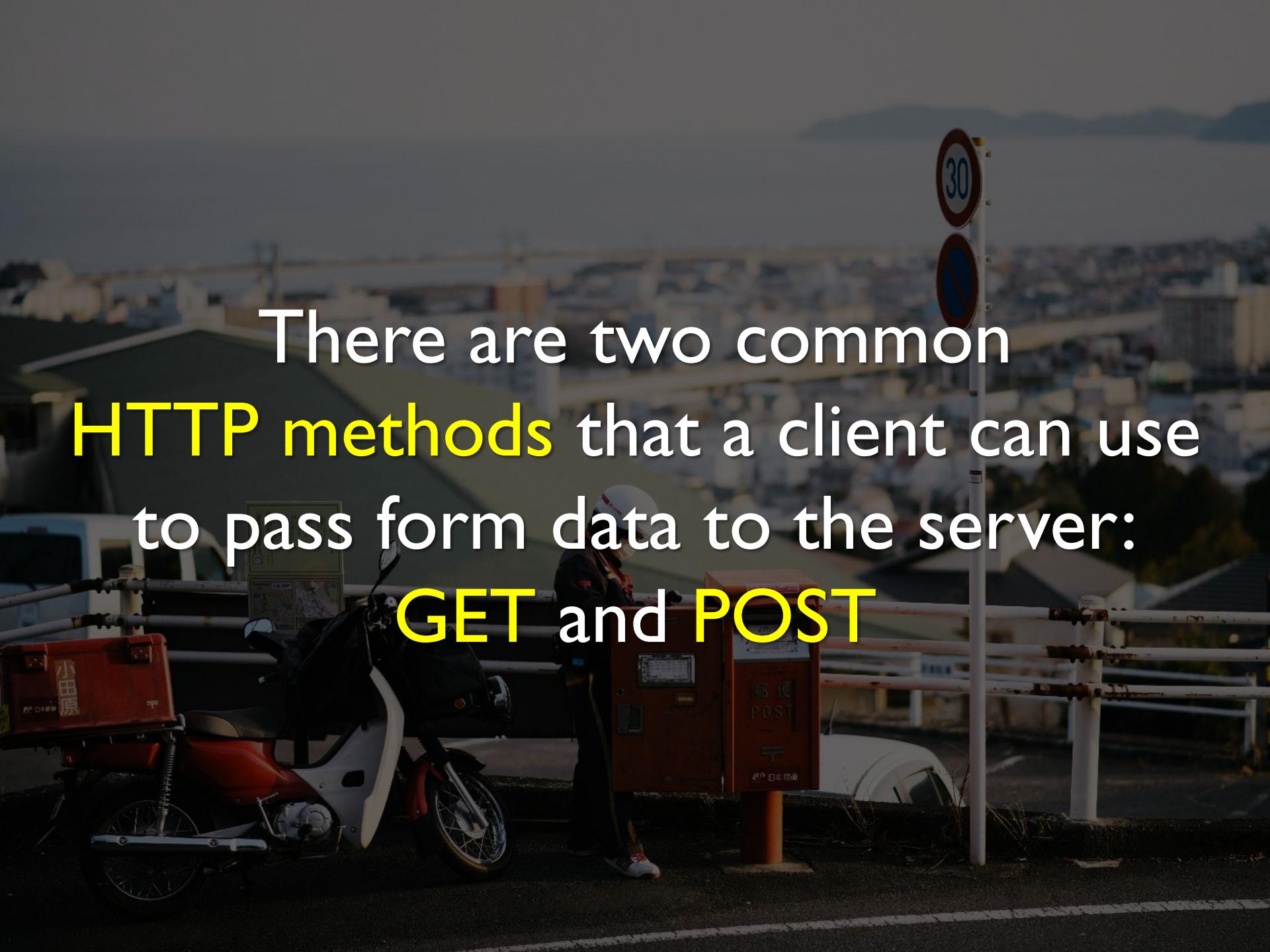


<form> *Handling*



30

A photograph of a person standing next to a red and white motorcycle. The person is wearing a white shirt and dark pants. In the background, there is a road sign indicating a speed limit of 30 and a no-parking zone. The scene is set outdoors with buildings and hills visible in the distance.

There are two common
HTTP methods that a client can use
to pass form data to the server:
GET and **POST**



Requests made by typing
the URL in the browser
uses the GET method



GET requests can also
be sent from a **form**

```
<form method="get">  
</form>
```

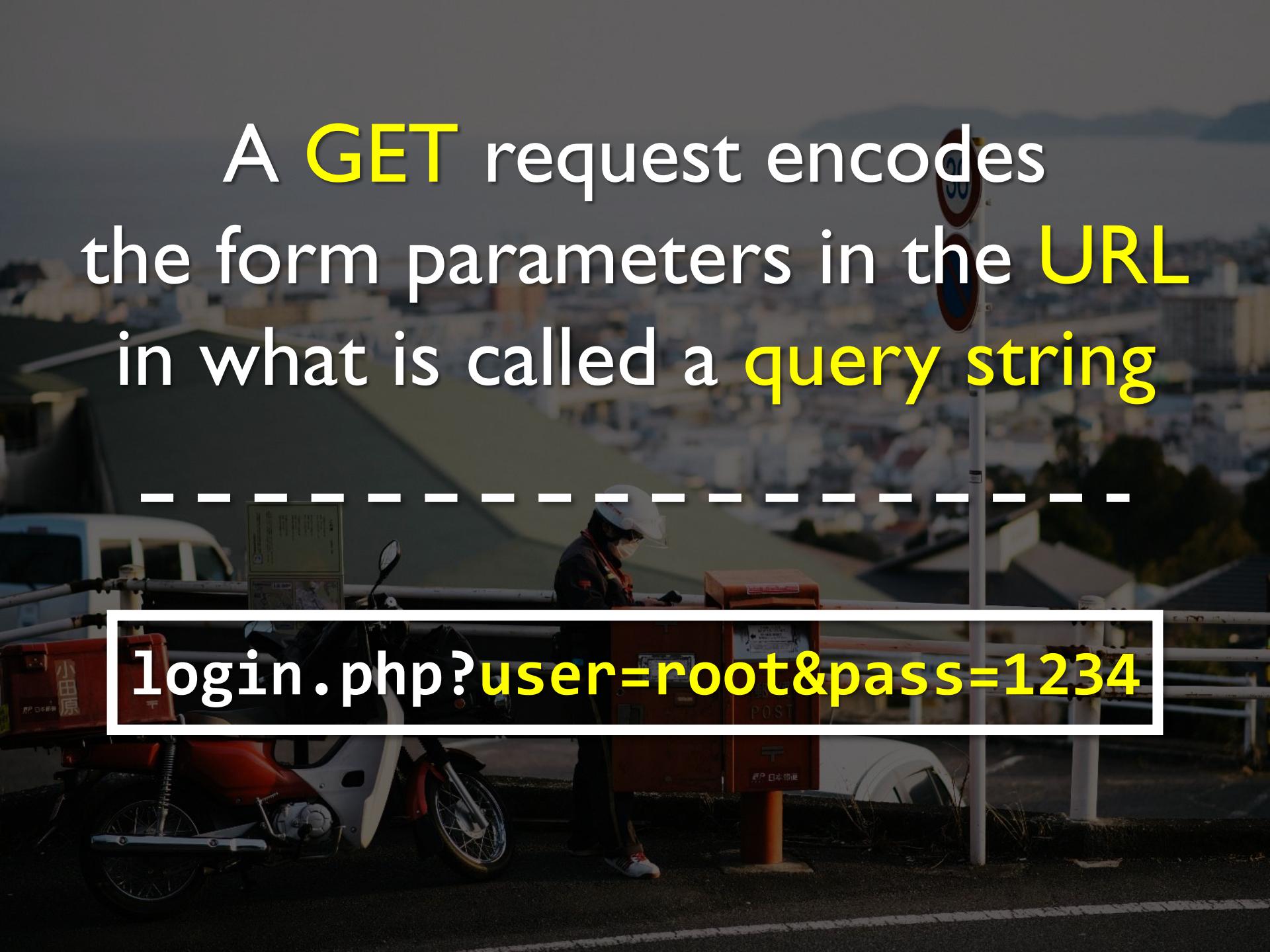
While POST requests must
be made from a form

```
<form method="post">  
</form>
```

