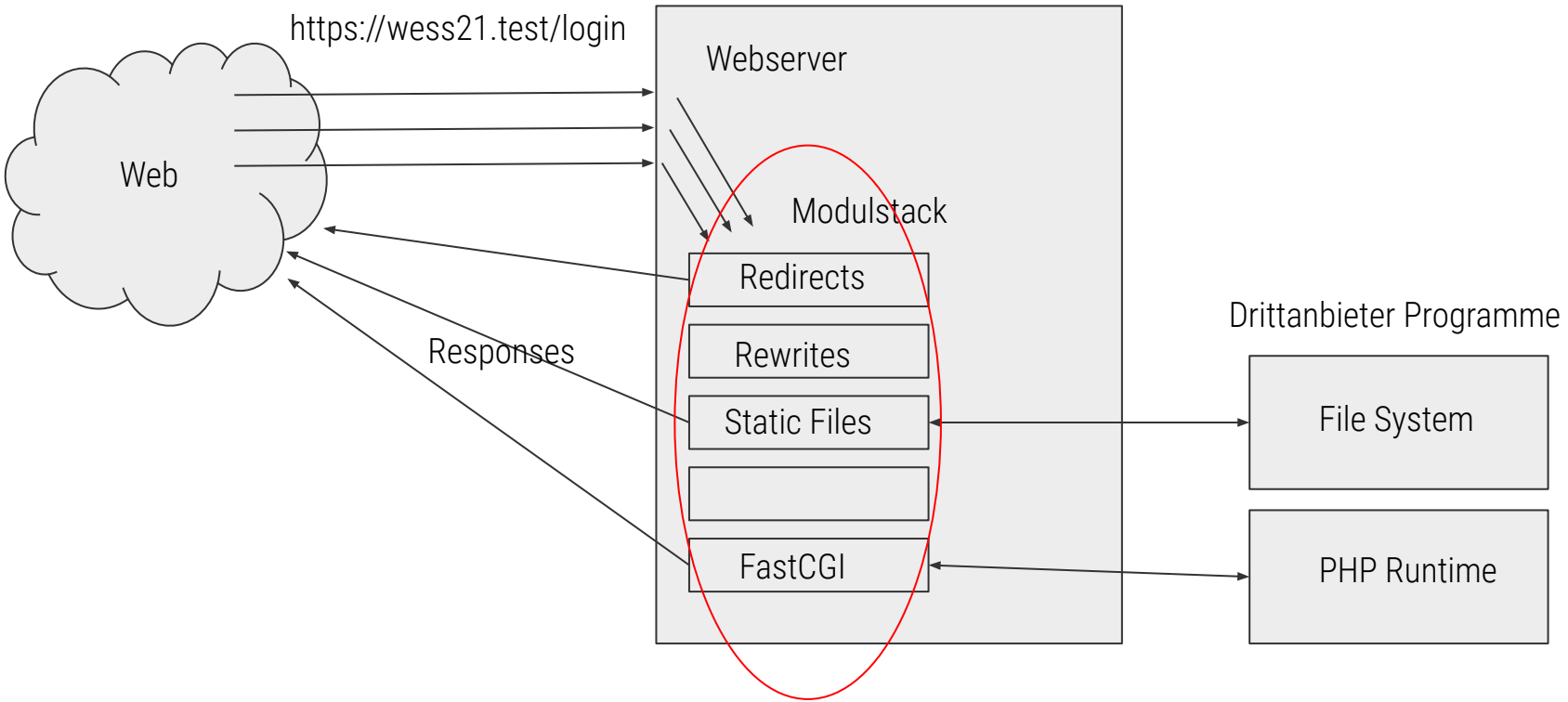
Praxiseinstieg

Webentwicklung und DevOps

- Kein Betrieb von Webanwendungen ohne “Operations”
  - Deployment, Infrastruktur und Administration gehört dazu
- Erfolgreiche Webanwendungen funktionieren nur im Ganzen
  - Z.B. keine gute Performance ohne CDN
  - Kein Continuous Delivery ohne gute Infrastruktur und Prozesse
- **DevOps** als Kombination von **Development** und **IT Operations**
  - Entwickler sollten grundlegende Administrationskenntnisse haben

