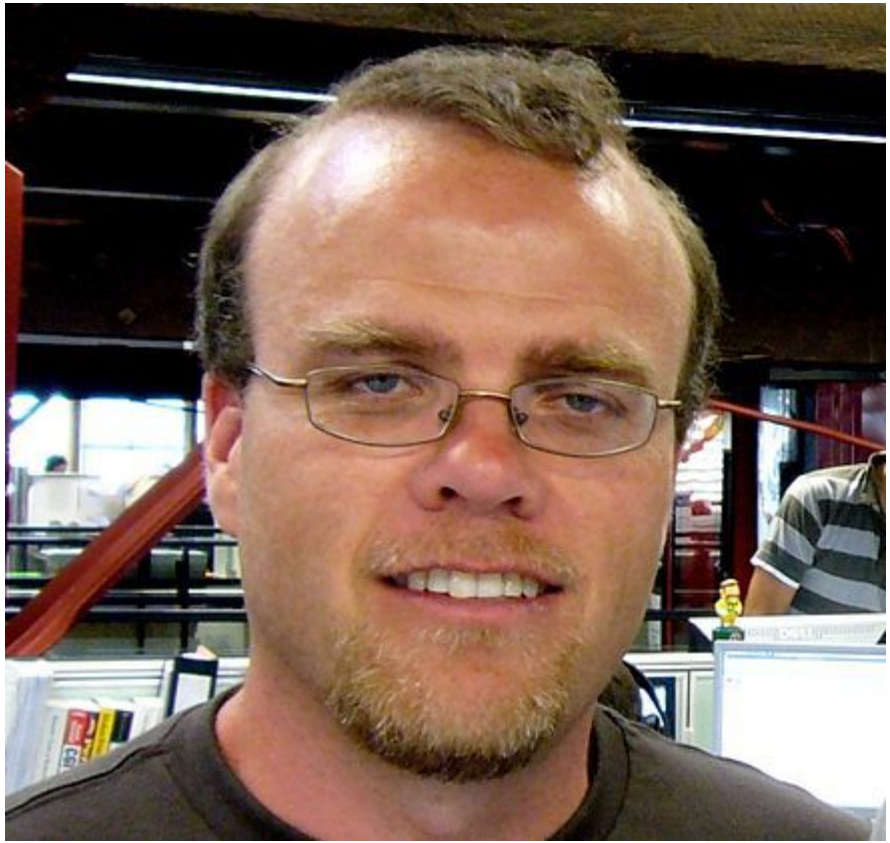
Server-side Scripting

Manfred Kaul

Ausschreibung Stud. Hilfskraftstelle (SHK)

- Korrekturhilfe SE1-Klausur im SoSe 2020
- SHK-Vertrag über 3 Monate von Juni bis August
- Klausur voraussichtlich Ende Juli 2020
- Bei Interesse bitte melden
- Voraussetzung: SE1-Klausur bereits bestanden



Rasmus Lerdorf *1968 Grönland

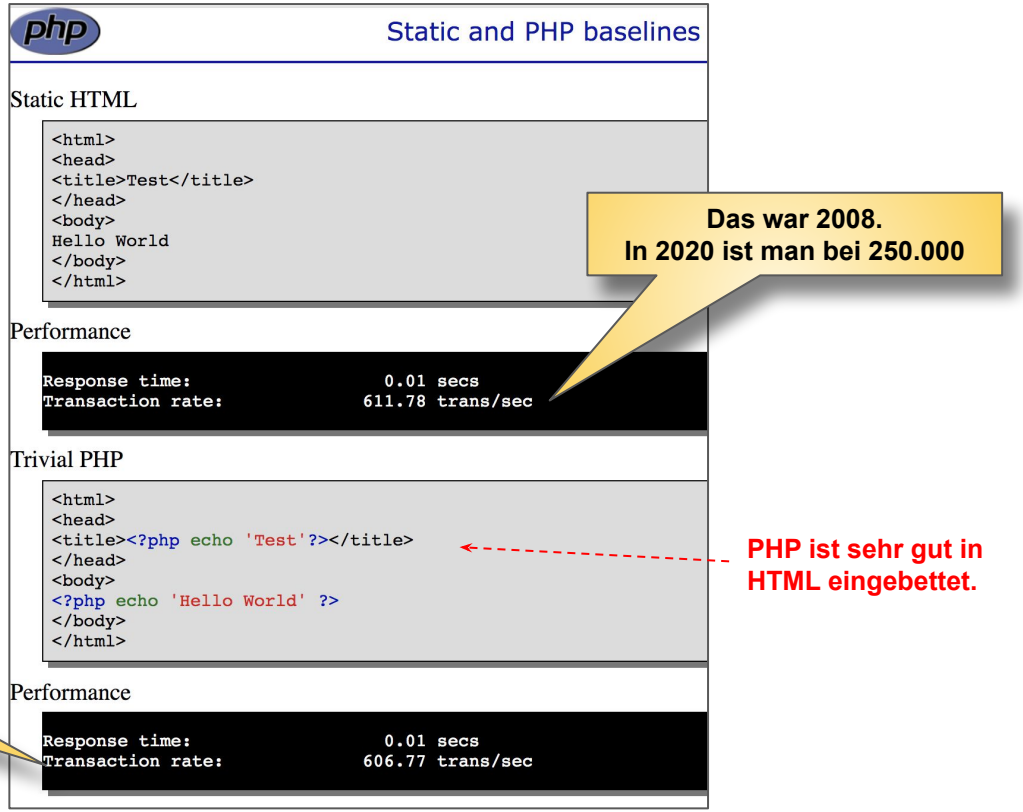
"I did not develop the PHP we know today. Dozens, if not hundreds of people, developed PHP. I was simply the first developer."

"I've never thought of PHP as more than a simple tool to solve problems."

"PHP is rarely the bottleneck."

https://en.wikiquote.org/wiki/Rasmus_Lerdorf

PHP is rarely the bottleneck.



0,8% Verlust durch PHP

http://talks.php.net/show/froscon08/24

Von NASA - NASA, Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=1324582

PHP Frameworks sind oft langsam

CakePHP 1.2.0rc2

Controller

```php
<?php
class HelloController extends AppController {
    var $layout = null;
    var $autoLayout = false;
    var $uses = array();
    var $helpers = array();
    function index()
    {
      $this->pageTitle = 'Test';
      $this->viewVars['body'] = 'Hello World';
    }
}
?>
```

View

```php
<html>
<head>
<title><?php echo $this->pageTitle?></title>
</head>
<body>
<?php echo $this->viewVars['body'] ?>
</body>
</html>
```

Performance

```
Response time:                  0.19 secs
Transaction rate:              25.88 trans/sec
```

95,8% Verlust durch CakePHP

Drupal 6.4

Drupal Module

```php
<?php
function hello_menu() {
  $items = array();

  $items['hello'] = array(
    'title' => 'Hello world page...',
    'page callback' => 'hello_page',
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK,
  );
  return $items;
}

function hello_page() {
  return '<p>Hello world</p>';
}
?>
```

Performance

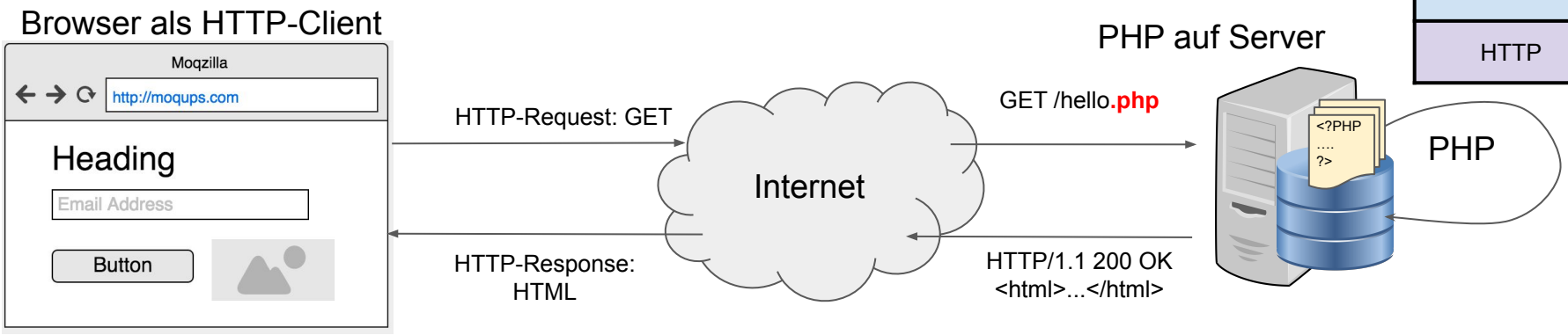
```
Response time:                  0.10 secs
Transaction rate:              51.37 trans/sec
```

91,6% Verlust durch Drupal

http://talks.php.net/show/froscon08/32

PHP ist eine Server-Skript-Sprache

PHP-Skripte werden an der Endung ***.php** erkannt.