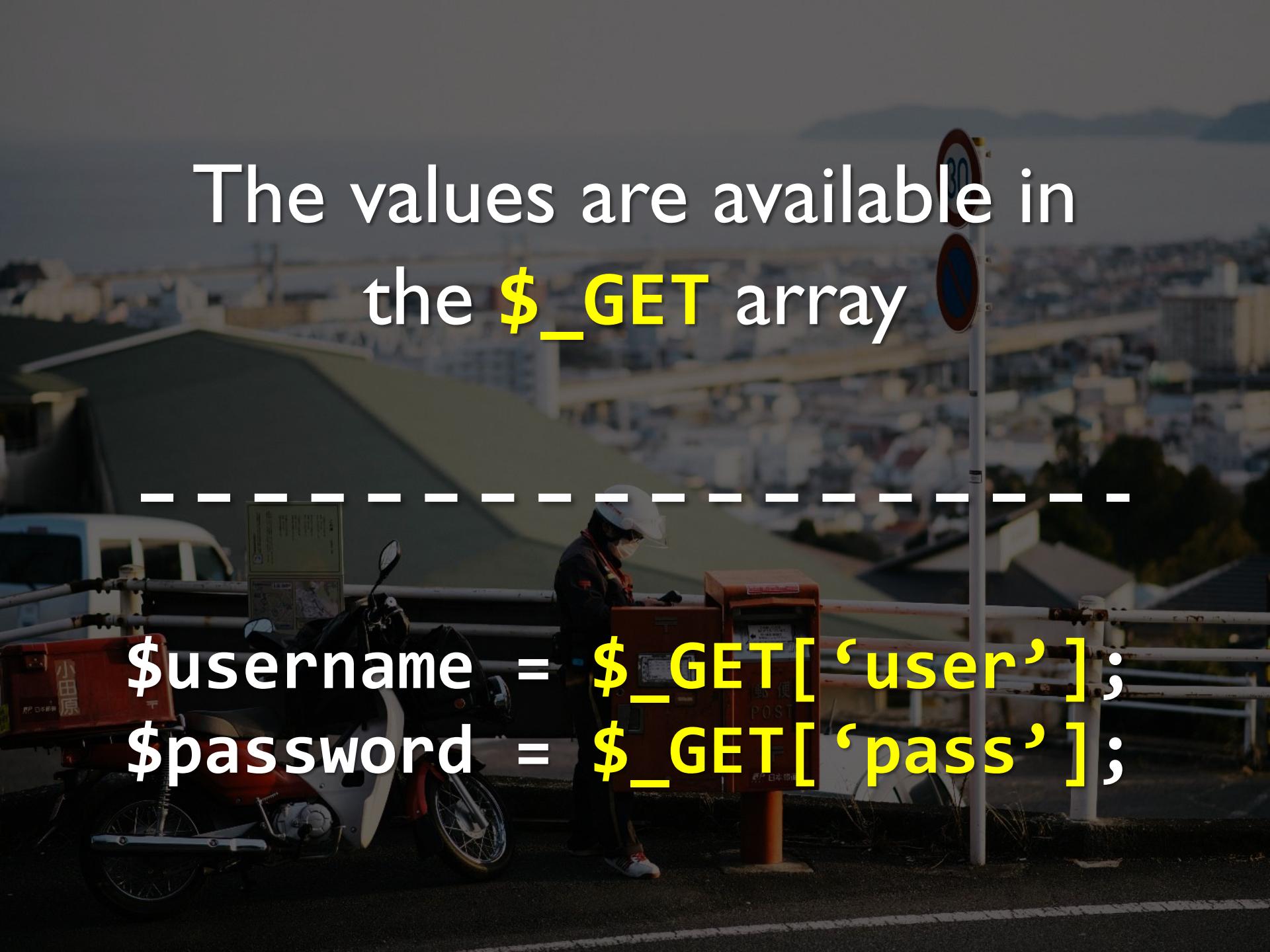


The most visible difference
between **GET** and **POST**
is the **URL** line

A **GET** request encodes
the form parameters in the **URL**
in what is called a **query string**

A photograph of a person wearing a white hard hat and a dark jacket standing next to a red and white motorcycle. They are leaning against a red utility box. In the background, there's a road, some buildings, and a blue circular road sign. The image has a semi-transparent dark overlay.

`login.php?user=root&pass=1234`

A photograph of a person wearing a white hard hat and a light-colored face mask, standing next to a red and white motorcycle. They are looking down at a small electronic device or map held in their hands. In the background, there's a white van, some utility poles, and a road sign indicating a speed limit of 30 km/h. The scene is set outdoors with buildings visible in the distance.

The values are available in
the **`$_GET`** array

```
$username = $_GET['user'];  
$password = $_GET['pass'];
```

A photograph of a person wearing a white helmet and a dark jacket standing next to a red and white motorcycle. They are at a roadside in Japan, with a wooden post office box labeled "郵便 POST" and "日本郵便" nearby. In the background, there's a white fence, some buildings, and a speed limit sign on a pole. The sky is overcast.