=> Daher ist Infrastruktur und ihre Wartung immer Teil der Vorlesung

Zusammenspiel Technologien



Zusammenspiel - CGI

- PHP ist prinzipiell eine Skriptsprache
  - Wir wollen Architektur mit objektorientierter Programmierung
  - Skript als Einstieg!
    - **index.php**
- index.php liefert Einstieg in Anwendung
  - Kann von Webserver anhand der File Extension erkannt werden
- Skript meistens nicht Teil der URL
  - Rewrite fügt Skript in URL ein

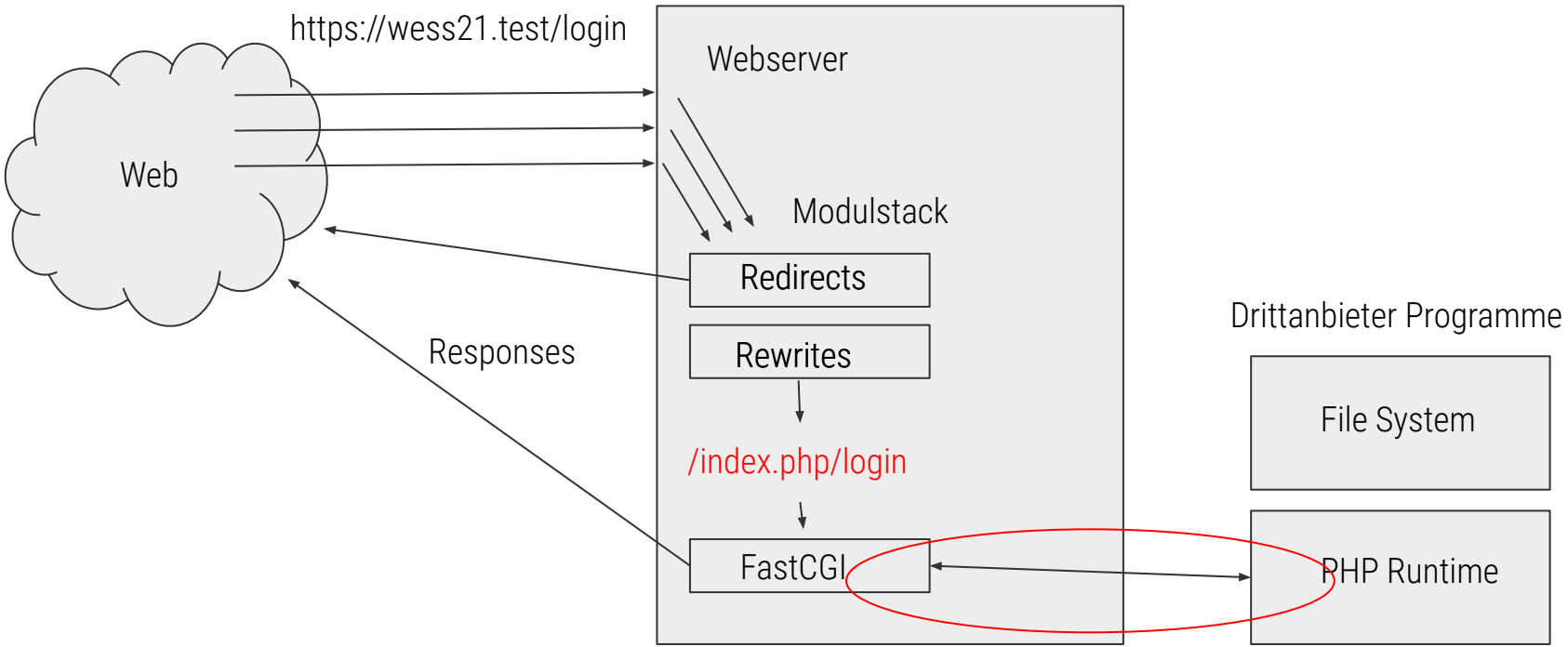
Zusammenspiel - Request zu Anwendung

```
http {
    index    index.html index.htm index.php;
    # ...

    server { # php/fastcgi
        listen       80;
        server_name  domain1.com www.domain1.com;
        root         html;
        try_files $uri $uri/ /index.php$uri$args;

        location ~ \.php$ {
            fastcgi_pass   127.0.0.1:1025;
        }
    }
}
```