OSI Layer	TCP/IP Protocol Stack
Application	HTTP
Transport	TCP
Network	IP
Physical	Ethernet, WLAN

Klartext

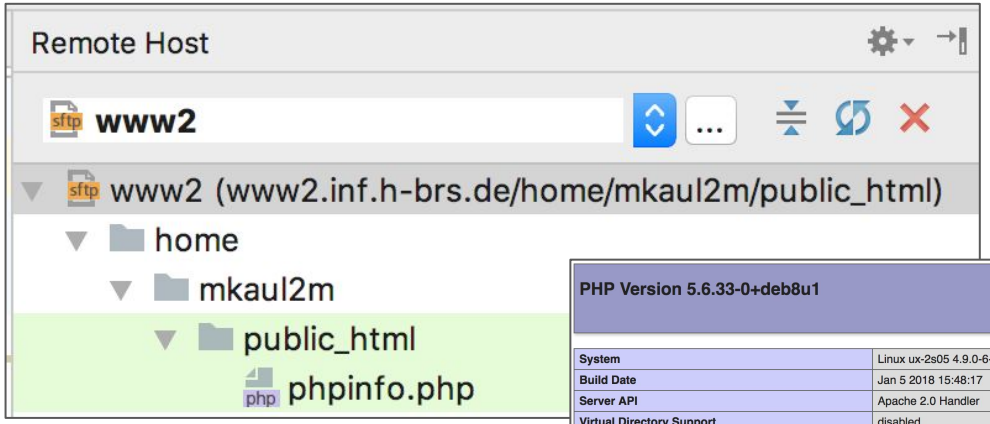
```
GET /hello.php HTTP/1.1
Host: www2.inf.h-brs.de
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

The server could answer:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Language: en
Content-Location: http://www2.inf.h-brs.de/hello.php
```

Quelle: https://www.w3.org/TR/cooluris/

Welche PHP-Version läuft auf meinem Server?




```php
<?PHP
   phpinfo();
?>
```

PHP Version 5.6.33-0+deb8u1

System	Linux ux-2s05 4.9.0-6-amd64 #1 SMP Debian 4.9.88-1 (2018-04-29) x86_64
Build Date	Jan 5 2018 15:48:17
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

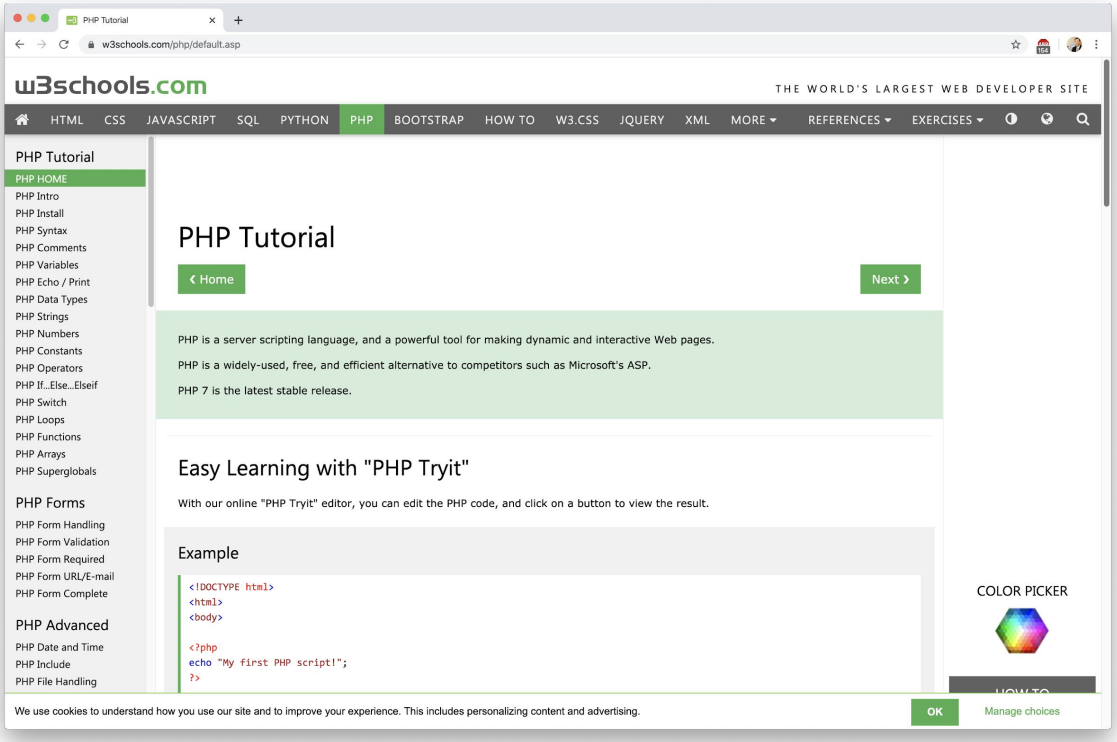
Wie lernt man PHP? http://de2.php.net/manual/de/

Einführung
- Einführung
- Ein kleines Tutorial

Sprachreferenz
- Grundlagen der Syntax
- Typen
- Variablen
- Konstanten
- Ausdrücke
- Operatoren
- Kontrollstrukturen
- Funktionen
- Klassen und Objekte
- Namespaces
- Fehler
- Ausnahmebehandlung (Exception-Handling)

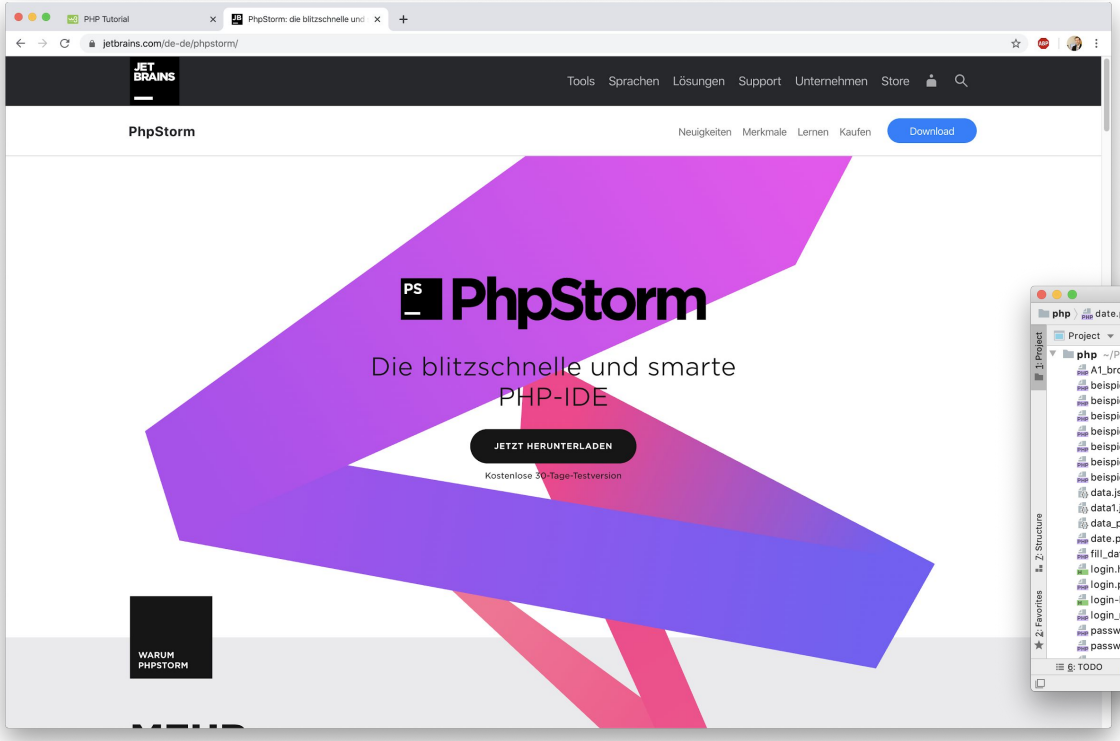
Vordefinierte Variablen
- Superglobals — Superglobals sind Built-in-Variablen, die immer in allen Gültigkeitsbereichen verfügbar sind
- $GLOBALS — Referenziert alle Variablen, die im globalen Gültigkeitsbereich vorhanden sind
- $_SERVER — Informationen über Server und Ausführungsumgebung
- $_GET — HTTP GET-Variablen
- $_POST — HTTP POST-Variablen
- $_FILES — HTTP Dateiupload-Variablen
- $_REQUEST — HTTP Request-Variablen
- $_SESSION — Sessionvariablen
- $_ENV — Umgebungsvariablen
- $_COOKIE — HTTP Cookies

