	Ahmet123, $Ali, <b>Ali</b>,
Age	integer between 1 and 110	0, -10, 20.5, 125, five
Cgpa	float value between 0.0 and 4.0	-0.1, 4.2, four
Email	valid Email	ali.test.tr, ali@test, #1ali@.com.tr
Days	valid day names	ali, 1234, <script>, sunnndayy
Checkbox	item must be checked	
Password	at least 6 chars, at least one digit,letter	ali, ali12, 123456,

Validation Points



Client-side Form Validation (optional)

- For security and proper execution of the application, the data acquired with any methods such as form data or url data (query string) must be validated.
- If you obtain data from a form field, you have to check its presence, type, format, range, white lists and so on, then, you should use them in your processing.
- Mainly, there are **two places** to validate data; client-side validation (*before sending/submitting data*) or server-side validation (*after sending data*).
- **Client-side validation** is done by using **Javascript** within browser.
- Server-side validation is performed by server-side program.
- Client-side validation is **optional but highly recommended** but Server-side validation is **mandatory**.
- **Javascript** can be turned on and off from browser settings. This is why client-side validation is **not reliable**.
- Client-side validation increases **performance** since it prevents network round trip.

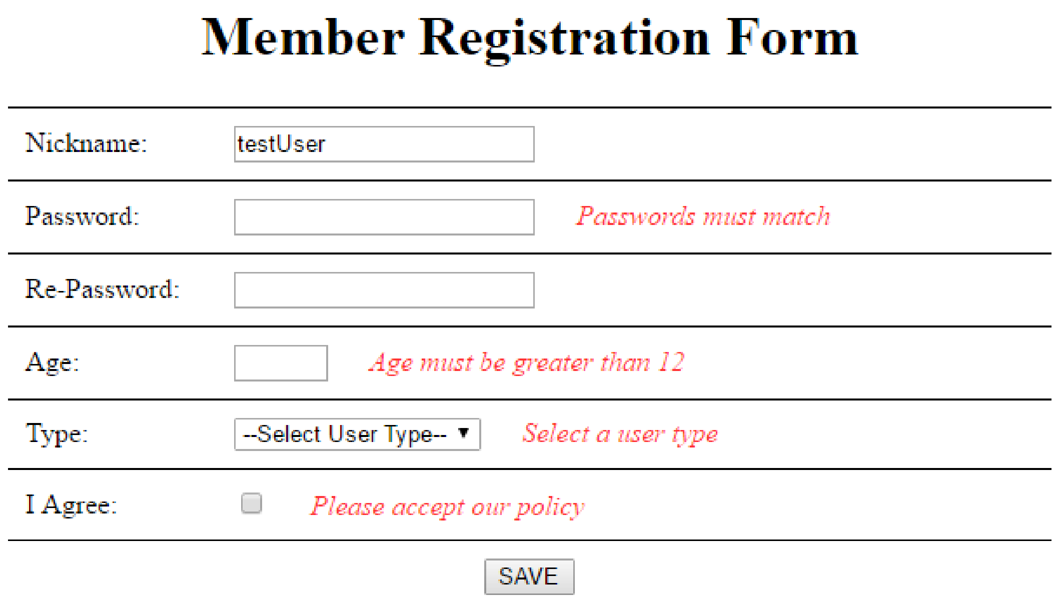


HOTEL RESERVATION	
Season:	[✓ June-August] Other
Type:	Half Board ▾
Number of Person:	hello
Flight:	☐
	Reserve

```
<script type="text/javascript" src="jquery-2.1.4.js"></script>
<script>
  $(function(){
   $("form").submit(function(e){
       var numPerson = $("[name=person]").val() ;
       if ( numPerson != parseInt(numPerson)) {
           e.preventDefault() ; ;
       }
   });
});
```