Zusammenspiel Technologien



Zusammenspiel - CGI

- PHP Daemon (z.B. php-fpm) stellt Laufzeitumgebung
  - Führt Skript aus
- CGI (**C**ommon **G**ateway **I**nterface) Verbindung
- CGI bietet uns Ein- und Ausgabekanäle
- Eingabekanäle in PHP als magische Konstanten
  - $_SERVER für Aufrufkontext (auch Header)
  - $_REQUEST für Request-Parameter
    - Trennt sich in $_POST und $_GET
  - $_COOKIE für Cookie Daten (Authentifikation)
- Ausgabekanal über stdout
  - Rückgabe an Webserver ist Standard-Rückgabe
  - Z.B. einfach über echo

```
array (size=33)
  'USER' => string 'wickb' (length=5)
  'HOME' => string '/home/wickb' (length=11)
  'HTTP_TE' => string 'trailers' (length=8)
  'HTTP_UPGRADE_INSECURE_REQUESTS' => string '1' (length=1)
  'HTTP_ACCEPT_ENCODING' => string 'gzip, deflate, br' (lengt
  'HTTP_ACCEPT_LANGUAGE' => string 'de,en-US;q=0.7,en;q=0.3'
  'HTTP_ACCEPT' => string 'text/html,application/xhtml+xml,ap
  'HTTP_USER_AGENT' => string 'Mozilla/5.0 (X11; Ubuntu; Linu
  'HTTP_HOST' => string 'wess21.test' (length=11)
  'REDIRECT_STATUS' => string '200' (length=3)
  'SERVER_NAME' => string 'wess21.test' (length=11)
  'SERVER_PORT' => string '443' (length=3)
  'SERVER_ADDR' => string '127.0.0.1' (length=9)
  'REMOTE_PORT' => string '54704' (length=5)
  'REMOTE_ADDR' => string '127.0.0.1' (length=9)
  'SERVER_SOFTWARE' => string 'nginx/1.18.0' (length=12)
  'GATEWAY_INTERFACE' => string 'CGI/1.1' (length=7)
  'HTTPS' => string 'on' (length=2)
  'REQUEST_SCHEME' => string 'https' (length=5)
  'SERVER_PROTOCOL' => string 'HTTP/2.0' (length=8)
  'DOCUMENT_ROOT' => string '/home/wickb/Workspace/academy/we
  'DOCUMENT_URI' => string '/index.php' (length=10)
  'REQUEST_URI' => string '/login?foo=bar' (length=14)
  'SCRIPT_NAME' => string '/index.php' (length=10)
  'CONTENT_LENGTH' => string '' (length=0)
  'CONTENT_TYPE' => string '' (length=0)
  'REQUEST_METHOD' => string 'GET' (length=3)
  'QUERY_STRING' => string 'foo=bar' (length=7)
  'SCRIPT_FILENAME' => string '/home/wickb/Workspace/academy/
  'FCGI_ROLE' => string 'RESPONDER' (length=9)
  'PHP_SELF' => string '/index.php' (length=10)
  'REQUEST_TIME_FLOAT' => float 1616537066.6796
  'REQUEST_TIME' => int 1616537066
```

```
array (size=1)
  'foo' => string 'bar' (length=3)
```