W3Schools PHP Tutorial mit Tryit - Editor

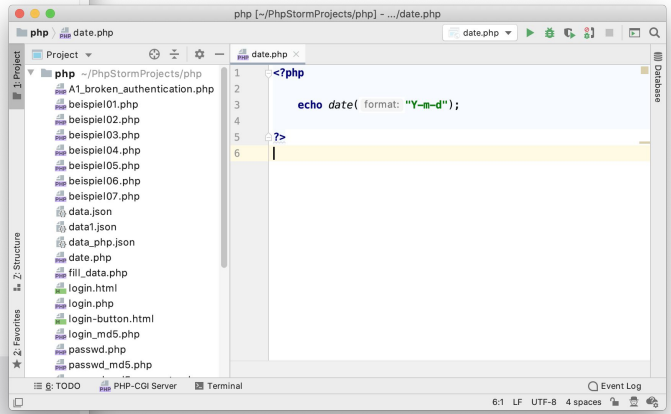


https://www.w3schools.com/php/default.asp

PHP-IDE PhpStorm



Free Student License available



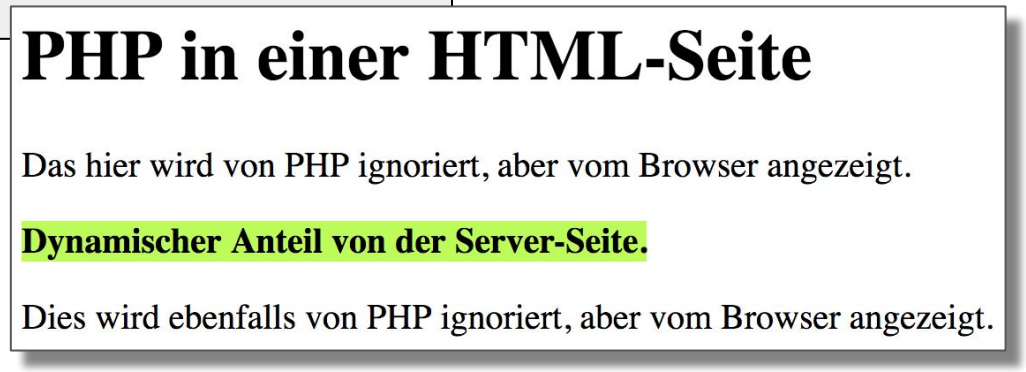
https://www.jetbrains.com/de-de/phpstorm/

PHP-Skripte mit <?PHP ... ?> in HTML eingebettet

```
<!doctype html>
<h1>PHP in einer HTML-Seite </h1>
<p>Das hier wird von PHP ignoriert, aber vom Browser angezeigt.</p>

<div>
<?PHP
   echo('<b style="background-color: greenyellow">
                  Dynamischer Anteil von der Server-Seite.</b>');
?>
</div>

<p>Dies wird ebenfalls von PHP ignoriert, aber vom Browser angezeigt.</p>
```

Server-Skript

PHP als HTML-Template Engine



Mischung Client- und Server-seitiger Berechnung

```
<!doctype html>
<style>
  div {background-color: greenyellow;}
</style>
<h1>PHP in einer HTML-Seite </h1>
<p>Das hier wird von PHP ignoriert, aber vom Browser angezeigt.
</p>

<div>
<?PHP
  echo 'Dynamischer Anteil: 2 * 3 = ' + (2 * 3);
  echo " wird auf der Server-Seite berechnet."
?>
</div>

<p>Dies wird ebenfalls von PHP ignoriert, aber vom Browser angezeigt.</p>

<script>
  const span = document.createElement('span');
  span.innerText = 2 * 3;
  document.querySelector('h1').appendChild(span);
</script>
```

PHP in einer HTML-Seite 6

Das hier wird von PHP ignoriert, aber vom Browser angezeigt.

6 wird auf der Server-Seite berechnet.

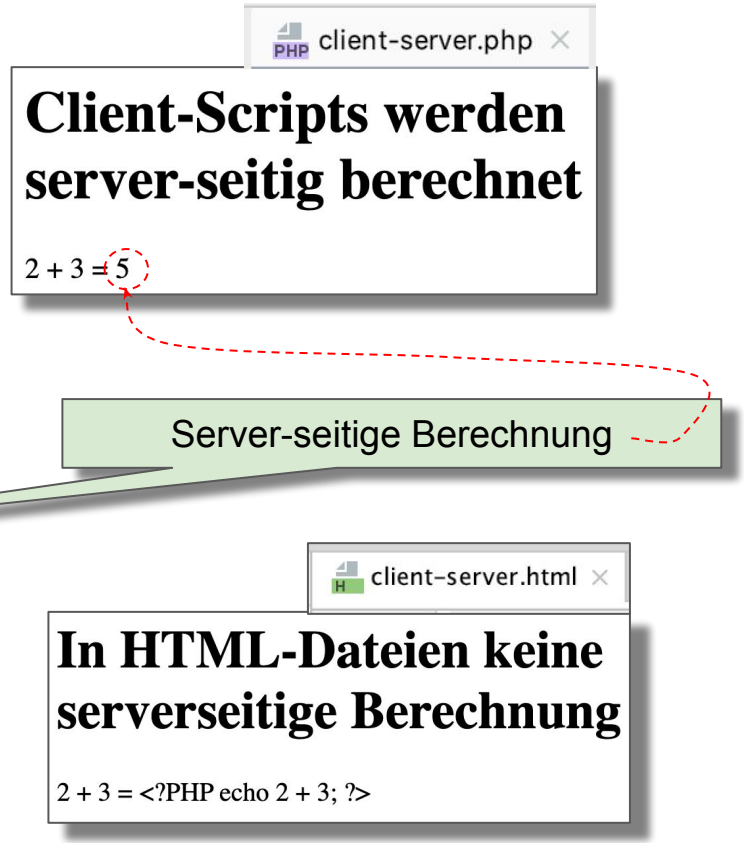
Dies wird ebenfalls von PHP ignoriert, aber vom Browser angezeigt.

Server-seitige Berechnung

Client-seitige Berechnung

Client-Scripts können auch server-seitig berechnet werden

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Client-Server</title>
</head>
<body>
<h1>Client-Scripts werden server-seitig berechnet</h1>
<p>2 + 3 = <span id="result"></span></p>
<script>
  const span = document.getElementById('result');
  span.textContent = "<?PHP echo 2 + 3; ?>";
</script>
</body>
</html>
```



Variablen in PHP beginnen mit $-Zeichen

```php
<!doctype html>
<?PHP
  $a = 2;
  $b = 3;

  function sum( $x, $y ){
     return $x * $y;
  }

  $faktor = 10;
  function sum2( $x, $y ){
     global $faktor;
     return $faktor * ( $x * $y );
  }
?>
```

globaler Geltungsbereich

lokaler Geltungsbereich

$faktor wäre in der Funktion nicht sichtbar

globale Variablen müssen importiert werden

Konstanten in PHP

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

https://www.w3schools.com/php/php_constants.asp

http://de2.php.net/manual/de/language.variables.basics.php

Werte von PHP nach JavaScript kopieren