A POST request passes
the form parameters in the
body of the HTTP request
header

POST /wp-login.php HTTP/1.1

Host: rendicahya.lecture.ub.ac.id

Connection: keep-alive

Content-Length: 117

Accept: text/html

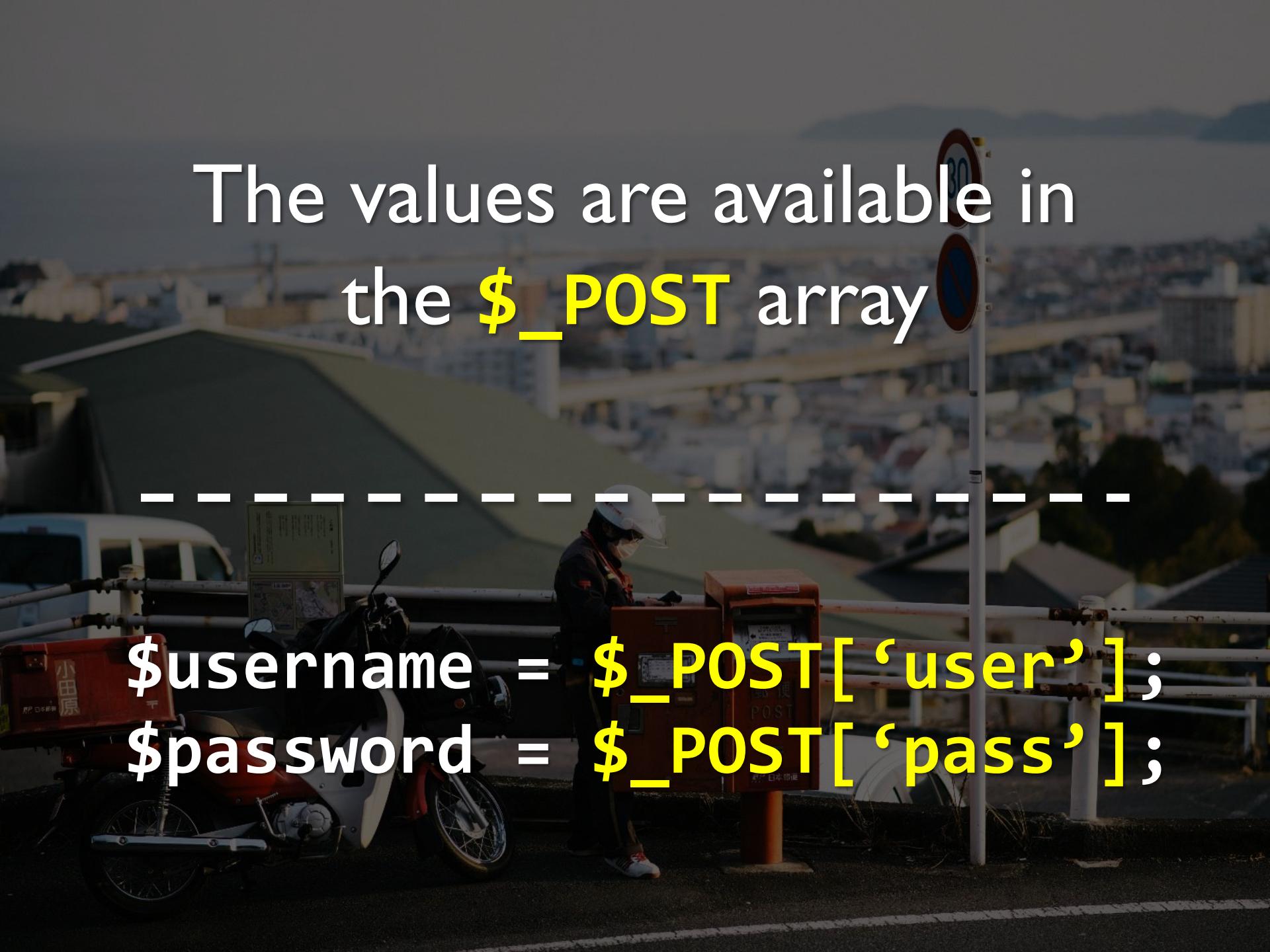
User-Agent: Chrome/40.0.2214.115

Accept-Encoding: gzip, deflate

Accept-Language: id,en-US

Referer: http://ub.ac.id

user=root&pass=1234

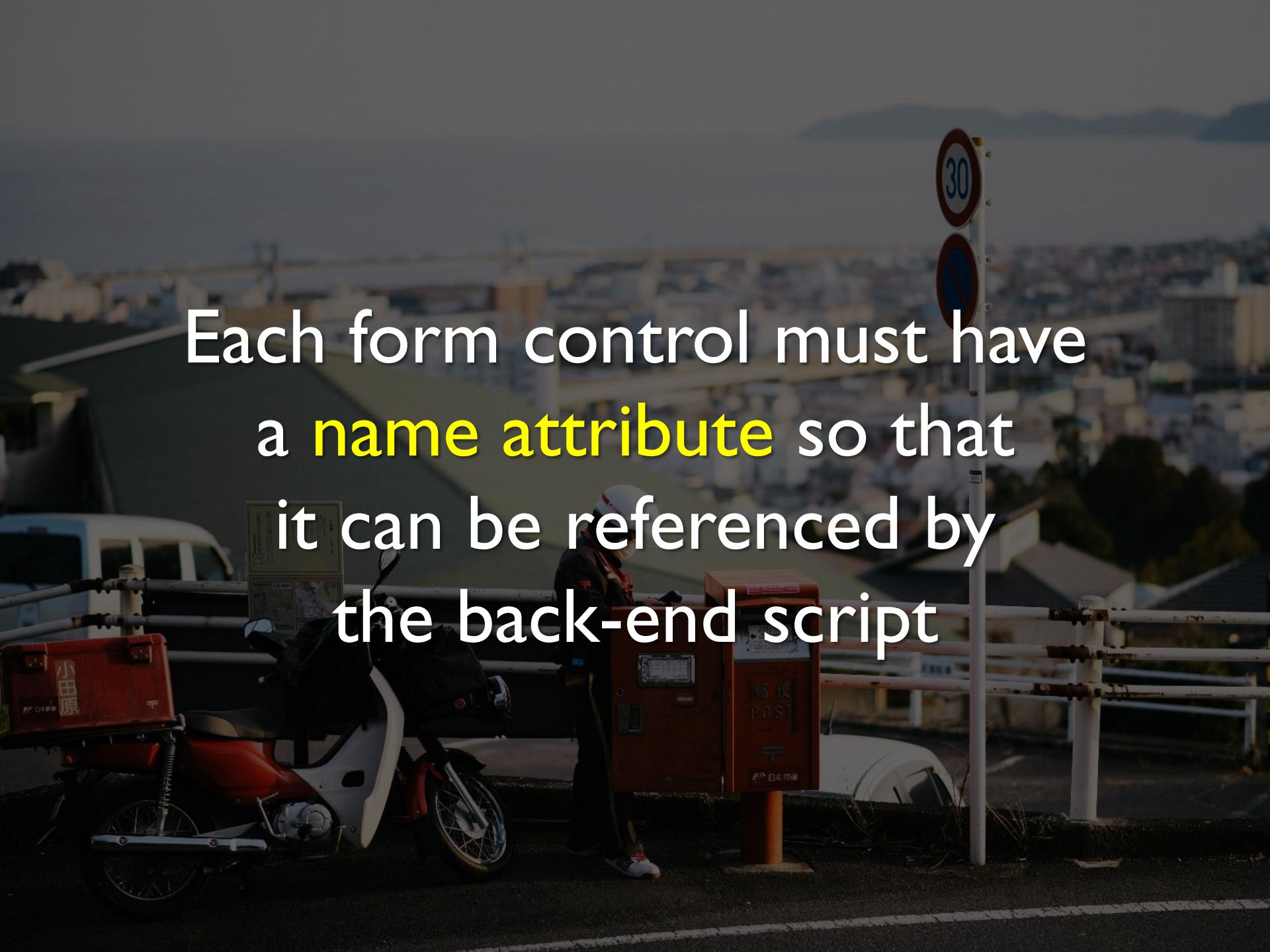
A photograph of a person wearing a white hard hat and a face mask, standing next to a red motorcycle. They are leaning against a metal railing near a bus stop. In the background, there's a road, some buildings, and a speed limit sign. A dashed horizontal line is overlaid across the middle of the image.

The values are available in
the **`$_POST`** array

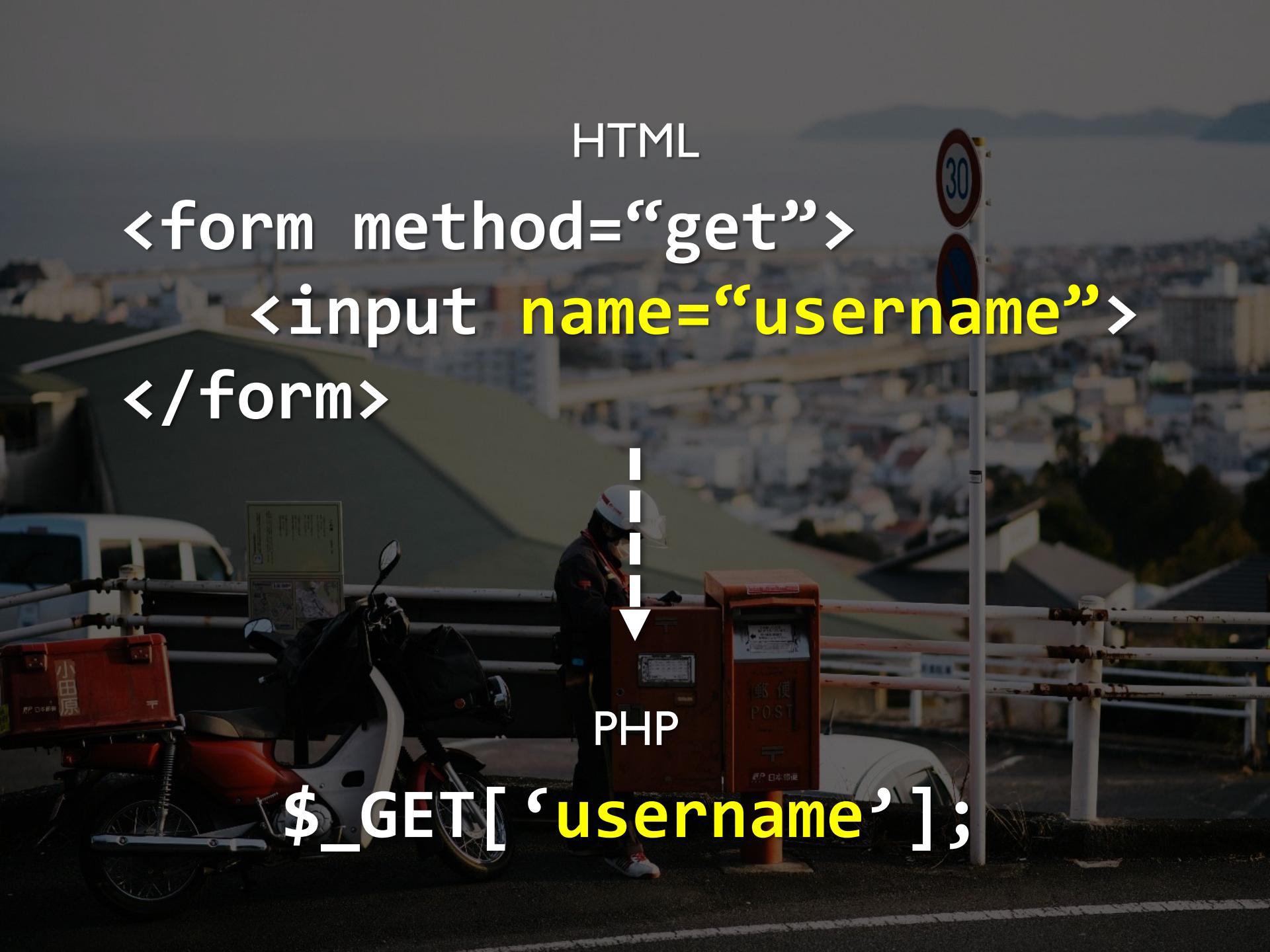
```
$username = $_POST['user'];  
$password = $_POST['pass'];
```

`$_REQUEST` array includes variables
passed from any method

```
-----  
$username = $_REQUEST['user'];  
$password = $_REQUEST['pass'];
```

A photograph of a person standing next to a red and white motorcycle. The person is wearing a white helmet and dark clothing. In the background, there is a road sign indicating a speed limit of 30 and a no-parking zone. The scene is set outdoors with buildings and hills visible in the distance.