client-side validation

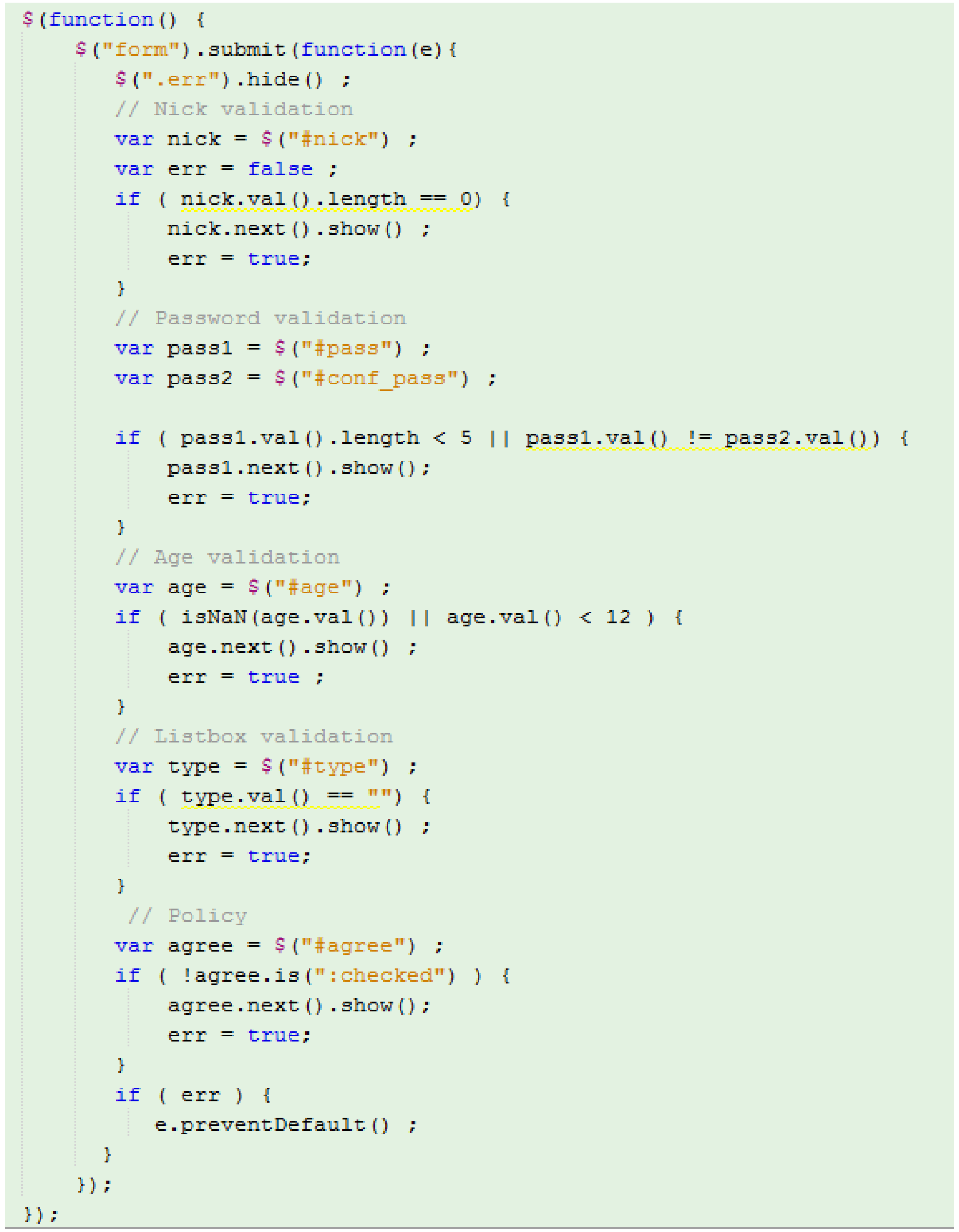
Validation With JQuery



Member Registration Form

Nickname:	testUser	
Password:		*Passwords must match*
Re-Password:		
Age:		*Age must be greater than 12*
Type:	--Select User Type-- ▾	*Select a user type*
I Agree:	☐	*Please accept our policy*

SAVE



```
$(function() {
    $("form").submit(function(e){
       $(".err").hide() ;
       // Nick validation
       var nick = $("#nick") ;
       var err = false ;
       if ( nick.val().length == 0) {
           nick.next().show() ;
           err = true;
       }
       // Password validation
       var pass1 = $("#pass") ;
       var pass2 = $("#conf_pass") ;

       if ( pass1.val().length < 5 || pass1.val()  != pass2.val()) {
           pass1.next().show();
           err = true;
       }
       // Age validation
       var age = $("#age") ;
       if ( isNaN(age.val()) || age.val() < 12 ) {
           age.next().show() ;
           err = true ;
       }
       // Listbox validation
       var type = $("#type") ;
       if ( type.val() == "") {
           type.next().show() ;
           err = true;
       }
        // Policy
       var agree = $("#agree") ;
       if ( !agree.is(":checked") ) {
           agree.next().show();
           err = true;
       }
       if ( err ) {
          e.preventDefault() ;
      }
    });
});
```

Server-side Form Validation

All data from $_POST/$_GET are in string format

Non-Empty String Validation

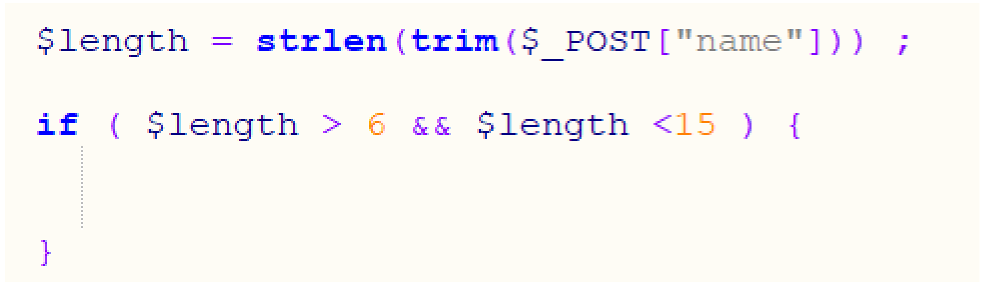
- trim(str) : removes whitespaces around the string
- strlen(str) : return the number of chars

```
if ( $_POST["name"] == "" ) {
// name = "     ", it takes the name as non-empty string

}
```

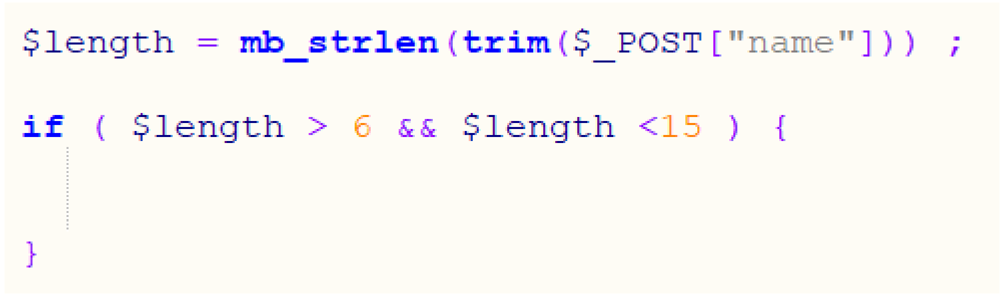
```
if ( strlen(trim($_POST["name"])) === 0 ) {


}
```

Input Length Validation

- Be aware of utf-8, utf-16 encoding.