```
<!doctype html>
<?PHP
  $a = 2;
  $b = 3;

  function sum( $x, $y ){
     return $x * $y;
  }
?>
<script>
  const sum = <?PHP echo sum( $a, $b ) ?>;
  console.log( sum );
</script>
```

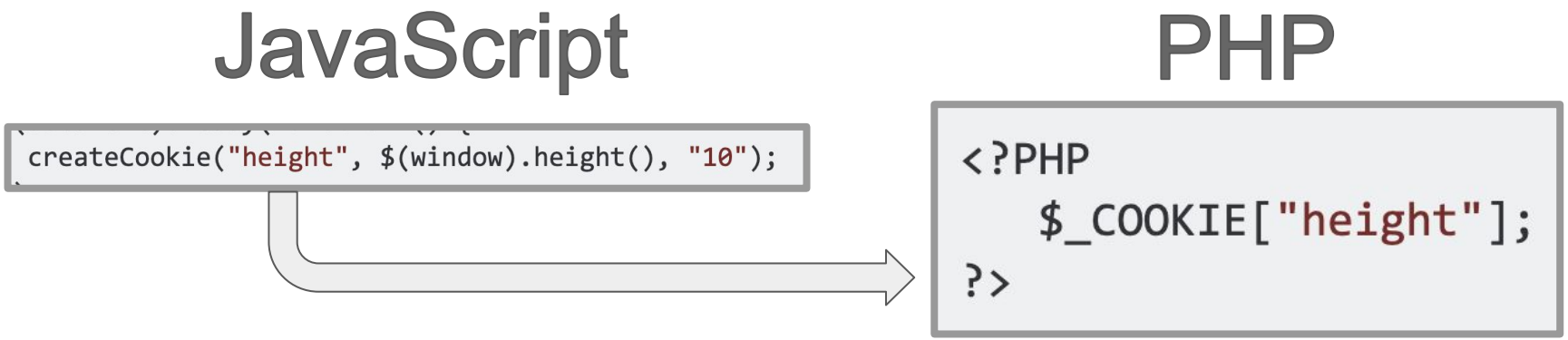
PHP als JavaScript-Template Engine

http://de2.php.net/manual/de/language.variables.basics.php

Werte von JavaScript nach PHP kopieren?

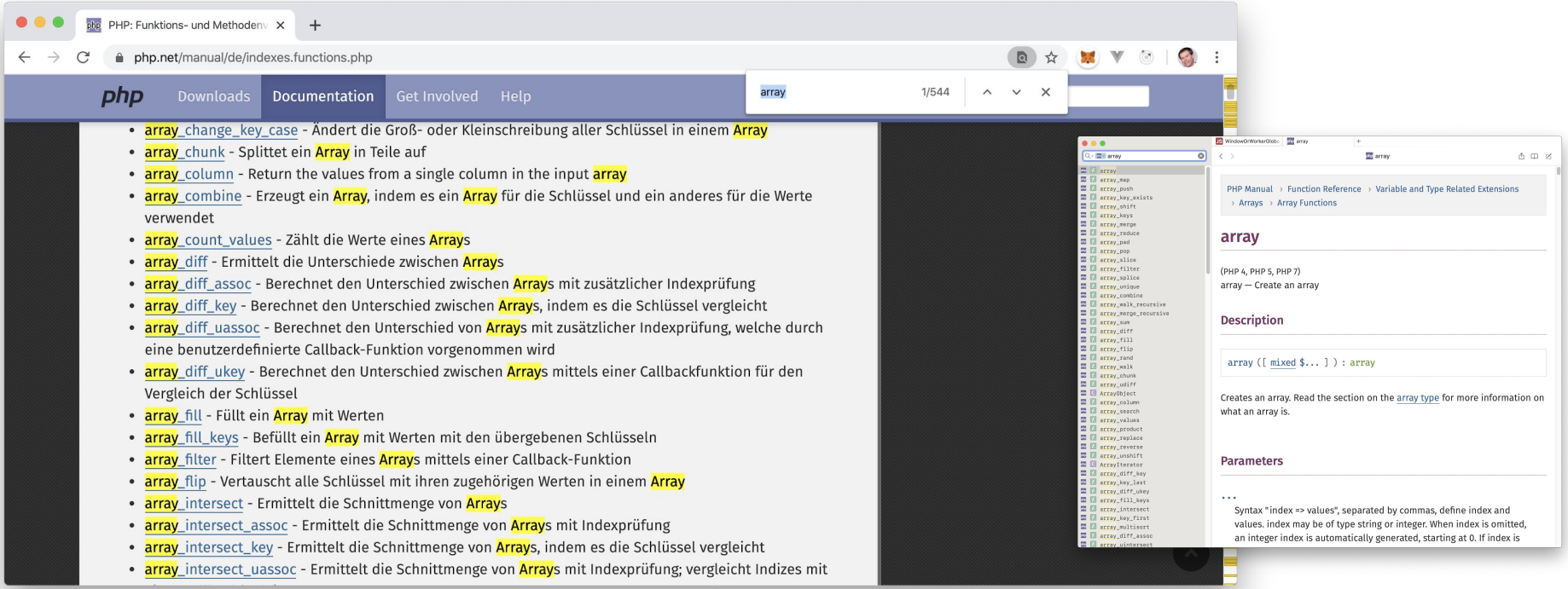
3 Wege, um Werte vom Client zum Server zu transferieren:

1. Cookie
2. Form
3. Ajax oder XMLHttpRequest oder fetch



https://stackoverflow.com/questions/1917576/how-do-i-pass-javascript-variables-to-php

PHP hat einen großen flachen Funktionsraum



https://www.php.net/manual/de/indexes.functions.php

Arrays -1-

```php
<!doctype html>
<?php

  $array = array(
     "foo" => "bar",
     "bar" => "foo",
  );
  echo "<h1>Array 1</h1>";
  var_export($array);

  // seit PHP 5.4
  $array = [
     "foo" => "bar",
     "bar" => "foo",
  ];

  echo "<h1>Array 2</h1>";
  var_export($array);

  $array = array("foo", "bar", "hello", "world");

  echo "<h1>Array 3</h1>";
  var_export($array);
```

Array 1

array ('foo' => 'bar', 'bar' => 'foo',)

Array 2

array ('foo' => 'bar', 'bar' => 'foo',)

Array 3

array (0 => 'foo', 1 => 'bar', 2 => 'hello', 3 => 'world',)

http://de2.php.net/manual/de/language.types.array.php
https://secure.php.net/manual/de/function.var-export.php

Arrays -2-

```php
$array = array(
    "foo" => "bar",
    42 => 24,
    "multi" => array(
        "dimensional" => array(
            "array" => "foo"
        )
    )
);

echo "<h1>Array 4</h1>";
var_export($array);
echo "<h1>Element 5</h1>";
var_export($array["foo"]);
echo "<h1>Element 6</h1>";
var_export($array[42]);
echo "<h1>Element 7</h1>";

var_export($array["multi"]["dimensional"]["array"]);

?>
```

Array 4

array ('foo' => 'bar', 42 => 24, 'multi' => array ('dimensional' => array ('array' => 'foo',),),)

Element 5

'bar'

Element 6

24

Element 7

'foo'

http://de2.php.net/manual/de/language.types.array.php
https://secure.php.net/manual/de/function.var-export.php

Array-Iteration

```php
<?PHP

  $array = [ 0,1,2,3,4,5,6,7,8,9 ];

  for ($i = 0; $i < 10; $i++) {
     echo $array[ $i ];
  }

 $sum = 0;
 foreach ( $array as $key => $value ){
     $sum += $value;
  }

  print_r( $array );
  var_export( $array );
  echo $sum;

?>
```

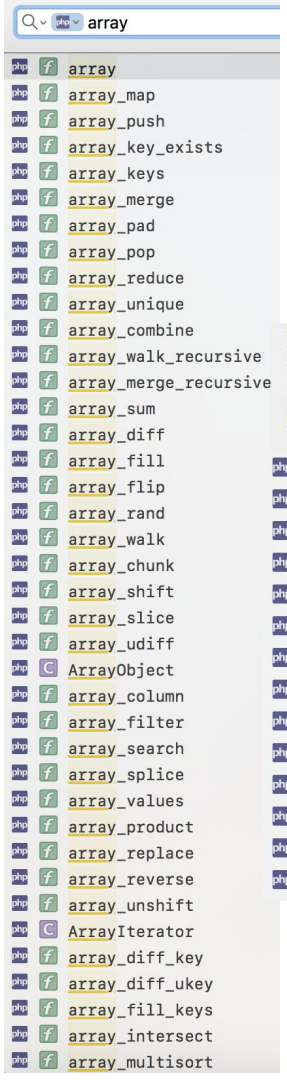
http://de2.php.net/manual/de/language.types.array.php

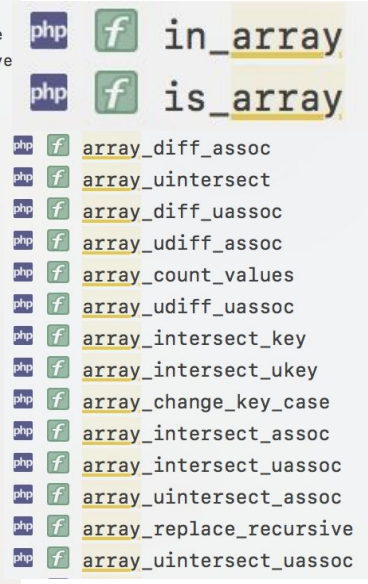
Funktionales Programmieren in PHP