Each form control must have
a **name** attribute so that
it can be referenced by
the back-end script

A photograph of a coastal town at sunset or sunrise. In the foreground, there's a red motorcycle, a white van, and a person in a hard hat standing next to a red wooden postbox. A speed limit sign on a pole indicates 30 km/h. The town is built on a hillside overlooking the sea, with buildings and trees visible in the background.

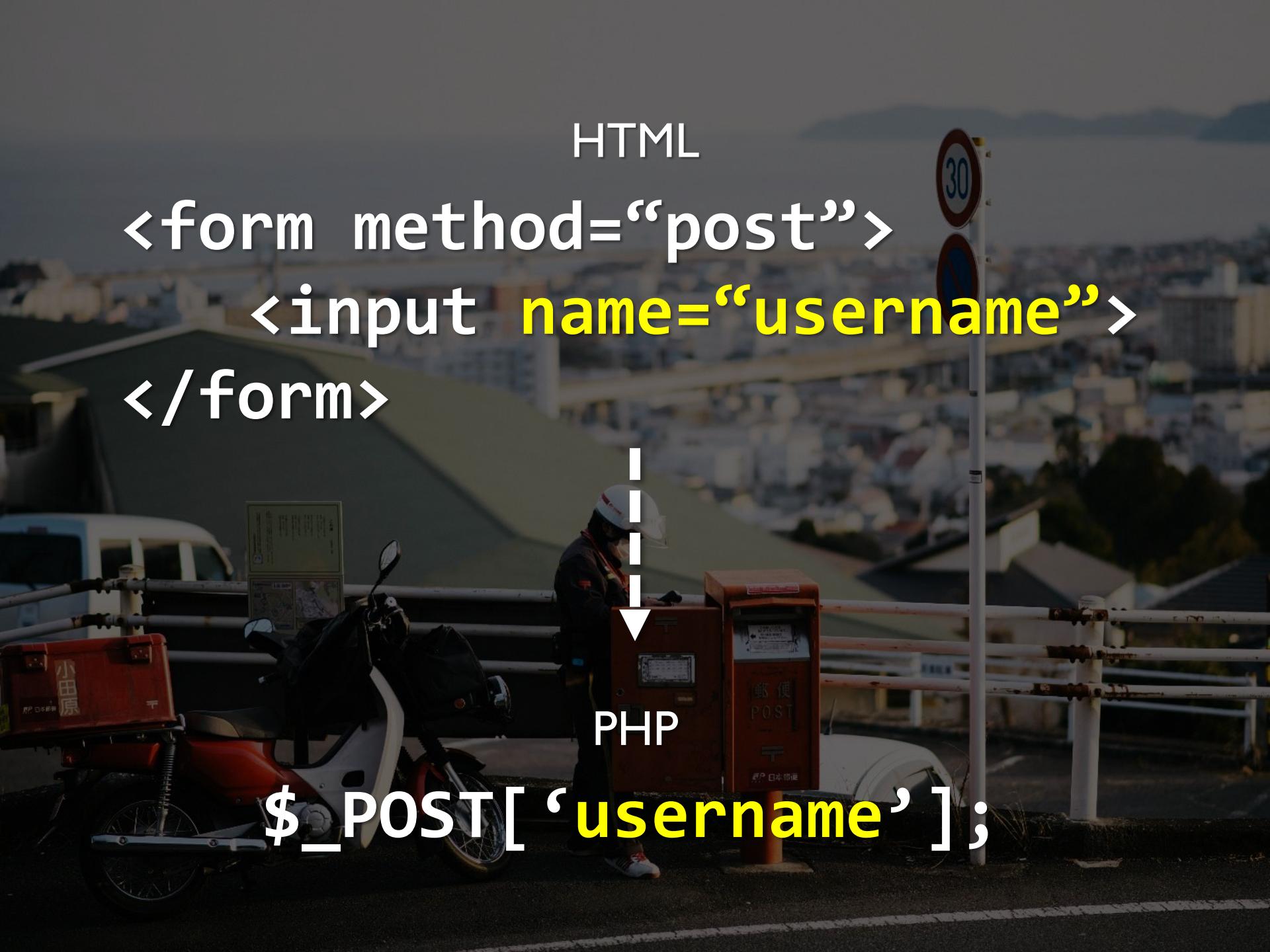
HTML

```
<form method="get">  
  <input name="username">  
</form>
```



PHP

```
$_GET['username'];
```

A photograph of a coastal town at sunset or sunrise. In the foreground, there's a red motorcycle with a sidecar, a white van, and a person in a hard hat standing next to a red postbox. A speed limit sign indicates 30 km/h. The town is built on a hillside overlooking the sea.

HTML

```
<form method="post">  
  <input name="username">  
</form>
```



PHP

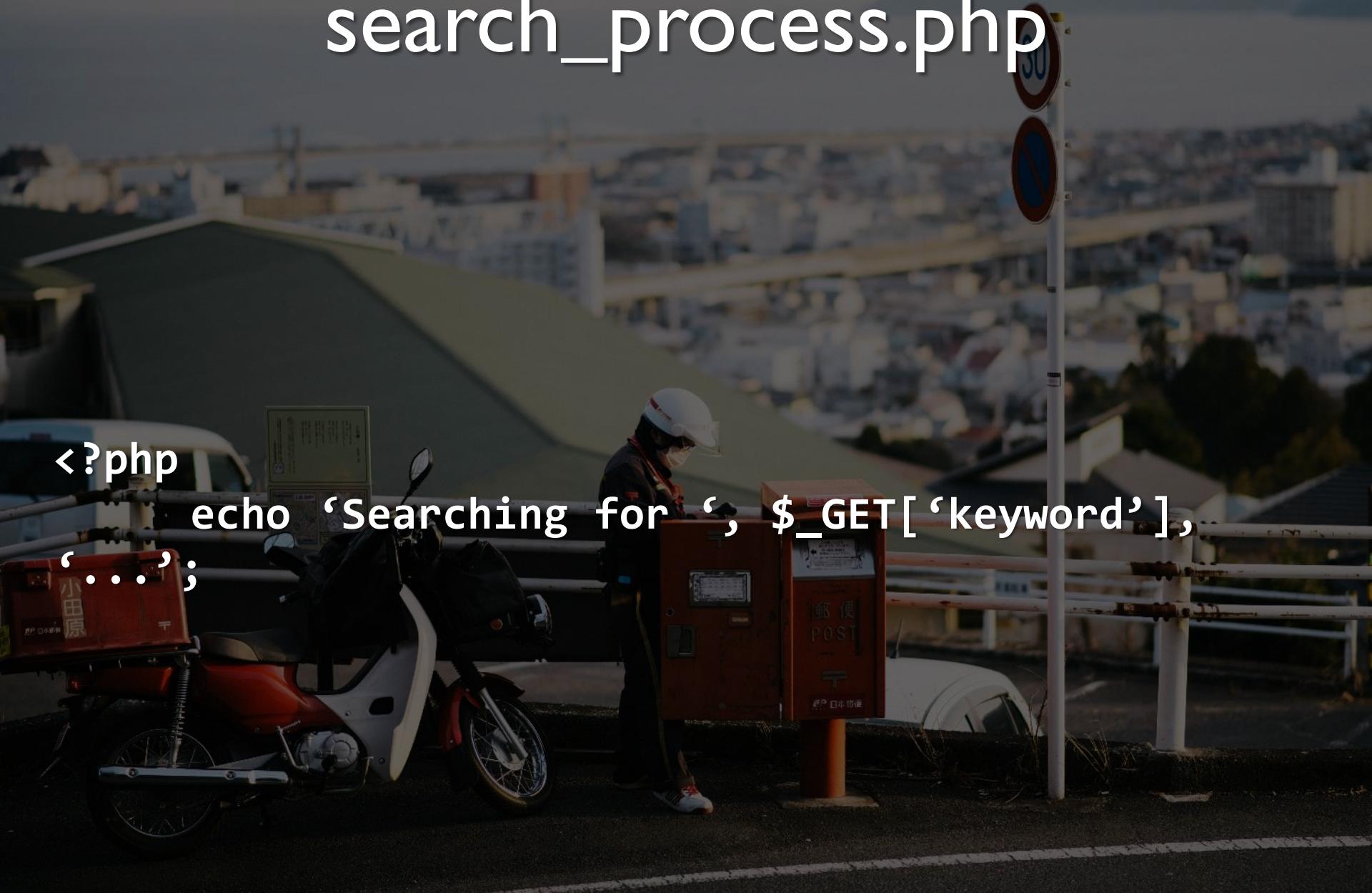
```
$_POST['username'];
```

search.php

```
...
<h1>Search</h1>
<form action="search_process.php" method="get">
    <label>Search:
        <input type="text" name="keyword">
    </label><br>
    <input type="submit" value="Search">
</form>
...
```

search_process.php

```
<?php  
    echo 'Searching for ', $_GET['keyword'],  
'...';
```



register.php

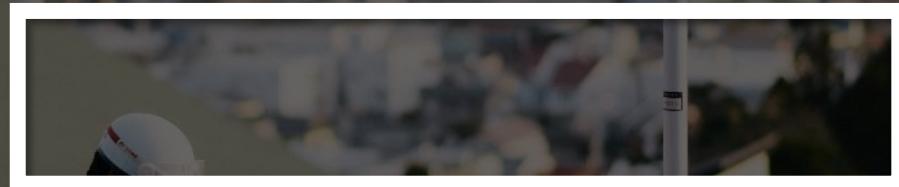
```
...  
<h1>Register</h1>  
<form action="register_process.php" method="post">  
    <label>Username:  
        <input type="text" name="username">  
    </label>  
    <label>Password:  
        <input type="password" name="password">  
    </label>  
    <input type="submit" value="Register">  
</form>  
...
```

register_process.php

```
<?php  
$username = $_POST[‘username’];  
$password = $_POST[‘password’];  
  
echo “Registering $username with password  
$password”;
```