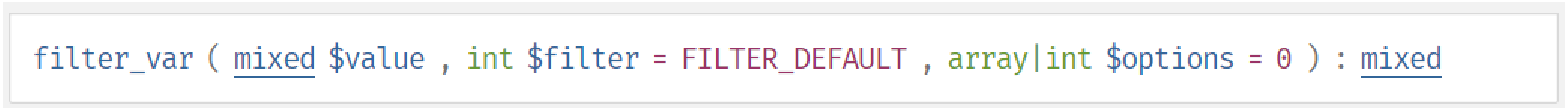
```
$length = strlen(trim($_POST["name"])) ;

if ( $length > 6 && $length <15 ) {

}
```



```
$length = mb_strlen(trim($_POST["name"])) ;

if ( $length > 6 && $length <15 ) {

}
```

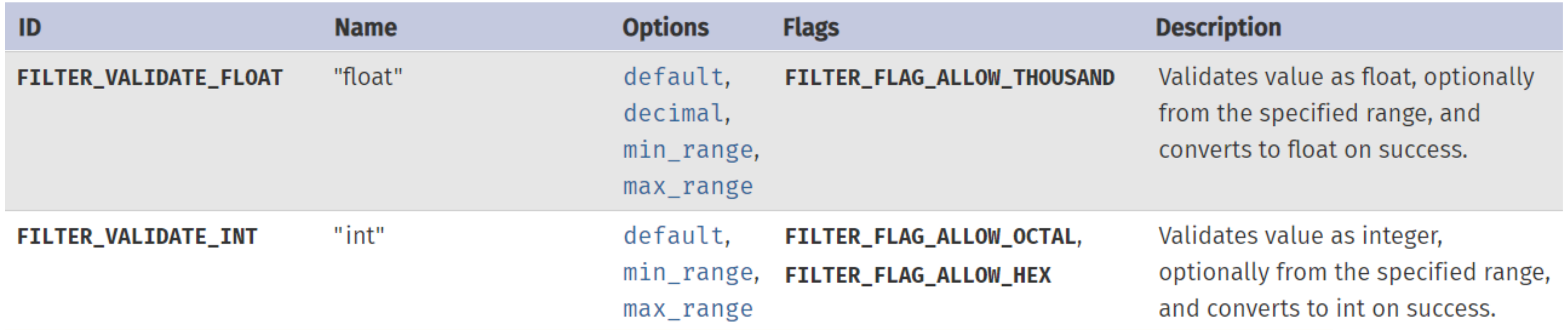
`mb_strlen` to support multi-lingual texts

Number (integer, float) Validation

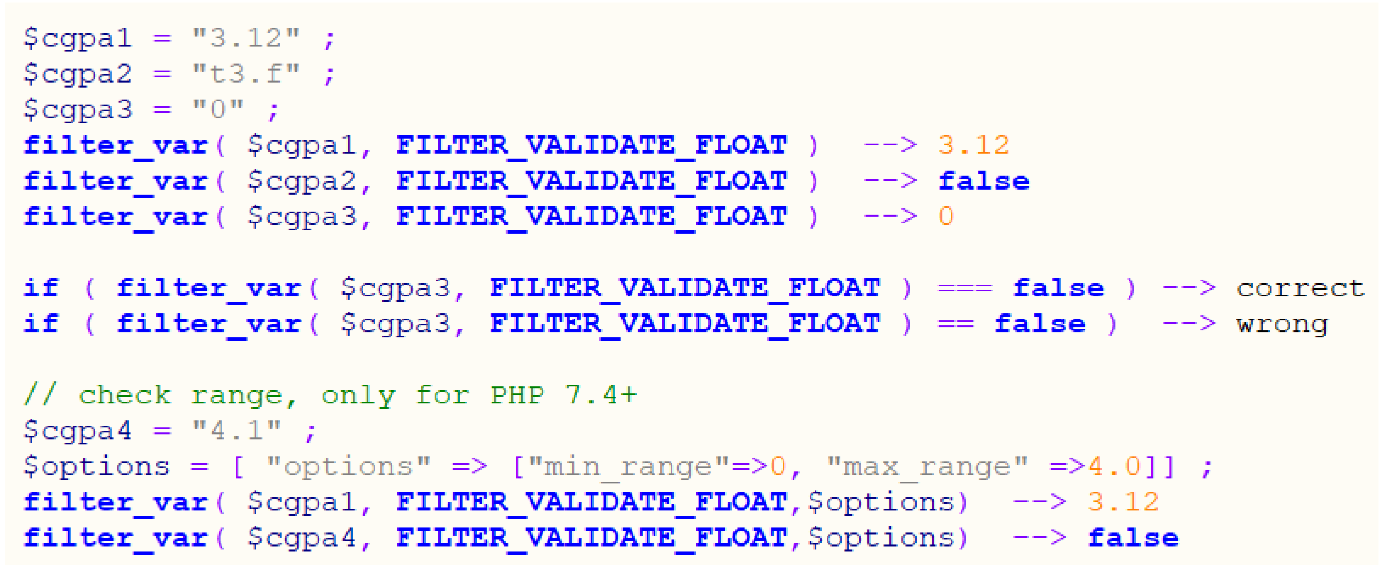


```
filter_var ( mixed $value , int $filter = FILTER_DEFAULT , array|int $options = 0 ) : mixed
```

it gets a variable “$value”, and validates based on filter id, and returns the filtered data, or false if the filter fails



ID	Name	Options	Flags	Description
FILTER_VALIDATE_FLOAT	"float"	`default`, `decimal`, `min_range`, `max_range`	**FILTER_FLAG_ALLOW_THOUSAND**	Validates value as float, optionally from the specified range, and converts to float on success.
FILTER_VALIDATE_INT	"int"	`default`, `min_range`, `max_range`	**FILTER_FLAG_ALLOW_OCTAL**, **FILTER_FLAG_ALLOW_HEX**	Validates value as integer, optionally from the specified range, and converts to int on success.