PHP OOP

- index.php als Übergabepunkt für CGI und Bootstrapping unserer Anwendung
- Anwendungs- und Framework Logik wird objektorientiert aufgebaut
- Wichtige Einstiegspunkte sind:
  - Kernel
    - Übernimmt Bootstrapping von Anwendungskomponenten
    - Evtl. Request/Response-Fluss
  - Request/Response Handling
    - Kapselung von Informationen
    - Zentrales Ausgabemanagement
  - Routing
    - Weiterleitung in Business-Logik

Zusammenspiel - index.php

```php
<?php

use Symfony\Component\HttpFoundation\Request;

require dirname( path: __DIR__).'/vendor/autoload.php';

$kernel = new Kernel($_SERVER['APP_ENV'], (bool) $_SERVER['APP_DEBUG']);
$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
```

PHP OOP

- index.php soll einziges Skript sein
- Andere Dateien dürfen nicht über den Webserver erreichbar sein
  - Achtung, Sicherheitslücke!
- OOP Code in anderem Verzeichnis
- index.php liegt bei statischen Dateien

- **DocumentRoot** des Webserver nutzen

```
root /var/www/project/public;
```

Anwendungsdesign - Dynamisch oder statisch?

- Wir wollen Kontrolle über unser HTML
  - HTML wird ausschließlich durch PHP ausgeliefert
- **Dynamische auf Serverseite veränderte Inhalte brauchen PHP**
- Klassische Aufteilung:

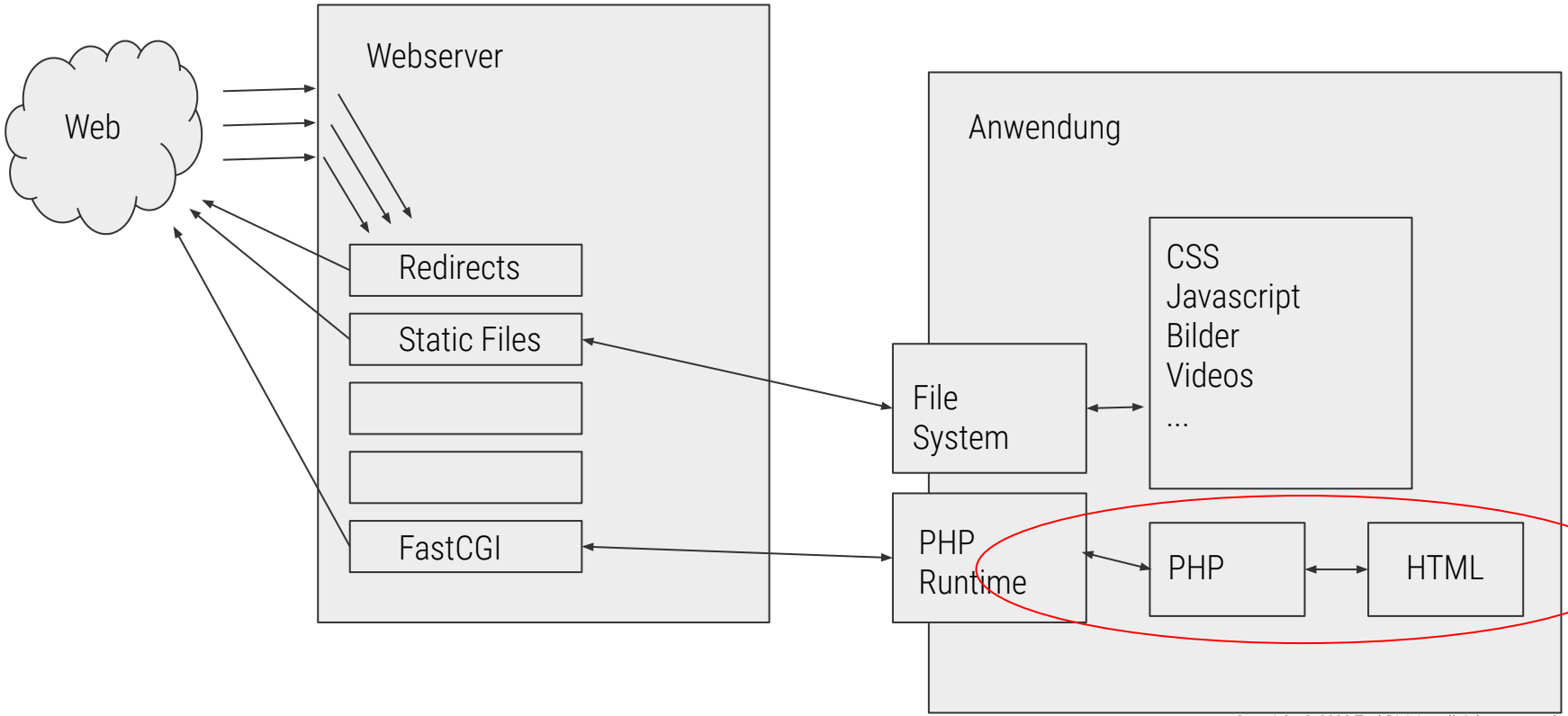
Dynamisch	Statisch
HTML	CSS
(PHP)	Javascript
	Medien (z.B. Bilder)