Create a **login form** like so:

Username:



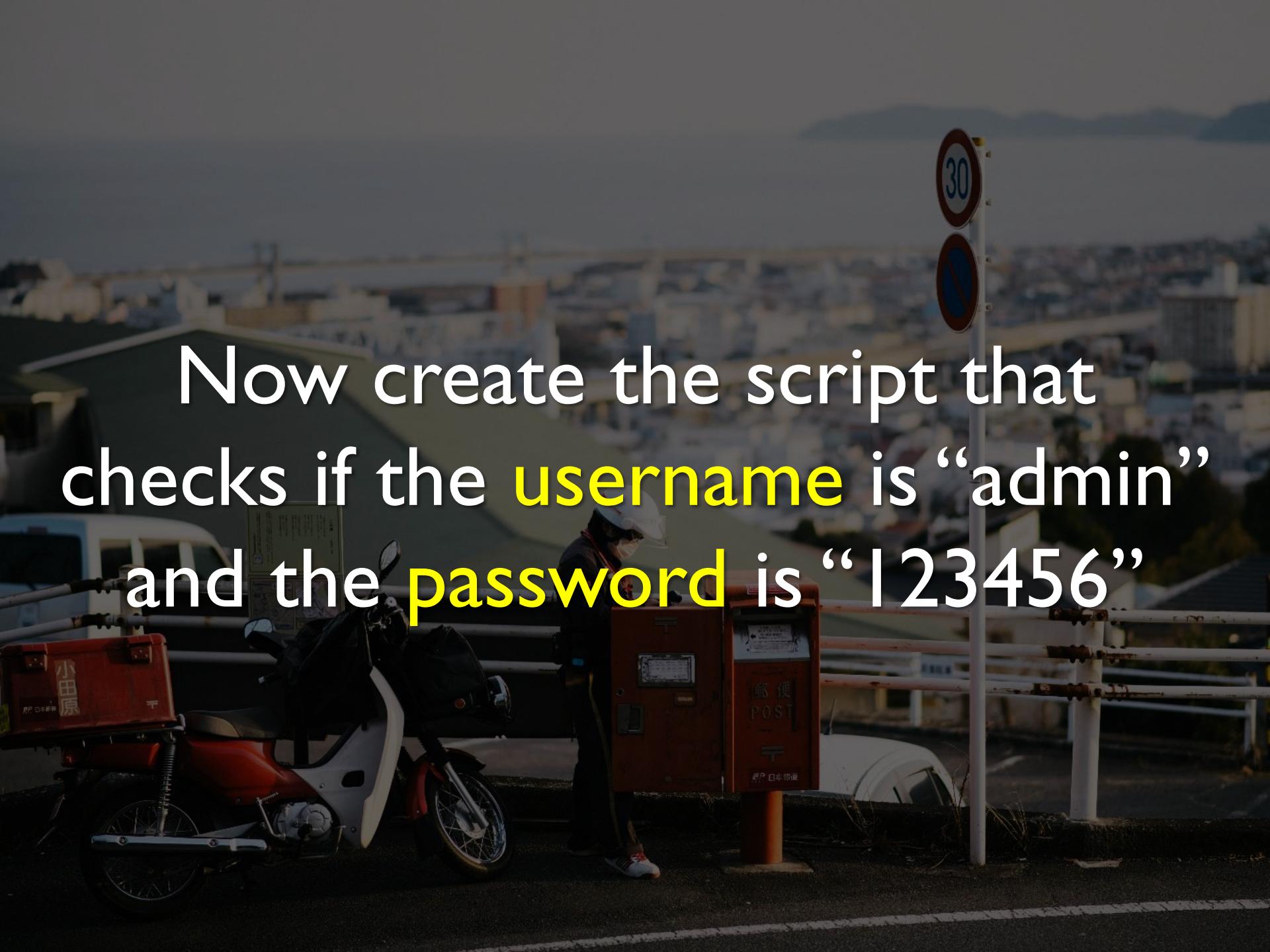
Password:



Login

login.php

```
...
<form action="login_process.php" method="post">
<label>
Username:
    <input type="text" name="username"><br>
</label>
<label>
Password:
    <input type="password" name="password"><br>
</label>
<input type="submit" value="Login">
</form>
...
```



Now create the script that
checks if the **username** is “admin”
and the **password** is “123456”

login_process.php

```
$username = $_POST[‘username’];  
$password = $_POST[‘password’];  
  
if ($username == ‘admin’ && $password == ‘123456’) {  
    echo ‘

# Login succeeds

’;  
} else {  
    echo ‘

# Login fails

’;  
}
```

