

# PHP Global Variables

- Part of the goal of PHP is to make interacting with HTTP and HTML as easy as possible.
- PHP processes the incoming HTTP request based on the protocol specifications and drops the data into various **super global** variables (usually arrays).

```
<h1>Contents of the $_GET array</h1>
<p>Using print_r:</p>
<pre>
<?php
    print_r($_GET);
?>
</pre>
<p>Using var_dump:</p>
<pre>
<?php
    var_dump($_GET);
?>
</pre>
```

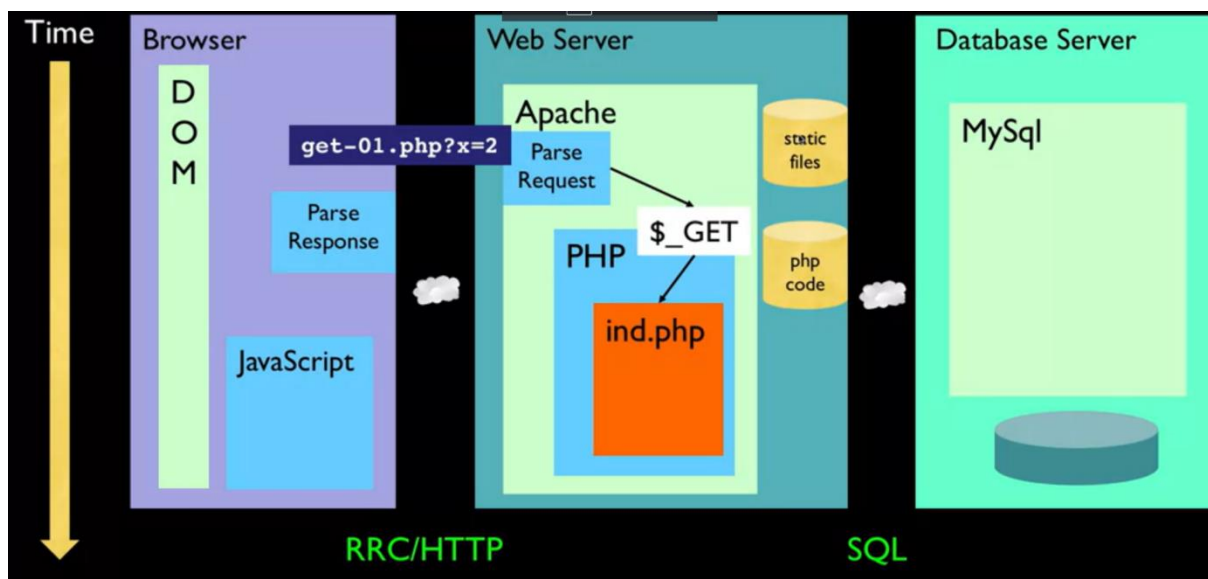
Contents of the `$_GET` array

Using `print_r`:

```
Array
(
    [x] => 2
    [y] => 4
)
```

Using `var_dump`:

```
array(2) {
    ["x"]=>
    string(1) "2"
    ["y"]=>
    string(1) "4"
}
```



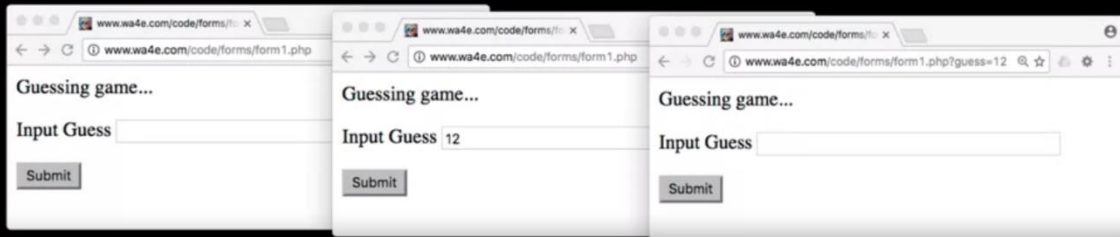
# Forms – User Input / Action

```
<p>Guessing game...</p>
<form>
  <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" id="guess"/></p>
    <input type="submit"/>
</form>
```



# Forms Submit Data

```
<p>Guessing game...</p>
<form>
  <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" id="guess"/></p>
    <input type="submit"/>
</form>
```



```
<p>Guessing game...</p>
<form>
  <p><label for="guess">Input Guess</label>
    <input type="text" name="guess" id="guess"/></p>
    <input type="submit"/>
</form>
<pre>
$_GET:
<?php
  print_r($_GET);
?>
</pre>
```