Anwendungsdesign - Dynamisch oder statisch?

- Andere Szenarien sind denkbar (aber nur selten sinnvoll)
  - Meist Umgehen von Caching zur Entwicklungszeit
    - CSS Preprocessing
    - Bildgrößen optimieren

- Faustregel: **Auf Performance für Enduser optimieren**
  - Enduser verbringen in Summe viel mehr Zeit mit Anwendung als jede andere Nutzergruppe (z.B. Entwickler)

Anwendungsdesign - Dynamisch oder statisch?



Zusammenspiel - PHP und HTML

- Wie können wir mit PHP HTML ausliefern?
  - PHP nur in HTML ist zu unflexibel

- Lösung muss:
  - Dynamisches Beeinflussen des HTML erlauben
  - Sollte klare Trennung der Sprachen erlauben
    - Erleichtert Wartbarkeit
    - Arbeitsteilung möglich
  - Wiederverwendbarkeit erleichtern

Zusammenspiel - PHP und HTML

- HTML direkt in PHP generieren

\+ Einfach zu verstehen
\+ Einfache Dynamisierung
\- Ab gewisser Menge schlecht wartbar
\- Keine Technologische Trennung

```php
<?php

$title = 'A fancy title';
$userName = 'Bernhard';

echo <<<EOT
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>$title</title>
</head>
<body>
    Hello $userName
</body>
</html>
EOT;
```

Zusammenspiel - PHP und HTML

- (P)HTML Inkludieren

+ Relativ saubere Tech-Trennung
- PHP Kenntnisse für Frontend-Entwickler
- Fehleranfällig durch PHP

```php
<?php

$title = 'A fancy title';
$userName = 'Bernhard';

include './phtml_inclusion.html';
```