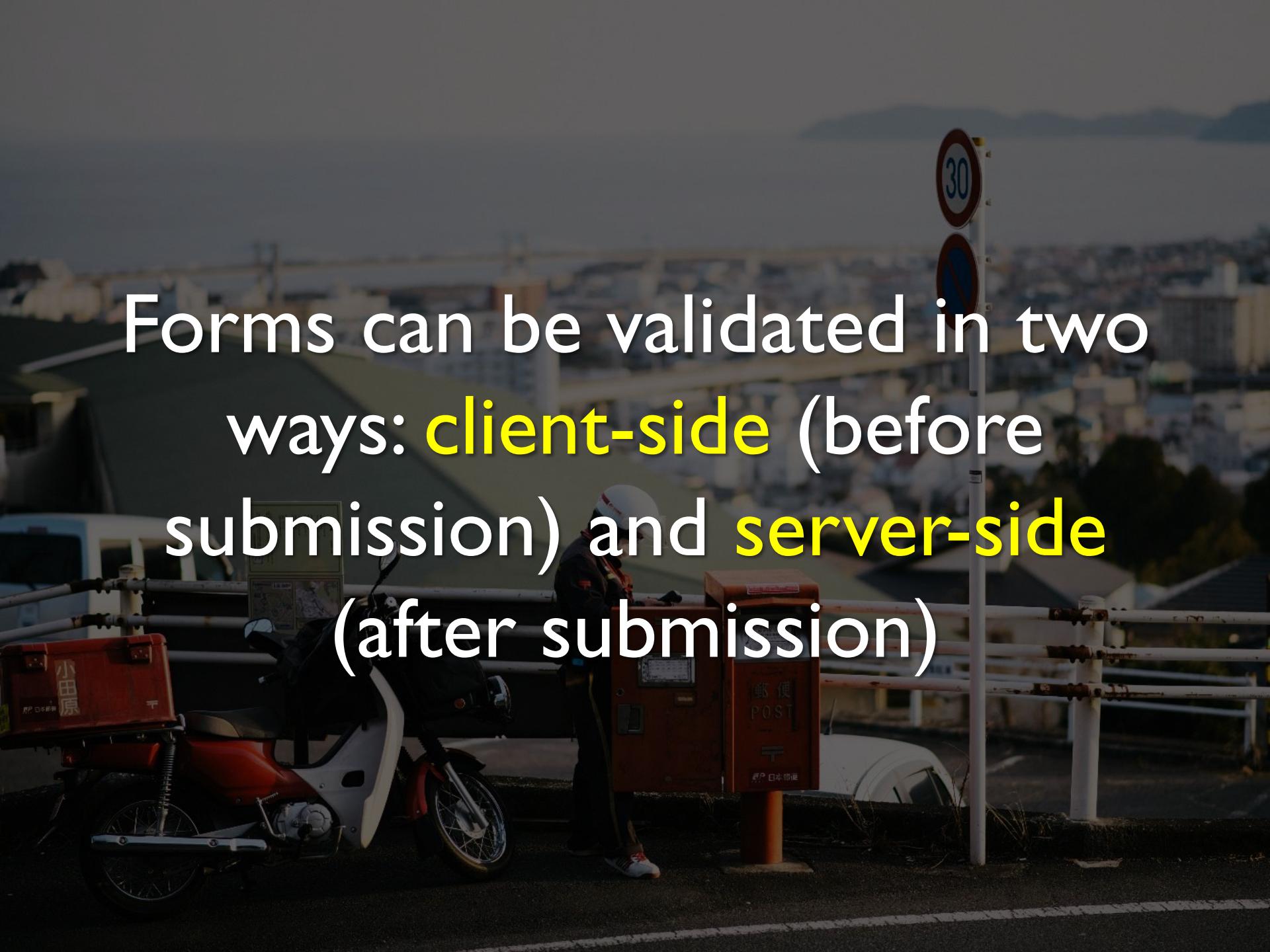


One PHP page can be used
to both generate a form
and process it

```
<html><head></head>
<body>
<?php if ($_SERVER[ 'REQUEST_METHOD' ] == 'GET'): ?>
    <form action="<?php echo $_SERVER[ 'PHP_SELF' ]; ?>" method="POST">
        Fahrenheit temperature:
        <input type="text" name="fahrenheit"><br>
        <input type="submit" value="Convert to Celsius!">
    </form>
<?php elseif ($_SERVER[ 'REQUEST_METHOD' ] == 'POST'):>
    $fahrenheit = $_POST[ 'fahrenheit' ];
    $celsius = ($fahrenheit - 32) * 5 / 9;
    printf("%.2fF is %.2fC", $fahrenheit, $celsius);
endif; ?>
</body></html>
```

A photograph of a street scene in Japan. In the foreground, a red and white motorcycle is parked on the left. A person wearing a dark jacket and light-colored pants stands near a wooden post box. To the right of the post box is a red and white speed limit sign indicating 30 km/h. In the background, there's a white fence, some buildings, and hills under a clear sky.

Every form needs to be
validated to avoid incorrect
data and malicious input



Forms can be validated in two ways: **client-side** (before submission) and **server-side** (after submission)

A photograph of a person standing next to a red and white motorcycle on a roadside. In the background, there's a speed limit sign indicating 30 km/h and a no parking sign. The scene is set against a backdrop of buildings and hills under a clear sky.

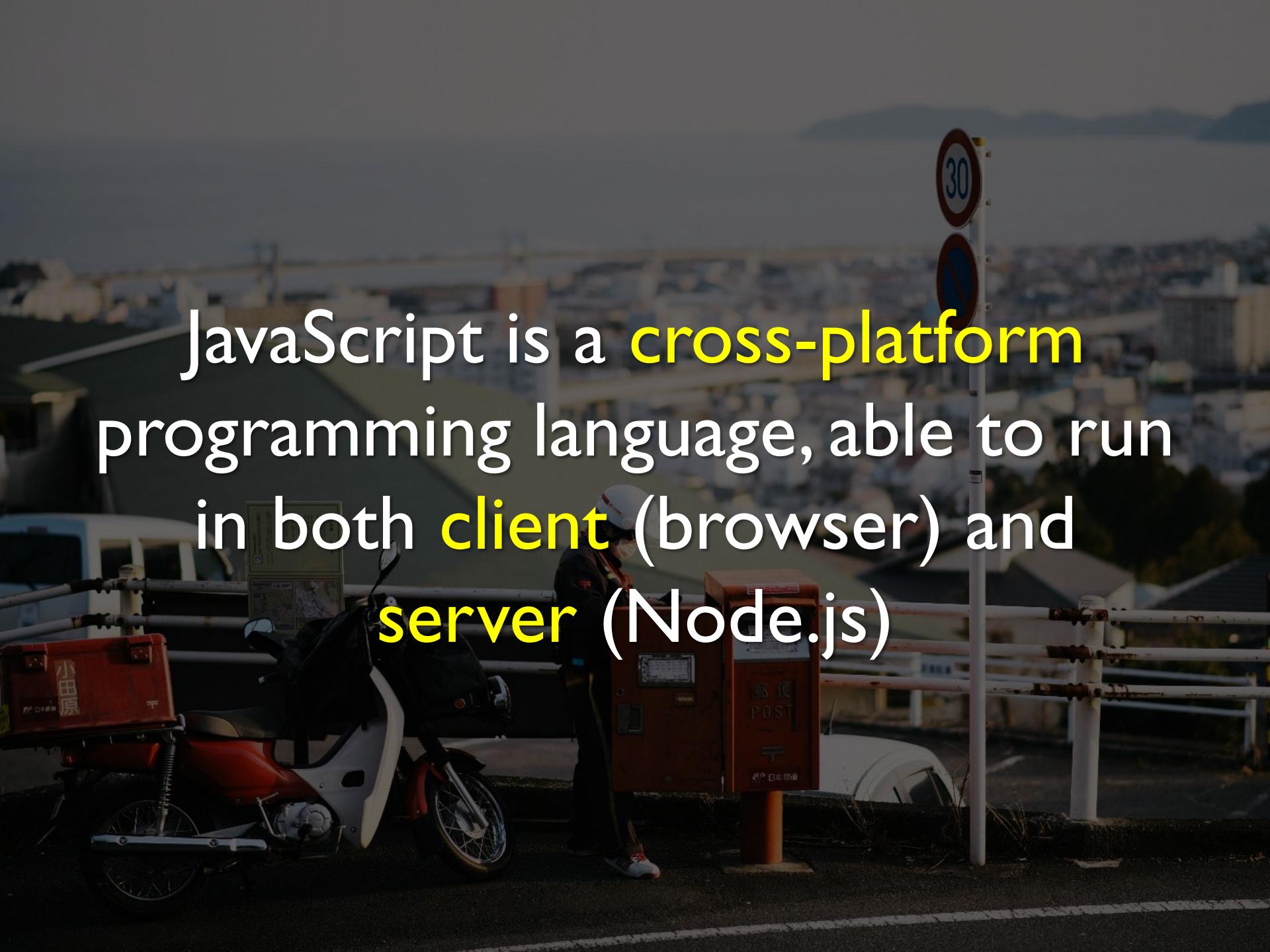
Client-side validation can be
performed with HTML or
JavaScript

login.php

```
...
<form action="login_process.php" method="post">
    <label>
        Username:
        <input type="text" name="username" required><br>
    </label>
    <label>
        Password:
        <input type="password" name="password" required><br>
    </label>
    <input type="submit" value="Login">
</form>
...
```

JavaScript & AJAX



A photograph of a person standing next to a red and white motorcycle on a roadside. In the background, there's a speed limit sign (30 km/h) and a Japanese post box. The scene is set against a backdrop of buildings and hills under a clear sky.

JavaScript is a **cross-platform**
programming language, able to run
in both **client** (browser) and
server (Node.js)

JavaScript adds **interactivity** to a web application, for example:

- Form validation
- Google's text suggestion
- Communication with server without page refresh (AJAX)
- Hide/show page elements

universitas brawijaya



+Randy

universitas brawijaya

universitas brunensis

universitas bina nusantara

universitas bunda mulia

About 708,000 results (0.51 seconds)

Brawijaya University

www.ub.ac.id/ ▾

The Head of Administration of Vocational Program of UB, Efraim Luturmas said that the event is regularly held every year for Vocational Pr... (readmore) ...

4.2 ★★★★☆ 183 Google reviews · Write a review

Jalan Veteran, Jawa Timur 65145, Indonesia
+62 341 551611[Selma UB - Undergraduate Admissions - Faculty](#)

SELMA UB - Seleksi Masuk Universitas Brawijaya

selma.ub.ac.id/ ▾ [Translate this page](#)

Seleksi Masuk Universitas Brawijaya. Akreditasi Internasional dari ABEST21 untuk Fakultas Ekonomi dan Bisnis (FEB) UB Ranking #4 Best University in ...

Universitas Brawijaya - BITS

bits.ub.ac.id/pemberitahuan/ ▾ [Translate this page](#)

<http://ub.ac.id>. Domain Universitas Brawijaya telah pindah dari <http://brawijaya.ac.id> ke <http://ub.ac.id> · Beranda | Kebijakan & Aturan | Layanan A-Z | Berita BITS ...

Universitas Brawijaya - Wikipedia bahasa Indonesia ...

id.wikipedia.org/.../Universitas_... ▾ [Translate this page](#) Indonesian Wikipedia ▾

Universitas Brawijaya (biasa disingkat UNBRA, UNIBRAW atau singkatan resmi UB) merupakan lembaga pendidikan tinggi negeri di Indonesia yang berdiri ...