```php
<!doctype html>
<?PHP

  function even( $param ){
     return $param % 2 === 0;
  }

  function square( $param ){
     return $param * $param;
  }

  function sum( $x, $y ){
     return $x + $y;
  }

  $array = [ 0,1,2,3,4,5,6,7,8,9 ];
  $even  = array_filter( $array, 'even' );
  $squares = array_map( 'square', $even);
  $sum = array_reduce( $squares, 'sum', 0);
```

```php
  print_r( $array );
  var_dump( $even );
  var_export( $squares );
  echo $sum;

?>
```

filter
map
reduce





http://de2.php.net/manual/de/ref.array.php

Ausgabe-Parameter mit & markieren

Call by reference statt Call by value

```php
<?php

  function foo( &$var ){
     $var++;
  }

  $a = 5;
  foo($a); // $a is 6 here

?>
```

https://secure.php.net/manual/en/language.references.pass.php

JSON

PHP als Daten-Lieferant

```
<!doctype html>
<?PHP

  $json = [
     "name" => "Kaul",
     "vorname" => "Manfred",
     "hochschule" => "H-BRS"
  ];

?>

<script>
  let json = <?PHP echo json_encode( $json ) ?>;
  console.log( JSON.stringify( json, null, 2 ) );
</script>
```

JavaScript als Empfänger

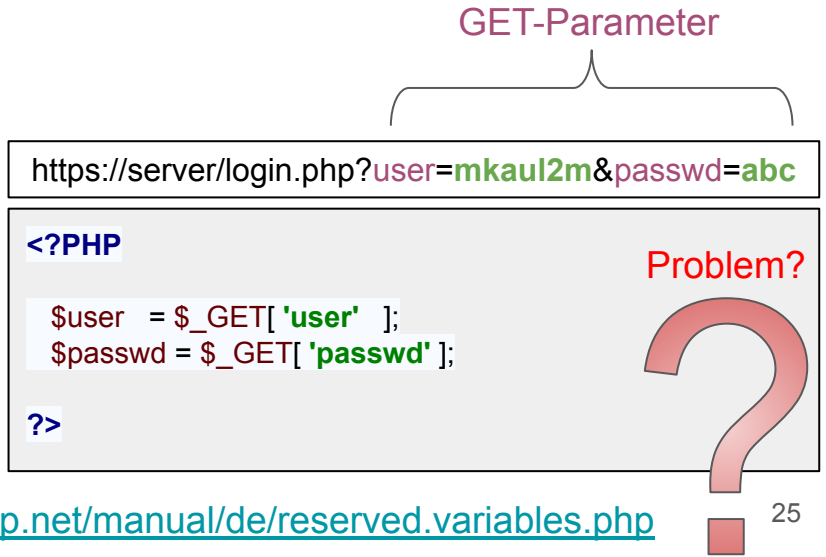
JSON

```
{
 "name": "Kaul",
 "vorname": "Manfred",
 "hochschule": "H-BRS"
}
```

http://de2.php.net/manual/de/function.json-encode.php

Vordefinierte Variablen in PHP

- $_SERVER — Informationen über Server und Ausführungsumgebung
- $_GET — HTTP GET-Variablen
- $_POST — HTTP POST-Variablen
- $_FILES — HTTP Dateiupload-Variablen
- $_REQUEST — HTTP Request-Variablen
- $_SESSION — Sessionvariablen
- $_ENV — Umgebungsvariablen
- $_COOKIE — HTTP Cookies



http://de2.php.net/manual/de/reserved.variables.php

Eingabe filtern mit filter_input

statt $user = $_GET['user'];

```
$user = filter_input( INPUT_GET, 'user', FILTER_SANITIZE_STRING );
```

- **INPUT_GET**, **INPUT_POST**, **INPUT_COOKIE**, **INPUT_SERVER** oder **INPUT_ENV**.
- Filters
  - FILTER_SANITIZE_EMAIL
  - FILTER_SANITIZE_NUMBER_FLOAT
  - FILTER_SANITIZE_NUMBER_INT
  - FILTER_SANITIZE_SPECIAL_CHARS
  - FILTER_SANITIZE_STRING
  - FILTER_SANITIZE_URL
- http://de2.php.net/manual/de/function.filter-input.php
- http://de2.php.net/manual/de/filter.filters.php
- http://de2.php.net/manual/de/filter.filters.sanitize.php

require und include

- Dateien werden unter dem angegebenen Pfad gesucht, oder, wenn keiner gegeben ist, im include_path.
- Wenn die Datei auch im include_path nicht gefunden werden kann, sucht *include* noch in dem Verzeichnis der aufrufenden Datei und dem aktuellen Arbeitsverzeichnis.
- Wenn keine Datei gefunden wurde, erzeugt *include* eine Warnung.
- Im Gegensatz dazu erzeugt require in diesem Fall einen Fatal Error.

php › php passwd.php ›

```php
<?PHP

  $My_Secrets = [
     'mkaul2m'   => 'abc',
     'sstudi2s'  => 'etc',
     'amuster2s' => 'xyz'
  ];

?>
```

php › php login.php ›

```php
<?PHP

  include "./passwd.php";
  if ( $My_Secrets } ...

?>
```

Login mit PHP

php › php passwd.php

```php
<?PHP

  $My_Secrets = [
      'mkaul2m'   => 'abc',
      'sstudi2s'  => 'etc',
      'amuster2s' => 'xyz'
  ];

?>
```

php › php login.php

```php
<?PHP

  $user   = $_GET[ 'user'   ];
  $passwd = $_GET[ 'passwd' ];

  include "./passwd.php";

  if (  array_key_exists( $user, $My_Secrets ) ){
      if ( $My_Secrets[ $user ] === $passwd ){
          echo "Login erfolgreich!";
      }
  }

?>
```

Problem?

Geheime Dateien aus public_html entfernen

php › passwd.php

```php
<?PHP

  $My_Secrets = [
     'mkaul2m'   => 'abc',
     'sstudi2s'  => 'etc',
     'amuster2s' => 'xyz'
  ];

?>
```

php › login.php

```php
<?PHP

  $user   = $_GET[ 'user'   ];
  $passwd = $_GET[ 'passwd' ];

  include "/etc/passwd.php";

  if (  array_key_exists( $user, $My_Secrets ) ){
     if ( $My_Secrets[ $user ] === $passwd ){
        echo "Login erfolgreich!";
     }
  }

?>
```

Problem?

Message-Digest Algorithm 5 (*md5*)

- Message-Digest Algorithm 5 (MD5) ist eine weit verbreitete kryptographische Hashfunktion,
- berechnet aus einem String einen 128-Bit-Hashwert
- Dies erlaubt beispielsweise die leichte Überprüfung von Downloads
- MD5 wurde 1991 von Ronald L. Rivest entwickelt.

Ronald L. Rivest

https://de.wikipedia.org/wiki/Message-Digest_Algorithm_5

Passwörter verschlüsseln mit *md5*()

php › passwd_md5.php

```php
<?PHP

  $MD5_Secrets = [
     'af9d9223e06b7023e2f1d475a8382ad1'  => '900150983cd24fb0d6963f7d28e17f72',
     '072f040463fdd000ddb70bbc355c0824'  => 'e80f17310109447772dca82b45ef35a5',
     '95b8c3d9dd0fabc7cf392a1b81889486'  => 'd16fb36f0911f878998c136191af705e'
  ];