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title><?php echo $title ?></title>
</head>
<body>
    Hello <?php echo $userName ?>
</body>
</html>
```

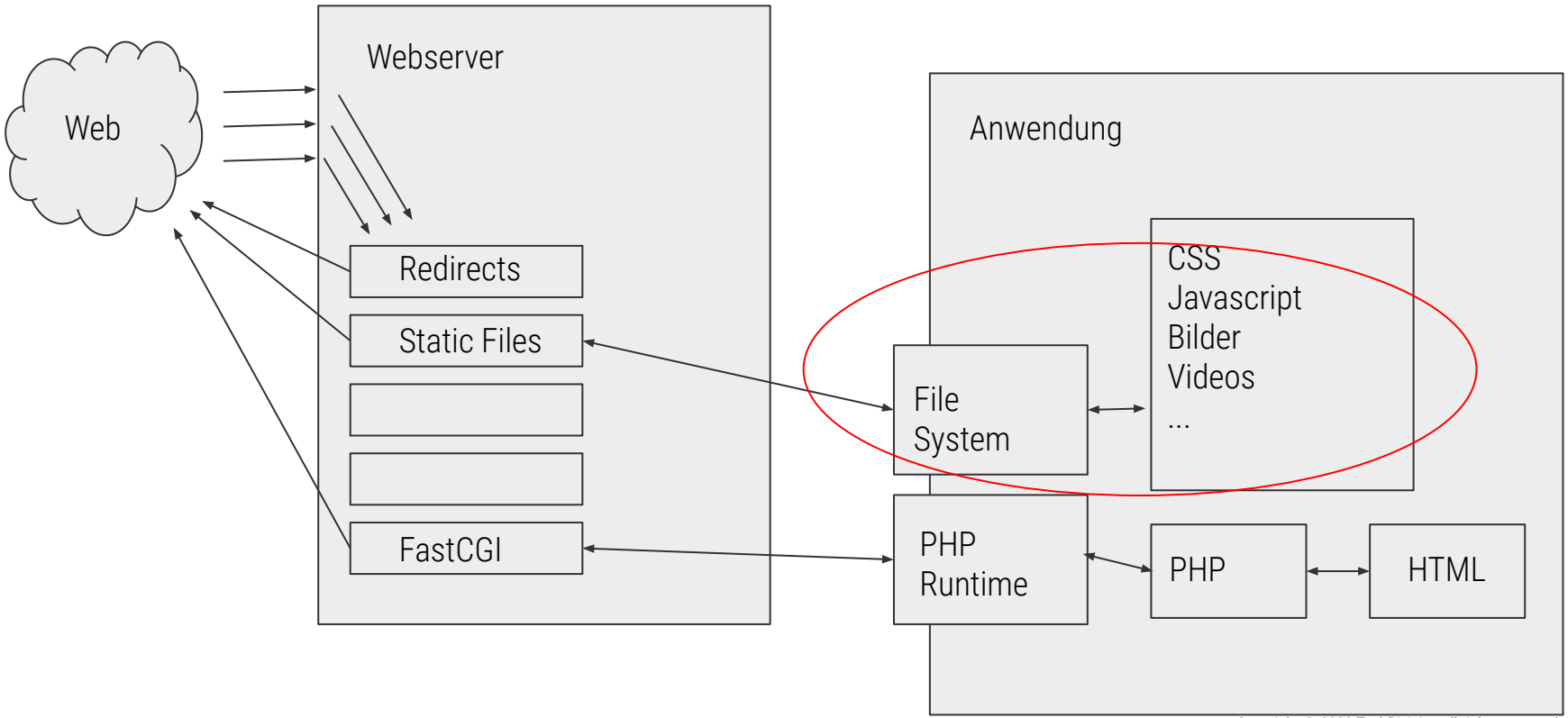
Zusammenspiel - PHP und HTML

- Template Compiling
  - Eigene Metasprache zur Dynamisierung
  - Mächtig aber stark fokussiert
  - Oft kompiliert zu PHP

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    {% for userName in users %}
        Hello {{ userName }}, you are user #{{ loop.index }}
    {% endfor %}
</body>
</html>
```

\+ Sehr saubere Tech-Trennung
\+ Wenig fehleranfällig
\+ Flexibel
\- Gelegentlich komplex in der Handhabung (Achtung Caches!)

Anwendungsdesign - Dynamisch oder statisch?