[Rektor Universitas Brawijaya - Templat:Universitas Brawijaya](#)

Universitas Brawijaya Malang | Facebook

<https://id-id.facebook.com/universitas.brawijaya> ▾ [Translate this page](#)

Universitas Brawijaya Malang ada di Facebook. Bergabunglah dengan Facebook untuk terhubung dengan Universitas Brawijaya Malang dan orang lain yang...



University of Brawijaya

[Directions](#)

University in Malang, Indonesia

Universitas Brawijaya, established in 1963 and located in Malang, is a state university in Indonesia. [Wikipedia](#)

Address: Jalan Veteran, Jawa Timur 65145, Indonesia**Phone:** +62 341 551611**Hours:** Open today · 7:00 am – 10:00 pm**Founded:** January 5, 1963[Get updates about University of Brawijaya](#)[Keep me updated](#)

Notable alumni



Munir Said



Jusuf Kalla



La Nyalla



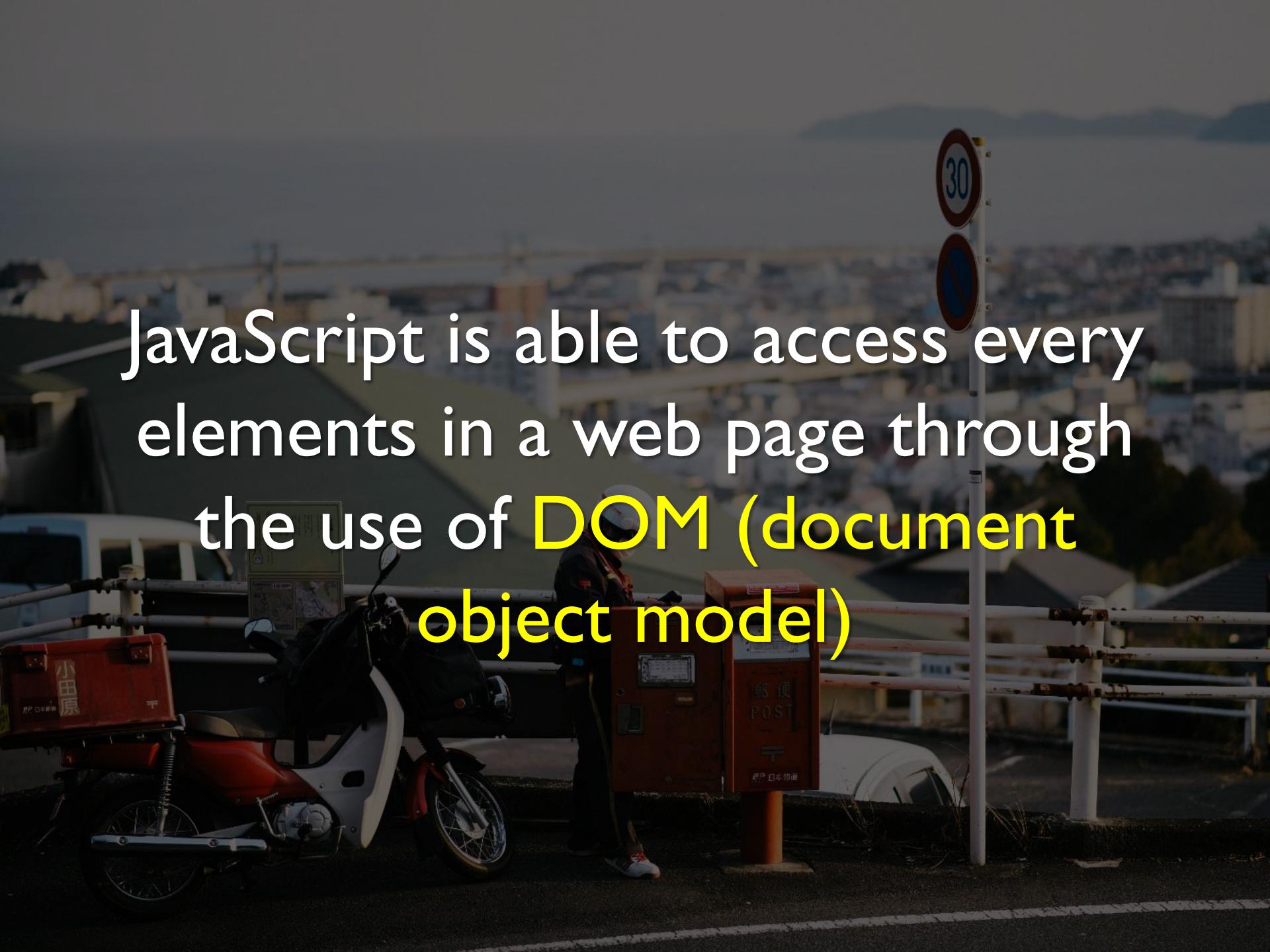
JavaScript can be added to a web page in two ways: internal and external

Internal

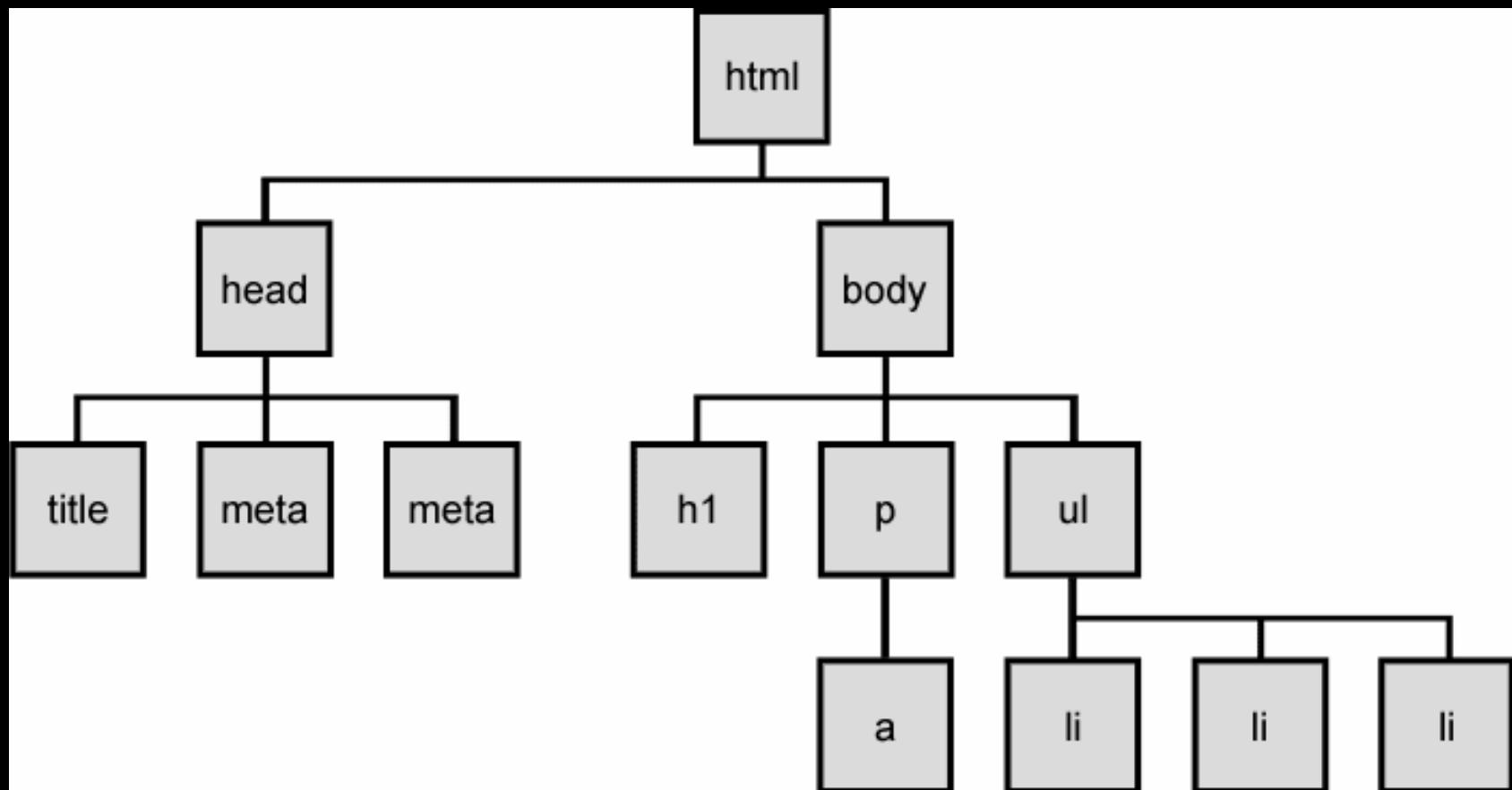
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <script>
      alert('Hello, world!');
    </script>
  </body>
</html>
```

External

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <script src="script.js"></script>
  </body>
</html>
```

A photograph of a person standing next to a red and white motorcycle. The motorcycle has "小田原" (Ogawara) written on its side. In the background, there is a red wooden post office box with "郵便" (Post) and "日本郵便" (Japan Post) written on it. A speed limit sign on a pole indicates "30".