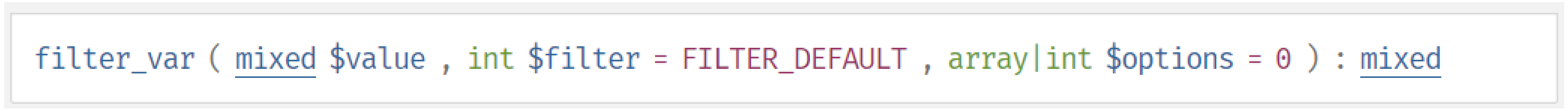
```
$cgpa1 = "3.12" ;
$cgpa2 = "t3.f" ;
$cgpa3 = "0" ;
filter_var( $cgpa1, FILTER_VALIDATE_FLOAT )  --> 3.12
filter_var( $cgpa2, FILTER_VALIDATE_FLOAT )  --> false
filter_var( $cgpa3, FILTER_VALIDATE_FLOAT )  --> 0

if ( filter_var( $cgpa3, FILTER_VALIDATE_FLOAT ) === false ) --> correct
if ( filter_var( $cgpa3, FILTER_VALIDATE_FLOAT ) == false )  --> wrong

// check range, only for PHP 7.4+
$cgpa4 = "4.1" ;
$options = [ "options" => ["min_range"=>0, "max_range" =>4.0]] ;
filter_var( $cgpa1, FILTER_VALIDATE_FLOAT,$options)  --> 3.12
filter_var( $cgpa4, FILTER_VALIDATE_FLOAT,$options)  --> false
```

Number (integer, float) Validation

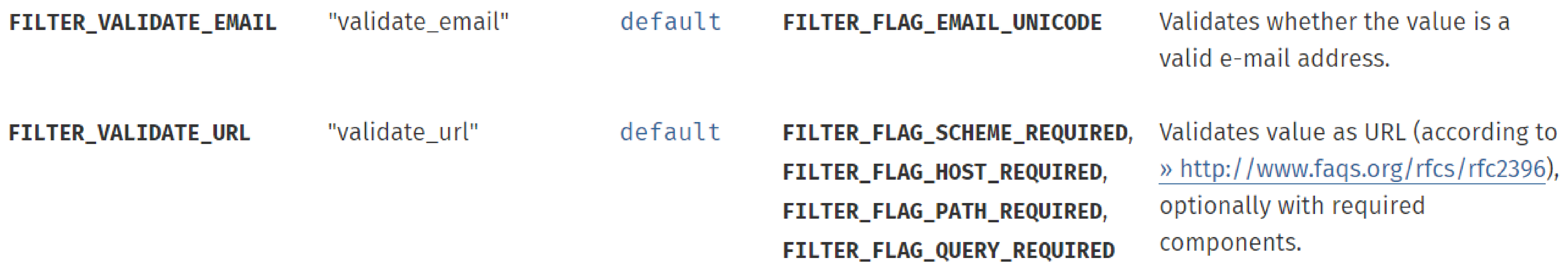
```
filter_var ( mixed $value , int $filter = FILTER_DEFAULT , array|int $options = 0 ) : mixed
```

```
$age1 = "15" ;
$age2 = "0.5" ;
$age3 = "-12" ;
filter_var( $age1, FILTER_VALIDATE_INT )  --> 15
filter_var( $age2, FILTER_VALIDATE_INT )  --> false
filter_var( $age3, FILTER_VALIDATE_INT )  --> -12

// check range, for all PHP versions
$options = [ "options" => ["min_range"=>0, "max_range" =>110]] ;
filter_var( $age3, FILTER_VALIDATE_INT, $options )  --> false

// !! Attention
// == leads to wrong result, use === (identical operator)
$age4 = "0" ;  // valid age
if ( filter_var( $age4, FILTER_VALIDATE_INT, $options ) == false ) {
    echo "<p>Invalid Age</p>"; // it returns "Invalid Age", which is wrong
} else {
    echo "<p>Age is Valid</p>" ;
}
// Hexadecimal Number String
$n = "0xFF" ;
$options = [
    "options" => ["min_range" => 0, "max_range" => 512],
    "flags" => FILTER_FLAG_ALLOW_HEX
] ;
filter_var($n, FILTER_VALIDATE_INT, $options) --> 255 or FF in hex
```

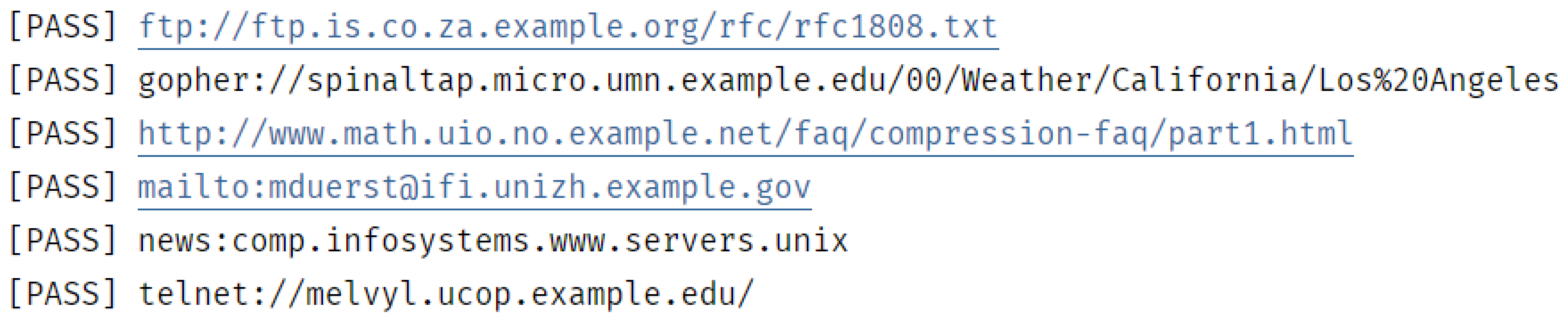
Email, URL Validation



```
filter_var ( mixed $value , int $filter = FILTER_DEFAULT , array|int $options = 0 ) : mixed
```