Zusammenspiel - HTML Referenzen

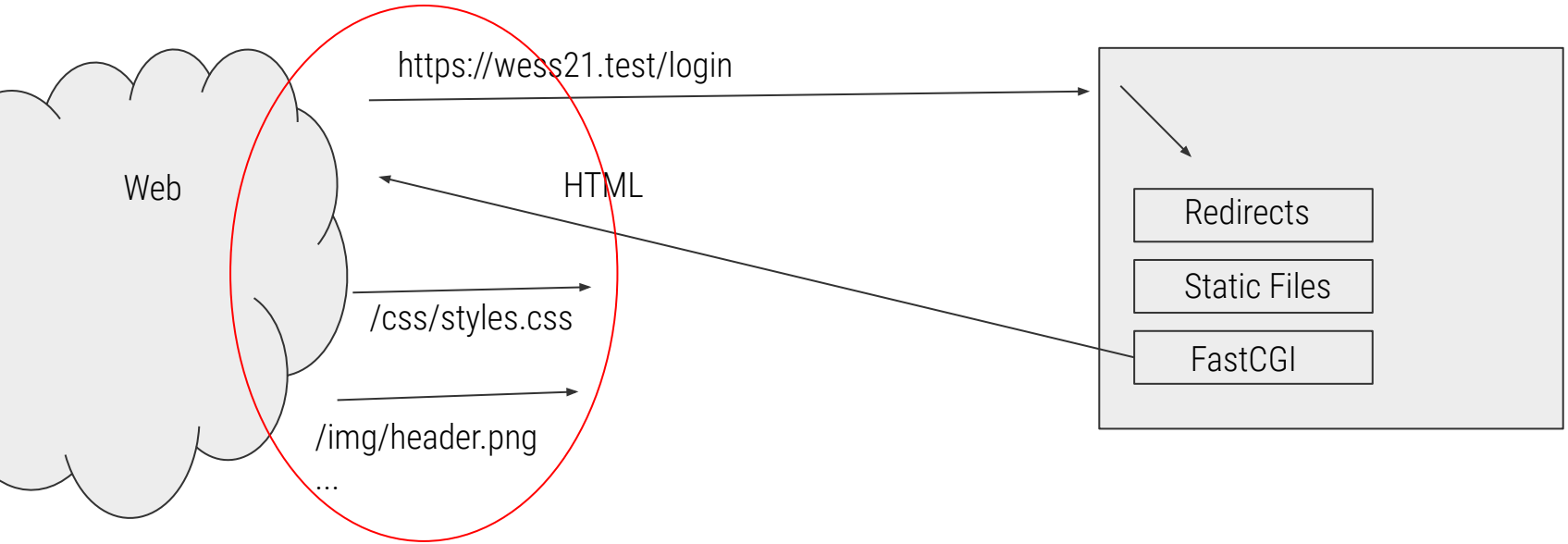
- Was haben wir?
  - index.php
    - liefert HTML
    - Einstieg in OOP Anwendungsentwicklung
  - Webserver der statische Dateien ausliefert
  - Statisches CSS, JS und evtl. Bilder

- Woher weiß der Browser welche statischen Ressourcen wir brauchen?
  - Wir nutzen sie in unserem HTML

Zusammenspiel - HTML Referenzen

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title><?php echo 'fancy title'; ?></title>
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link rel="stylesheet" type="text/css" href="theme.css">
    <script src="some_javascript.js" type="text/javascript" defer></script>
    <script src="some_more_javascript.js" type="text/javascript" defer></script>
</head>
<body>
    Look at this fancy picture
    <img src="smiley.gif" alt="Smiley face" height="42" width="42">
    Look at this fancy video
    <video width="320" height="240" controls>
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
        Your browser does not support the video tag.
    </video>
</body>
</html>
```

Anwendungsdesign - Dynamisch oder statisch?



Zusammenspiel - HTML Referenzen

- JS und CSS kann auch in HTML integriert werden

+ Keine zusätzlichen Requests
- Schlechte Tech-Trennung
- Schnell unübersichtlich
- Schlecht wartbar
- Schlecht anpassbar (CSS)

- Nur in Sonderfällen empfohlen!

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title><?php echo $title; ?></title>
    <style>
        body {
            color: red;
        }
    </style>
    <script>
        function createAlert (message) {
            alert(message);
        }
    </script>
</head>
<body>
    <script>
        createAlert('Achtung');
    </script>
</body>
</html>
```

Quellen:

- https://www.nginx.com/resources/wiki/start/topics/recipes/symfony/
- https://github.com/symfony/symfony
- https://symfony.com/doc/current/configuration.html
- https://de.wikipedia.org/wiki/FastCGI
- https://www.cs.usfca.edu/~parrt/papers/mvc.templates.pdf