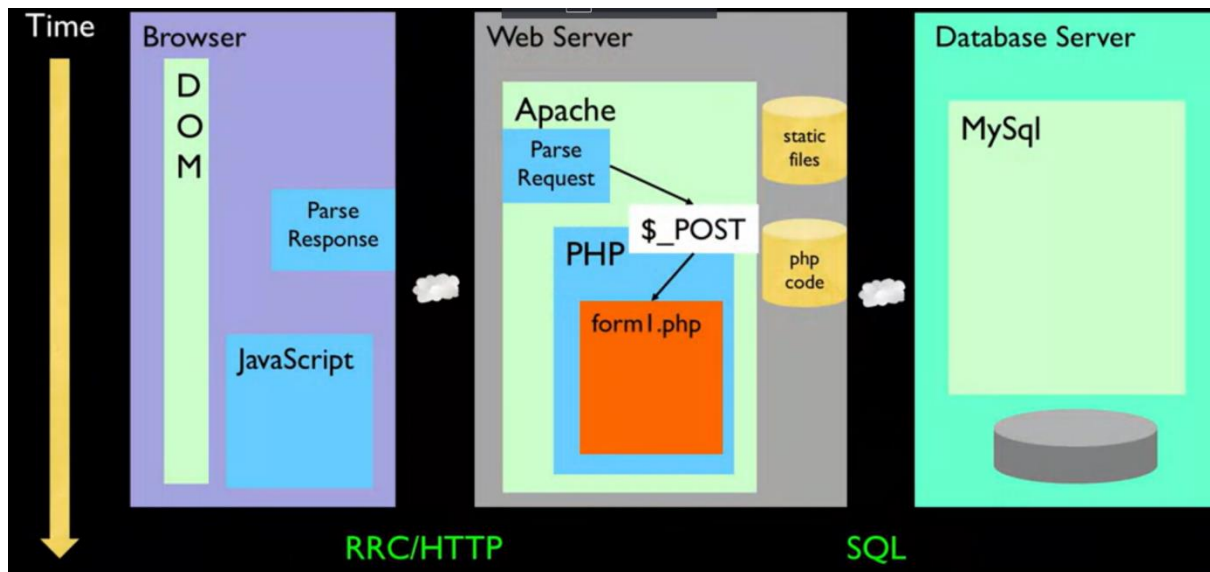
form2.php



## Using GET and POST with Forms

PHP `$_GET` is a PHP super global variable which is used to collect form data after submitting an HTML form with `method="get"`. `$_GET` can also collect data sent in the URL.

PHP `$_POST` is a PHP super global variable which is used to collect form data after submitting an HTML form with `method="post"`. `$_POST` is also widely used to pass variables.

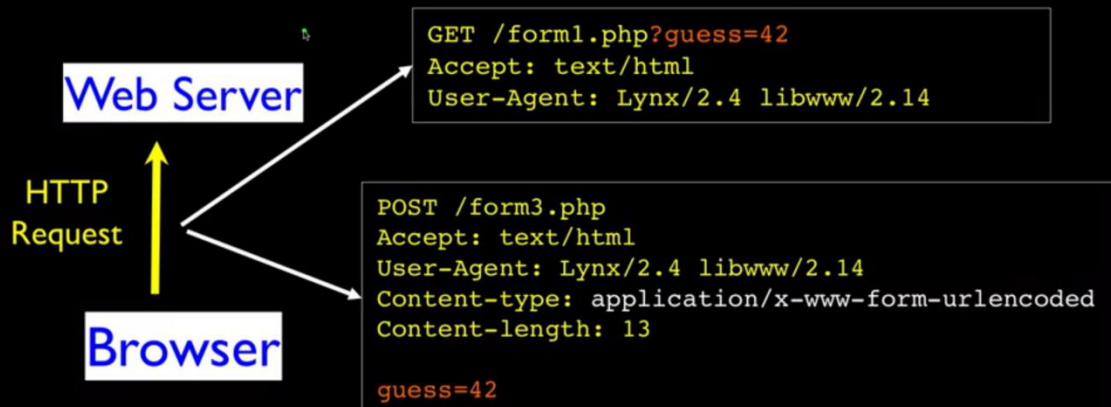


```
<p>Guessing game...</p>
<form method="post">
  <p><label for="guess">Input Guess</label>
  <input type="text" name="guess" size="40" id="guess" /></p>
  <input type="submit" />
</form>
<pre>
$_POST:
<?php
  print_r($_POST);
?>
$_GET:
<?php
  print_r($_GET);
?>
</pre>
```

form3.php

```
$_POST:
Array
(
    [guess] => 12
)
$_GET:
Array
(
)
```

# Passing Parameters to The Server



`<input type="text" name="guess" id="yourid" />`

## Forms GET vs. POST

Two ways the browser can send parameters to the web server

- **GET** - Parameters are placed on the URL which is retrieved.
- **POST** - The URL is retrieved and parameters are appended to the request in the the HTTP connection.

## Rules of the POST/GET Choice

- POST is used when data is being created or modified.
- GET is used when you are reading or searching things.
- Web search spiders will follow GET URLs but generally not POST URLs.
- GET URLs should be “idempotent” - the same URL should give the “same thing” each time you access it.
- GET has an upper limit of the number of bytes of parameters and values (think about 2K).