?>
```

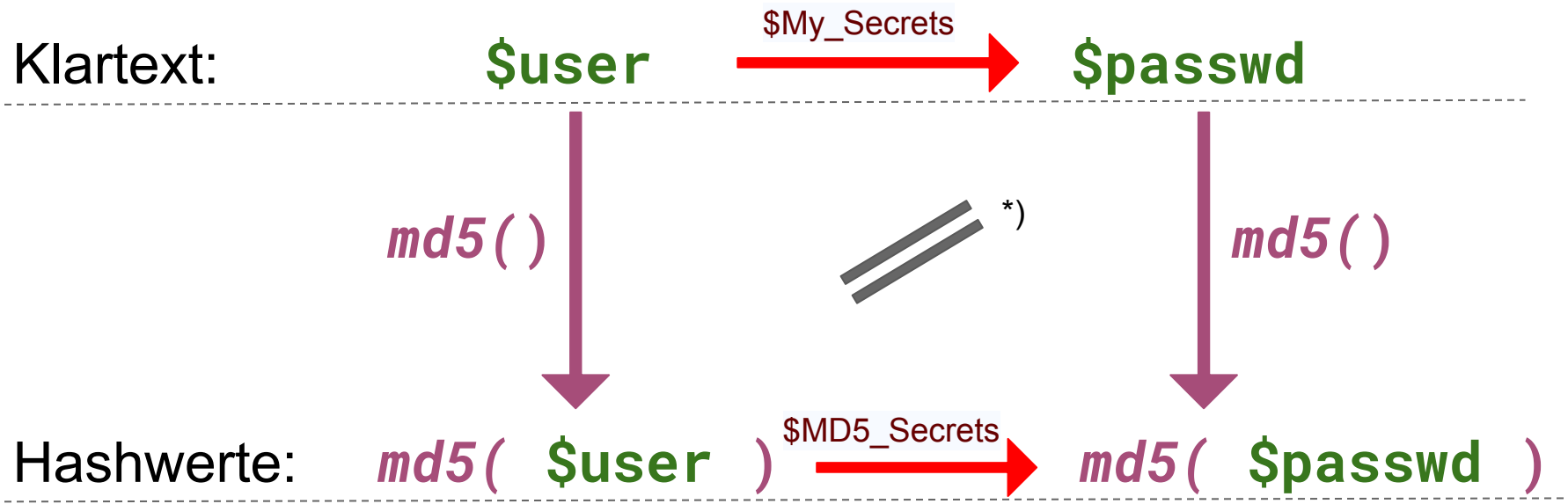
php › login.php

```php
<?PHP

  $user   = $_GET[ 'user'   ];
  $passwd = $_GET[ 'passwd' ];

  include "./passwd_md5.php";

  if (  array_key_exists( md5( $user ), $MD5_Secrets ) ){
     if ( $My_Secrets[ md5( $user ) ] === md5( $passwd ) ){
        echo "Login erfolgreich!";
     }
  }

?>
```

string md5 (string $str)
berechnet den MD5-Hash von **$str** unter Verwendung des » RSA Data Security, Inc. MD5 Message-Digest Algorithm und gibt das Ergebnis zurück.

http://de2.php.net/manual/de/function.md5.php

Kommutierendes Diagramm mit *md5*()



*) Die 2 Balken bedeuten: Das Diagramm "kommutiert": Egal, welchen Weg man von $user zu md5($passwd) geht, so erhält man doch immer das gleiche Ergebnis.

SALT in *md5()*

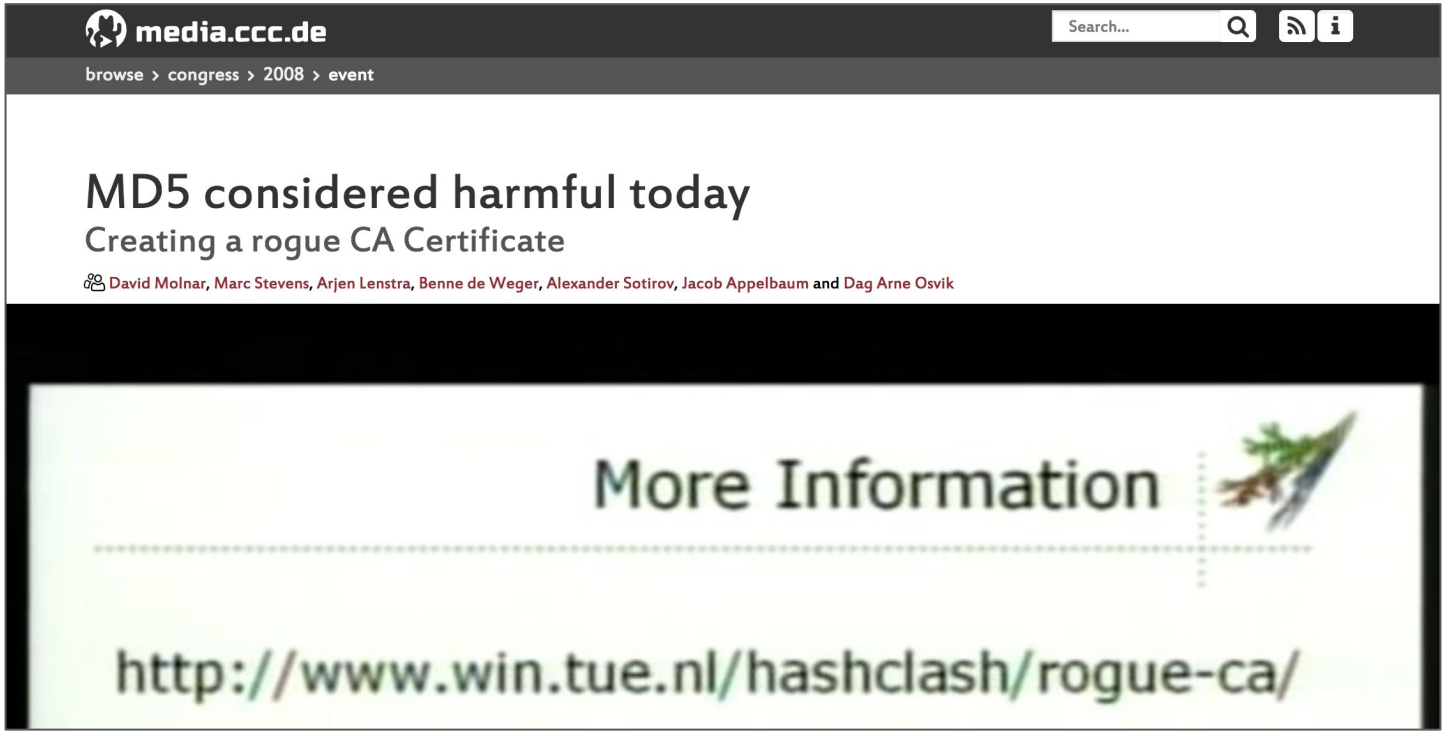
```php
<?PHP

  $SALT = '#khaszi398fhlkvkhjewoiu+';

  function salter( $x ){
     global $SALT;
     return md5( $x . $SALT );
  }

  $MD5_Secrets = [];

  foreach( $My_Secrets as $key => $value ){
     $MD5_Secrets[ salter($key) ] = salter($value);
  }

  var_dump($MD5_Secrets);

?>
```

Konkatenation

Zusätzliche Erschwernis:
Es reicht nicht, md5 zu knacken, sondern man muss auch den SALT raten, um an die Klartexte zu kommen.

- SALT geheim halten

md5() considered harmful today



https://media.ccc.de/v/25c3-3023-en-making_the_theoretical_possible

Eine bessere Hashfunktion wählen:

hash() statt *md5()*

```
string hash ( string $algo , string $data [, bool $raw_output = FALSE ] )
```