JavaScript is able to access every elements in a web page through the use of **DOM** (document object model)



Available functions to access page/DOM elements:

- `document.getElementById()`
- `document.getElementsByTagName()`
- `document.getElementsByClassName()`
- `document.querySelector()`
- `document.querySelectorAll()`



JavaScript dan **wait** (listen) for an event to happen and **do** something when that action happens

Table 19-2. Common events

Event handler	Event description
onblur	An element loses focus
onchange	The content of a form field changes
onclick	The mouse clicks an object
onerror	An error occurs when the document or an image loads
onfocus	An element gets focus
onkeydown	A key on the keyboard is pressed
onkeypress	A key on the keyboard is pressed or held down
onkeyup	A key on the keyboard is released
onload	A page or an image is finished loading
onmousedown	A mouse button is pressed
onmousemove	The mouse is moved
onmouseout	The mouse is moved off an element
onmouseover	The mouse is moved over an element
onmouseup	A mouse button is released
onsubmit	The submit button is clicked in a form

selector.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Selector and Event</title>
  </head>
  <body>
    <label>
      Name: <input type="text" id="name">
    </label>
    <button id="greet">Greet</button>
    <script src="selector.js"></script>
  </body>
</html>
```

selector.js

```
const button = document.querySelector('#greet');

button.addEventListener('click', () => {
  const name = document.querySelector('#name').value;

  alert('Hello, ' + name);
});
```

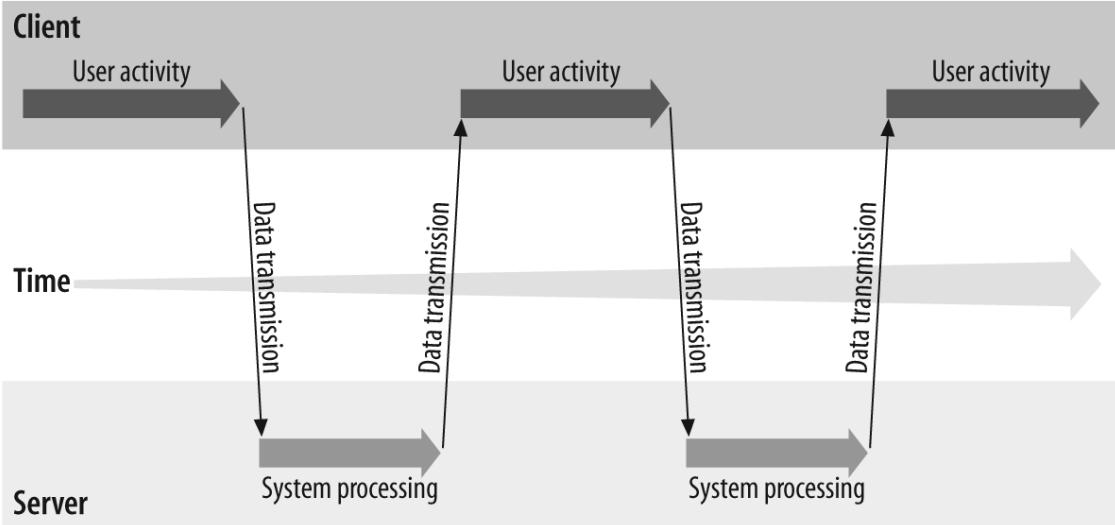


AjAX = Asynchronous JavaScript and XML

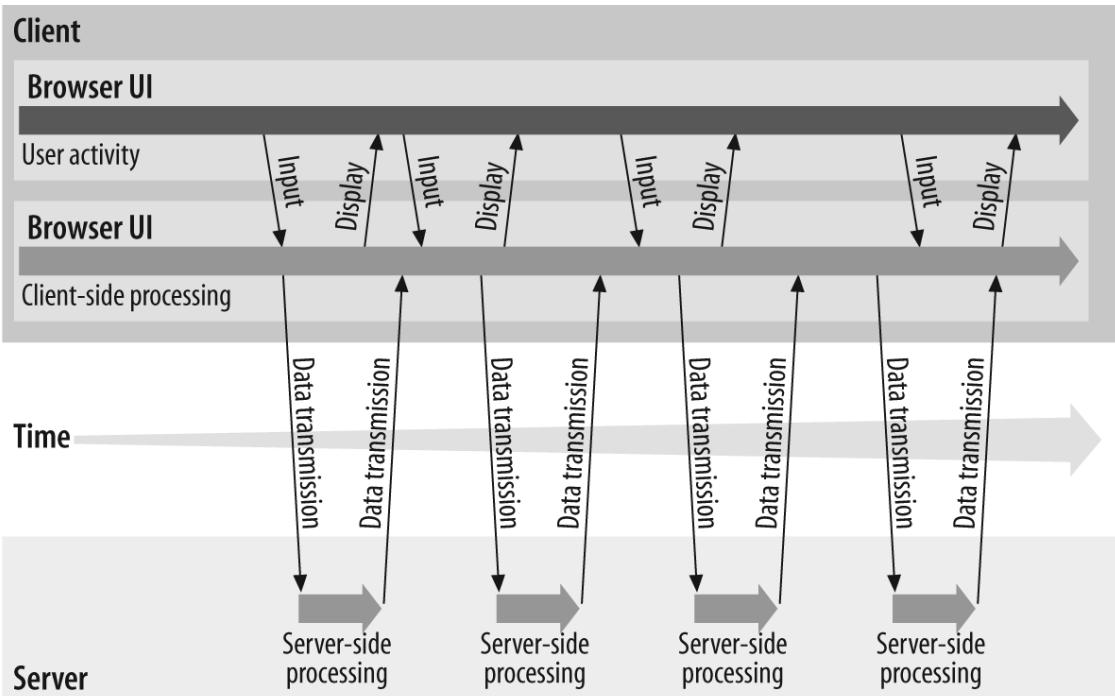
A photograph of a person wearing a white helmet and dark clothing standing next to a red and white motorcycle. The motorcycle has "小田原" (Ogawara) written on its side. They are positioned near a white metal fence. In the background, there's a road, some buildings, and a speed limit sign on a pole. The sky is overcast.

In a web page, **asynchronous** means the communication with server can happen without having to reload the page

Classic web application model (synchronous)



Ajax web application model (asynchronous)



htdocs/ajax/ajax.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>AJAX</title>
  </head>
  <body>
    <label>
      Name: <input type="text" id="name">
    </label>
    <button id="greet">Greet</button>
    <script src="ajax.js"></script>
  </body>
</html>
```

htdocs/ajax/ajax.js

```
const button = document.querySelector('#greet');

button.addEventListener('click', () => {
    const name = document.querySelector('#name').value;
    const url = 'http://localhost/ajax/ajax.php';

    fetch(url, {
        method: 'post',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `name=${name}`
    })
    .then(res => res.text())
    .then(res => {
        alert(res);
    });
});
```

htdocs/ajax/ajax.php

```
<?php  
sleep(1);  
$name = $_GET[‘name’];  
echo “Hello $name!”;
```



Let's make an **AJAX** chat app!

htdocs/chat/chat.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>AJAX Chat App</title>
  </head>
  <body>
    <textarea id="chat" rows="20" cols="31"></textarea><br>
    <label>
      Message: <input type="text" id="message">
    </label>
    <script src="chat.js"></script>
  </body>
</html>
```

htdocs/chat/chat.js

```
const chat = document.querySelector('#chat');
const message = document.querySelector('#message');
const baseUrl = 'http://localhost/chat';

function readChat() {
  fetch(` ${baseUrl}/chat-read.php`)
    .then(res => res.text())
    .then(res => {
      chat.value = res;
    });
  setTimeout(readChat, 1000);
}

readChat();
```

htdocs/chat/chat.js

```
message.addEventListener('keyup', e => {
  if (e.keyCode === 13) {
    fetch(` ${baseUrl}/chat-write.php`, {
      method: 'post',
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
      },
      body: `text=${message.value}`
    });
    message.value = '';
  }
});
```

htdocs/chat/chat-read.php

```
<?php
```

```
echo file_get_contents('chat.txt');
```

htdocs/chat/chat-write.php

```
<?php  
  
$message = $_GET['message'];  
  
file_put_contents('chat.txt', "$message\n", FILE_APPEND);
```



Create an empty file:
htdocs/chat/chat.txt

A photograph of a woman in a black suit waving from a train platform towards a man looking out from a train window. The train is silver with red stripes. The background shows railway tracks and greenery.

bye.