## GET vs. POST

Both GET and POST create an array (e.g. `array( key1 => value1, key2 => value2, key3 => value3, ...)`). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as `$_GET` and `$_POST`. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

`$_GET` is an array of variables passed to the current script via the URL parameters.

`$_POST` is an array of variables passed to the current script via the HTTP POST method.

### When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data. GET should NEVER be used for sending passwords or other sensitive information!

### When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page. Developers prefer POST for sending form data

# Other Input Types

- Text
- Password
- Radio Button
- Check Box
- Select / Drop-Down
- Textarea

<http://www.wa4e.com/code/forms/more.php>

more.php

```
<p>Preferred Time:<br/>
<input type="radio" name="when" value="am">AM<br>
<input type="radio" name="when" value="pm" checked>PM</p>
```

```
$_POST:
Array(
    ...
    [nick] => BK
    [when] => pm
    [class] => si502
    ...
)
```

```
<p>Classes taken:<br/>
<input type="checkbox" name="class1" value="si502" checked>
    SI502 - Networked Tech<br>
<input type="checkbox" name="class2" value="si539">
    SI539 - App Engine<br>
<input type="checkbox" name="class3">
    SI543 - Java<br> </p>
```

```
$_POST:
Array(
    ...
    [when] => pm
    [class1] => si502
    [soda] => 0
    ...
)
```

```
$_POST:
Array(
    ...
    [when] => pm
    [class3] => on
    [soda] => 0
    ...
)
```



```

<p><label for="inp06">Which soda:
<select name="soda" id="inp06">
  <option value="0">-- Please Select --</option>
  <option value="1">Coke</option>
  <option value="2">Pepsi</option>
  <option value="3">Mountain Dew</option>
  <option value="4">Orange Juice</option>
  <option value="5">Lemonade</option>
</select>
</p>

```

The values can be any string, but numbers are used quite often.

more.php

```

$_POST:
Array(
    ...
    [class] => si502
    [soda] => 0
    [snack] => peanuts
    ...
)

```

```

<p><label for="inp07">Which snack:
<select name="snack" id="inp07">
  <option value="">-- Please Select --</option>
  <option value="chips">Chips</option>
  <option value="peanuts" selected>Peanuts</option>
  <option value="cookie">Cookie</option>
</select>
</p>

```

more.php

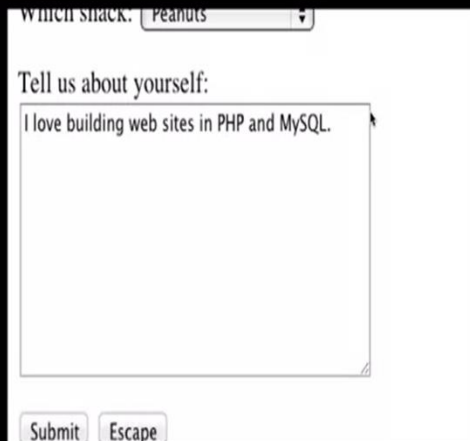
```

$_POST:
Array(
    ...
    [class] => si502
    [soda] => 0
    [snack] => peanuts
    ...
)

```

more.php

```
<p><label for="inp08">Tell us about yourself:<br/>
  <textarea rows="10" cols="40" id="inp08" name="about">
    I love building web sites in PHP and MySQL.
  </textarea>
</p>
```



```
$_POST:
Array(
    ...
    [about] => I love
building web sites in
PHP and MySQL.
    [dopost] => Submit
    ...
)
```

more.php

```
<p><label for="inp09">Which are awesome?<br/>
<select multiple="multiple" name="code[ ]" id="inp09">
  <option value="python">Python</option>
  <option value="css">CSS</option>
  <option value="html">HTML</option>
  <option value="php">PHP</option>
</select>
```



```
$_POST:
Array(
    ...
    [code] => Array
        (
            [0] => css
            [1] => html
        )
    [dopost] => Submit
    ...
)
```



```

<p>
<input type="submit" name="dopost" value="Submit"/>
<input type="button"
  onclick="location.href='http://www.wa4e.com/'; return false;"
  value="Escape">
</p>

```

```

$_POST:
Array(
    ...
    [dopost] => Submit
    ...
)

```

On submit input types, the text is both in the UI and in `$_POST` so we tend to look for the key, not the value.

## Data Validation (Server Side)

### Incoming Data Validation

Making sure all user data is present and the correct format before proceeding

- Non-empty `strlen($var) > 0`
- A number `is_numeric($var)`
- An email address `strpos($var, '@') > 0`
- Or `filter_var($var, FILTER_VALIDATE_EMAIL) !== false`
- ....

## PHP \$\_REQUEST

PHP `$_REQUEST` is a PHP super global variable which is used to collect data after submitting an HTML form.