FILTER_VALIDATE_EMAIL	"validate_email"	default	**FILTER_FLAG_EMAIL_UNICODE**	Validates whether the value is a valid e-mail address.
FILTER_VALIDATE_URL	"validate_url"	default	**FILTER_FLAG_SCHEME_REQUIRED**, **FILTER_FLAG_HOST_REQUIRED**, **FILTER_FLAG_PATH_REQUIRED**, **FILTER_FLAG_QUERY_REQUIRED**	Validates value as URL (according to » http://www.faqs.org/rfcs/rfc2396), optionally with required components.



```
[PASS] ftp://ftp.is.co.za.example.org/rfc/rfc1808.txt
[PASS] gopher://spinaltap.micro.umn.example.edu/00/Weather/California/Los%20Angeles
[PASS] http://www.math.uio.no.example.net/faq/compression-faq/part1.html
[PASS] mailto:mduerst@ifi.unizh.example.gov
[PASS] news:comp.infosystems.www.servers.unix
[PASS] telnet://melvyl.ucop.example.edu/
```

HTML Sanitization

Sanitization is about removing illegal characters or transforming harmful characters such as <, >, “, ‘ to html entities (&lt; &gt; etc) in a string.
All inputs should be validated, and all outputs should be sanitized. Output means “echo” or putting a string into a database table or saving into a file.

Removing html tags

```
$input = "<p>This is a <span>text</span></p>" ;
// remove all html tags
$removed = filter_var($input, FILTER_SANITIZE_STRING) ;
// $removed = "This is a text" ;
```

```
<p>This is a <span>text</span></p>
```

```
This is a text
```

Transforming html tags

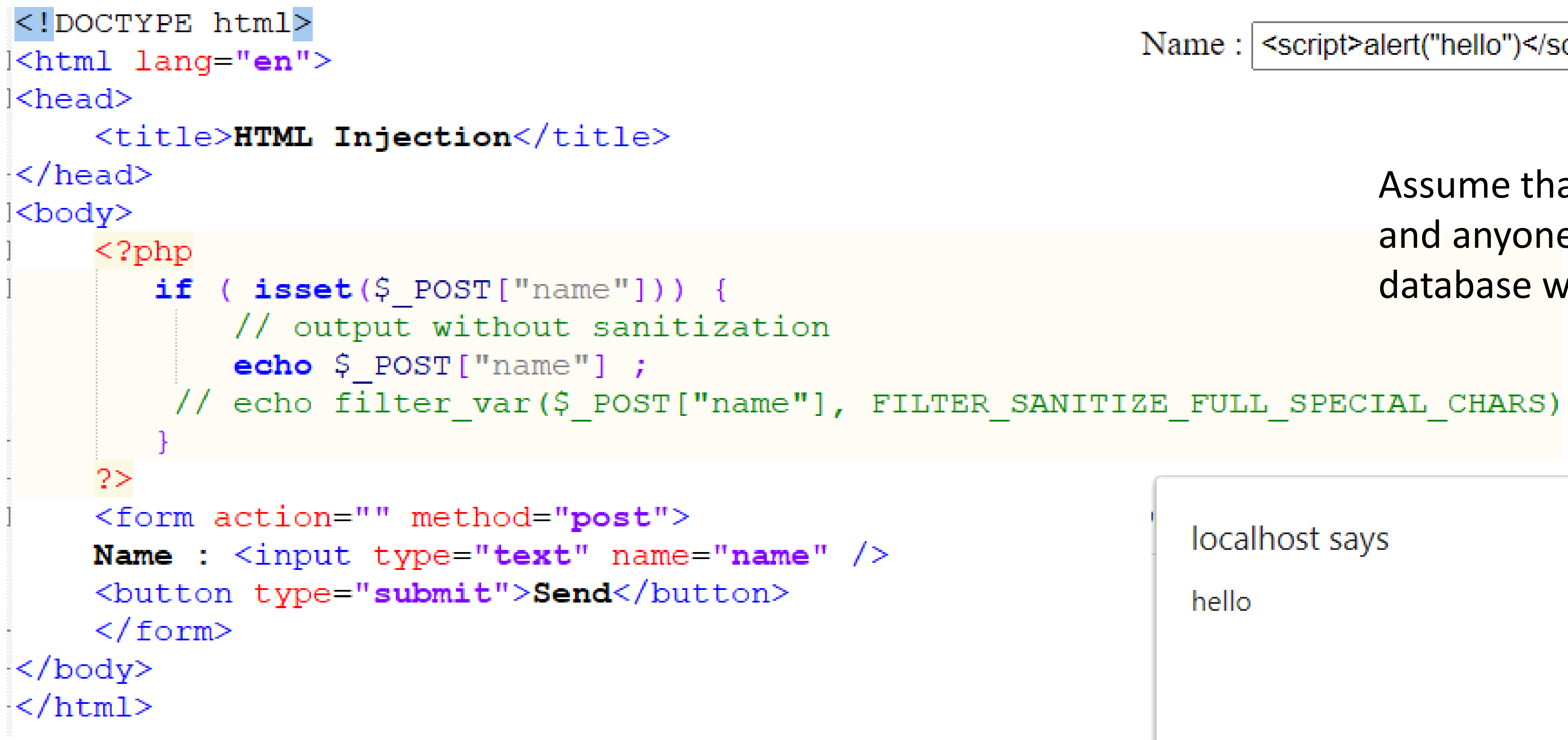
Transforms html special characters such as **<** : &lt; **>**: &gt;

```
$input = "<p>This is a <span>text</span></p>" ;
$transformed = filter_var($input, FILTER_SANITIZE_FULL_SPECIAL_CHARS) ;
// $transformed = "&lt;p&gt;This is a &lt;span&gt;text&lt;/span&gt;&lt;/p&gt;gt;"
```