```
hash( "sha384", "klartext" . SALT );
```

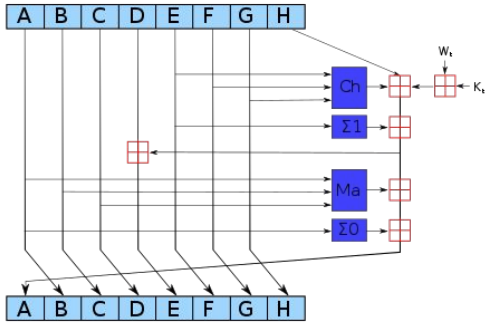
md2	32	a9046c73e00331af68917d3804f70655
md4	32	866437cb7a794bce2b727acc0362ee27
md5	32	5d41402abc4b2a76b9719d911017c592
sha1	40	aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
sha256	64	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e730
sha384	**96**	**59e1748777448c69de6b800d7a33bbfb9ff1b463e44354c3553**
sha512	128	9b71d224bd62f3785d96d46ad3ea3d73319bfbc2890caadae2d
ripemd128	32	789d569f08ed7055e94b4289a4195012
ripemd160	40	108f07b8382412612c048d07d13f814118445acd
ripemd256	64	cc1d2594aece0a064b7aed75a57283d9490fd5705ed3d66bf9a
ripemd320	80	eb0cf45114c56a8421fbcb33430fa22e0cd607560a88bbe14ce
whirlpool	128	0a25f55d7308eca6b9567a7ed3bd1b46327f0f1ffdc804dd8bb
tiger128,3	32	a78862336f7ffd2c8a3874f89b1b74f2
tiger160,3	40	a78862336f7ffd2c8a3874f89b1b74f2f27bdbca
tiger192,3	48	a78862336f7ffd2c8a3874f89b1b74f2f27bdbca39660254
tiger128,4	32	1c2a939f230ee5e828f5d0eae5947135
tiger160,4	40	1c2a939f230ee5e828f5d0eae5947135741cd0ae
tiger192,4	48	1c2a939f230ee5e828f5d0eae5947135741cd0aefeeb2adc
snefru	64	7c5f22b1a92d9470efea37ec6ed00b2357a4ce3c41aa6e28e3b
gost	64	a7eb5d08ddf2363f1ea0317a803fcef81d33863c8b2f9f6d7d1
adler32	8	062c0215
crc32	8	3d653119
crc32b	8	3610a686

haval128,3	32	85c3e4fac0ba4d85519978fdc3d1d9be
haval160,3	40	0e53b29ad41cea507a343cdd8b62106864f6b3fe
haval192,3	48	bfaf81218bbb8ee51b600f5088c4b8601558ff56e2de1c4f
haval224,3	56	92d0e3354be5d525616f217660e0f860b5d472a9cb99d6766be
haval256,3	64	26718e4fb05595cb8703a672a8ae91eea071cac5e7426173d4c
haval128,4	32	fe10754e0b31d69d4ece9c7a46e044e5
haval160,4	40	b9afd44b015f8afce44e4e02d8b908ed857afbd1
haval192,4	48	ae73833a09e84691d0214f360ee5027396f12599e3618118
haval224,4	56	e1ad67dc7a5901496b15dab92c2715de4b120af2baf661ecd92
haval256,4	64	2d39577df3a6a63168826b2a10f07a65a676f5776a0772e0a87
haval128,5	32	d20e920d5be9d9d34855accb501d1987
haval160,5	40	dac5e2024bfea142e53d1422b90c9ee2c8187cc6
haval192,5	48	bbb99b1e989ec3174019b20792fd92dd67175c2ff6ce5965
haval224,5	56	aa6551d75e33a9c5cd4141e9a068b1fc7b6d847f85c3ab16295
haval256,5	64	348298791817d5088a6de6c1b6364756d404a50bd64e645035f

http://de2.php.net/manual/de/function.hash.php

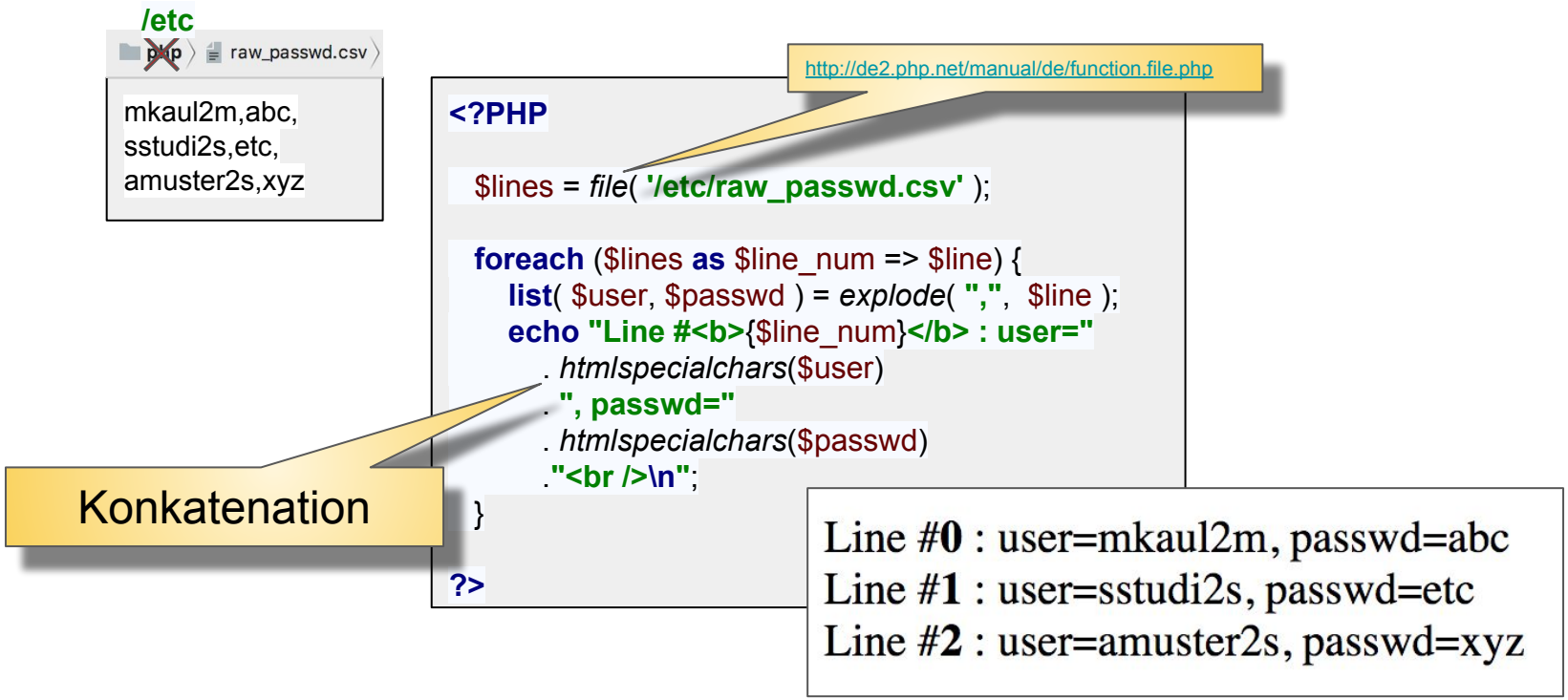
Secure Hash Algorithm (**sha**)

- Der Begriff Secure Hash Algorithm (kurz SHA, englisch für sicherer Hash-Algorithmus) bezeichnet eine Gruppe standardisierter kryptologischer Hashfunktionen.
- Diese dienen zur Berechnung eines Prüfwerts für beliebige digitale Daten
- Grundlage zur Erstellung einer digitalen Signatur
- SHA-1 hielt bis 2005.
- SHA-2-Familie (SHA-224, SHA-256, SHA-384, SHA-512).
- SHA-3-Familie (SHA3-224, SHA3-256, SHA3-384, SHA3-512)



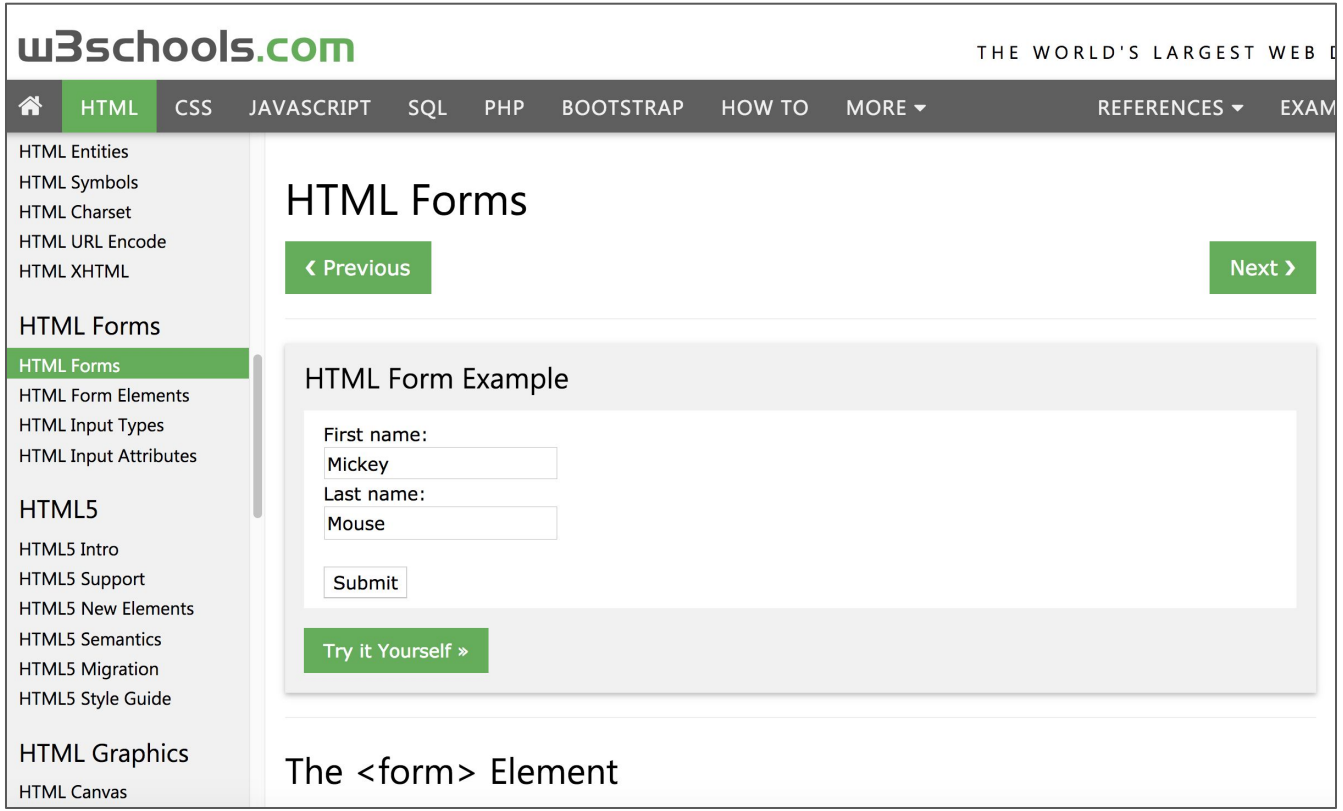
https://de.wikipedia.org/wiki/SHA-2
https://de.wikipedia.org/wiki/SHA-3

Server-Dateien lesen und mit PHP verarbeiten



http://de2.php.net/manual/de/function.explode.php

HTML Formulare



https://www.w3schools.com/html/html_forms.asp

Eingabe neuer Benutzer und Passwörter

php 〉 register.php 〉