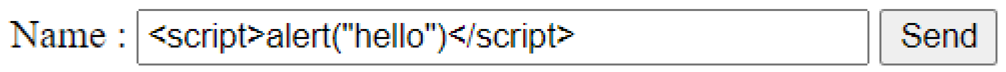
```
<p>This is a <span>text</span></p>
```

```
&lt;p&gt;This is a &lt;span&gt;text&lt;/span&gt;&lt;/p&gt;gt;
```

HTML/JS Injection

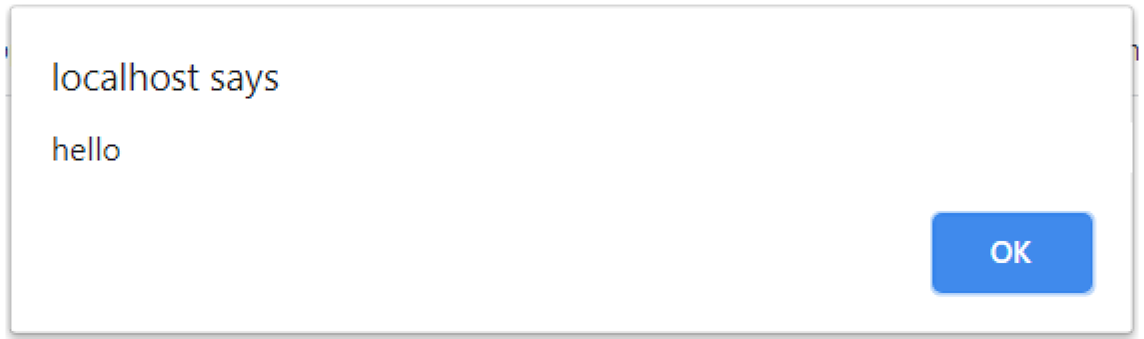


```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>HTML Injection</title>
</head>
<body>
    <?php
      if ( isset($_POST["name"])) {
          // output without sanitization
          echo $_POST["name"] ;
        // echo filter_var($_POST["name"], FILTER_SANITIZE_FULL_SPECIAL_CHARS)
      }
    ?>
    <form action="" method="post">
    Name : <input type="text" name="name" />
    <button type="submit">Send</button>
    </form>
</body>
</html>
```



Name : <script>alert("hello")</script> Send

Assume that this name is stored in a database, and anyone who wants to see the name from database will execute the script above.

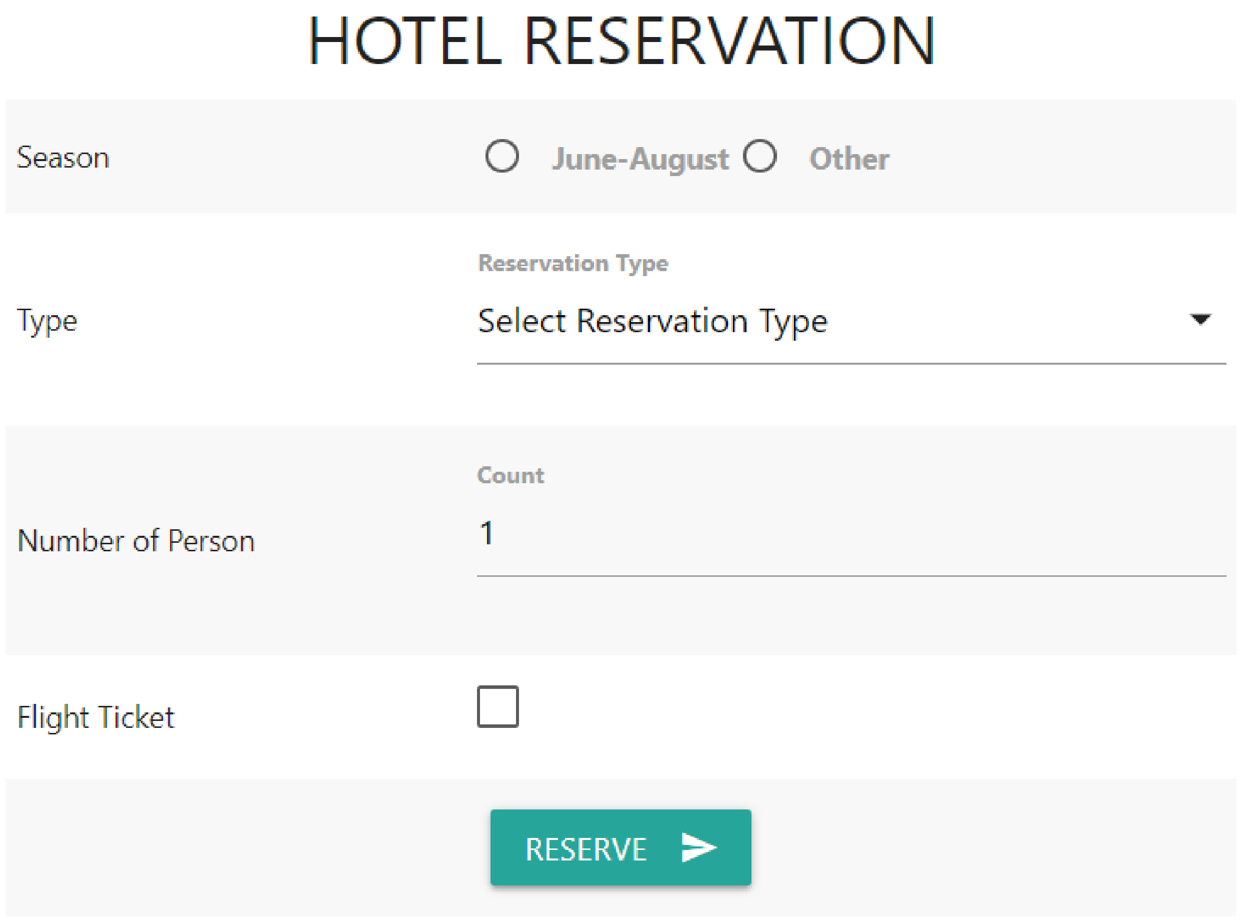


localhost says

hello

Example

Field	Rule
Season	"0" or "1"
Type	"0" or "1"
Count	posititive integer


HOTEL RESERVATION

Season	○ June-August ○ Other
Type	Reservation Type: Select Reservation Type
Number of Person	Count: 1
Flight Ticket	☐

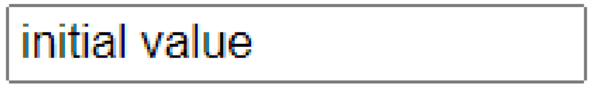
Validation

```php
if ( $_SERVER["REQUEST_METHOD"] == "POST") {
    extract($_POST) ;
    // Input Validation

    // to store errors
    $error = [] ;

    // test "season"
    if ( !isset($season) || !in_array($season, ["0", "1"])) {
        $error["season"] = "Season is required" ;
    }

    // test "type"
    if (!isset($type) || !in_array($type, ["0", "1"])) {
        $error["type"] = "Reservation type not selected" ;
    }

    // test "count"
    $options = ["options" => ["min_range" =>1]] ;
    if ( filter_var($count, FILTER_VALIDATE_INT, $options) === false) {
        $error["count"] = "Count is not correct" ;
    }
 }
```

Sticky Form – (initialization)

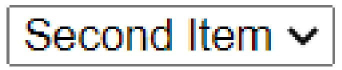
Textbox :

```
<input type="text" name="user" value="initial value" />
```


initial value

Listbox :

```
<select name="type">
  <option value="0" >First Item</option>
  <option value="1" selected="selected">Second Item</option>
  <option value="2" >Third Item</option>
</select>
```


Second Item

Checkbox :

```
<input type="checkbox" name="flight" checked />
```

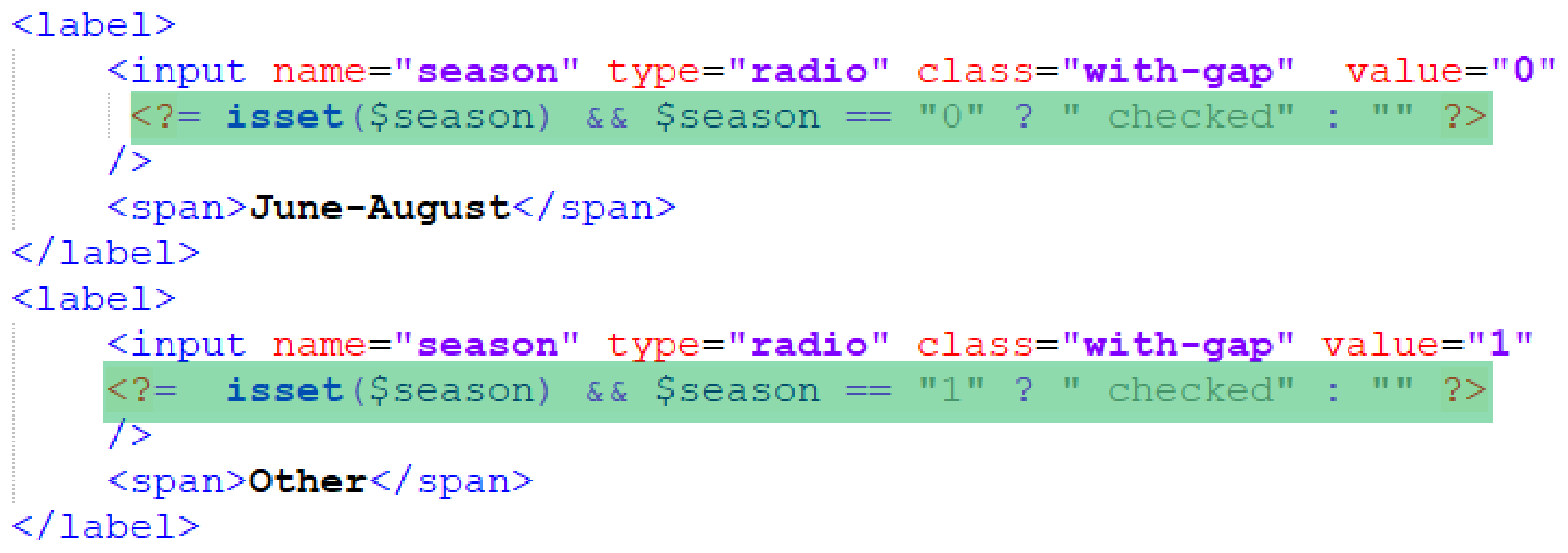
☑

Radio Button :

```
<input type="radio" name="season" value="0" checked />
```

◉

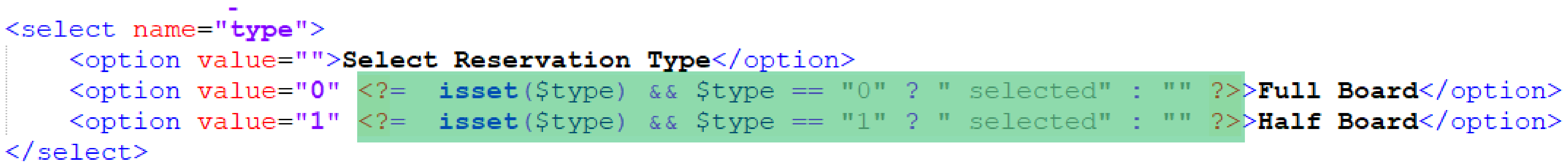
Sticky Form – Radio Button



```
<label>
    <input name="season" type="radio" class="with-gap"  value="0"
     <?= isset($season) && $season == "0" ? " checked" : "" ?>
    />
    <span>June-August</span>
</label>
<label>
    <input name="season" type="radio" class="with-gap" value="1"
    <?=  isset($season) && $season == "1" ? " checked" : "" ?>
    />
    <span>Other</span>
</label>
```

It initializes the radio buttons depending on the value of **$season** from previous POST

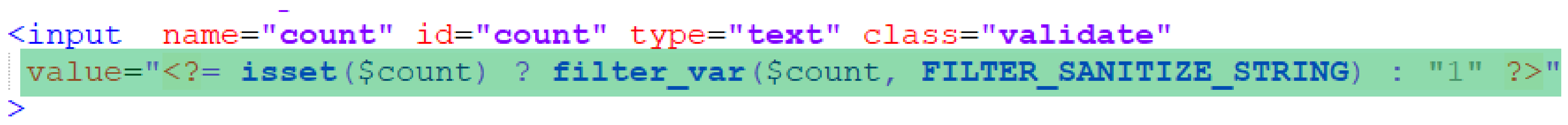
Sticky Form – Listbox



```
<select name="type">
    <option value="">Select Reservation Type</option>
    <option value="0" <?=  isset($type) && $type == "0" ? " selected" : "" ?>>Full Board</option>
    <option value="1" <?=  isset($type) && $type == "1" ? " selected" : "" ?>>Half Board</option>
</select>
```

It selects the list item based on the value of **$type** from previous POST

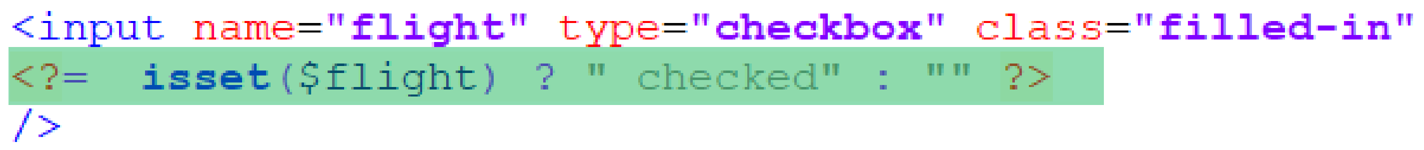
Sticky Form – Text/Password/Textarea



```
<input  name="count" id="count" type="text" class="validate"
value="<?= isset($count) ? filter_var($count, FILTER_SANITIZE_STRING) : "1" ?>"
>
```

$count is used in the output, this is why it must be sanitized to prevent html/js injection

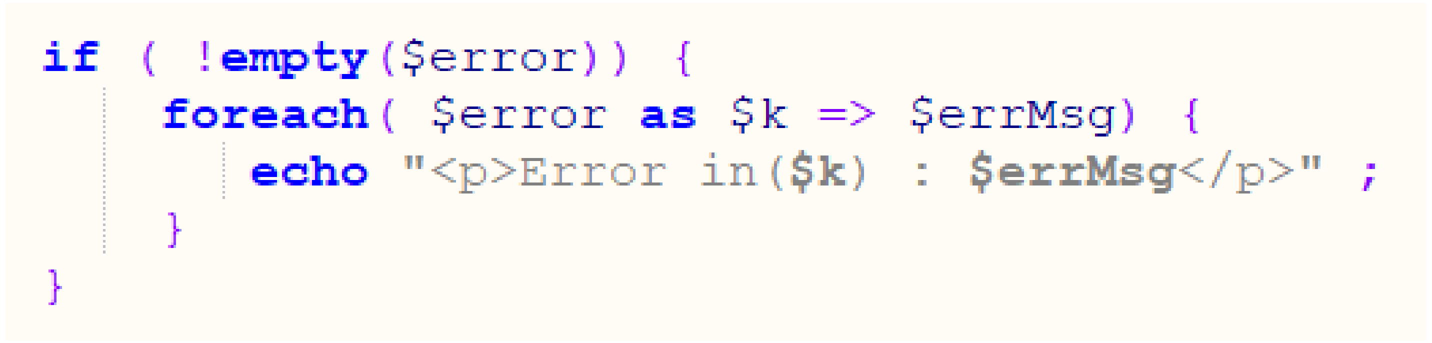
Sticky Form – Checkbox



```
<input name="flight" type="checkbox" class="filled-in"
<?=  isset($flight) ? " checked" : "" ?>
/>
```

$flight from the previous form POST is used to set the initial state of the checkbox.

Display Errors



```
if ( !empty($error)) {
    foreach( $error as $k => $errMsg) {
        echo "<p>Error in($k) : $errMsg</p>" ;
    }
}
```

Validation part fills the errors within **$error** array if it encounters any invalid input.
If the array is empty, it means there is no error.