```
<!doctype html>
<h1>Register</h1>
<form>
  <fieldset>
    <legend>Register a new Account:</legend>
    Account name:<br>
    <input type="text" name="account">
    <br>
    Password:<br>
    <input type="password" name="password">
    <br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
…
```

php 〉 register.php 〉 Fortsetzung

```
…
<?PHP

  if ( isset($_GET[ 'account'  ]) && isset($_GET[ 'password' ]) ){

     $account = $_GET[ 'account'  ];
     $passwd  = $_GET[ 'password' ];

     $file = './raw_passwd.csv';
     $new_line = $account . ',' . $passwd . "\n";

     if ( file_put_contents( $file, $new_line, FILE_APPEND | LOCK_EX ) ){
        echo "<script>alert('Registered successfully!')</script>";
     }

  }

?>
```

https://www.php.net/manual/de/function.file-put-contents.php

Nachteil

Request URL: `http://localhost:63343/php/register.php?account=my_account&password=my_passwd`

Request Method: `GET`

Status Code: 🟢 `200 OK`

- Passwort erscheint im Klartext in der URL
  - als HTTP-GET-Request-Parameter
- Passwort erscheint im Klartext im Log des Server:

```
access.log        access.log.2.gz   access.log.4.gz   error.log.21.gz  error.log.41.gz
access.log.1      access.log.30.gz  access.log.50.gz  error.log.22.gz  error.log.42.gz
access.log.10.gz  access.log.31.gz  access.log.51.gz  error.log.23.gz  error.log.43.gz
access.log.11.gz  access.log.32.gz  access.log.52.gz  error.log.24.gz  error.log.44.gz
access.log.12.gz  access.log.33.gz  access.log.5.gz   error.log.25.gz  error.log.45.gz
access.log.13.gz  access.log.34.gz  access.log.6.gz   error.log.26.gz  error.log.46.gz
access.log.14.gz  access.log.35.gz  access.log.7.gz   error.log.27.gz  error.log.47.gz
access.log.15.gz  access.log.36.gz  access.log.8.gz   error.log.28.gz  error.log.48.gz
access.log.16.gz  access.log.37.gz  access.log.9.gz   error.log.29.gz  error.log.49.gz
access.log.17.gz  access.log.38.gz  error.log         error.log.2.gz   error.log.4.gz
access.log.18.gz  access.log.39.gz  error.log.1       error.log.30.gz  error.log.50.gz
access.log.19.gz  access.log.3.gz   error.log.10.gz   error.log.31.gz  error.log.51.gz
access.log.20.gz  access.log.40.gz  error.log.11.gz   error.log.32.gz  error.log.52.gz
access.log.21.gz  access.log.41.gz  error.log.12.gz   error.log.33.gz  error.log.5.gz
access.log.22.gz  access.log.42.gz  error.log.13.gz   error.log.34.gz  error.log.6.gz
access.log.23.gz  access.log.43.gz  error.log.14.gz   error.log.35.gz  error.log.7.gz
access.log.24.gz  access.log.44.gz  error.log.15.gz   error.log.36.gz  error.log.8.gz
access.log.25.gz  access.log.45.gz  error.log.16.gz   error.log.37.gz  error.log.9.gz
```

Lösung POST statt GET

Request URL: `http://localhost:63343/php/register.php`

Request Method: `POST`

Status Code: 200 OK

```
<!doctype html>
<h1>Register</h1>
<form method="post">
  <fieldset>
    <legend>Register a new Account:</legend>
    Account name:<br>
    <input type="text" name="account">
    <br>
    Password:<br>
    <input type="password" name="password">
    <br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

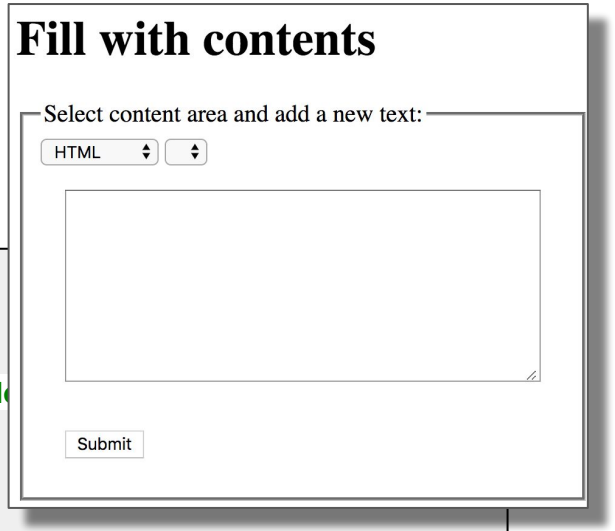
```
<?PHP

  if ( isset($_POST[ 'account'  ]) && isset($_POST[ 'password' ]) ){

     $account = $_POST[ 'account'  ];
     $passwd  = $_POST[ 'password' ];

     $file = './raw_passwd.csv';
     $new_line = $account . ',' . $passwd . "\n";

     if ( file_put_contents( $file, $new_line, FILE_APPEND | LOCK_EX ) ){
        echo "<script>alert('Registered successfully!')</script>";
     }

  }

?>
```

WWW-Navigator mit Daten füllen

php › data.json

```
{
  "html": {
    "headings": {
      "h1": "First order Header",
      "h2": "Second order Header"
    },
    "paragraphs": {},
    "links": {},
    "images": {},
    "tables": {}
  },
  "css": {
    "selectors": {},
    "colors": {},
    "boxes": {},
    "display": {},
    "float": {}
  },
  "javascript": {
    "functions": {},
    "objects": {},
    "scope": {},
    "events": {},
    "hoisting": {},
    "strict_mode": {},
    "JSON": {}
  }
}
```

```
<!doctype html>
<h1>Fill with contents</h1>
<style>
  textarea { margin: 1rem; display: bl
  input { margin: 1rem; }
</style>
<form method="post">
  <fieldset>
    <legend>Select content area and add a new text:</legend>
    <select name="top_header">
      <option value="html">HTML</option>
      <option value="css">CSS</option>
      <option value="javascript">JavaScript</option>
    </select>
    <select name="sub_header">
    </select>
    <textarea name="content"></textarea>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```



Subheader-Menü erzeugen

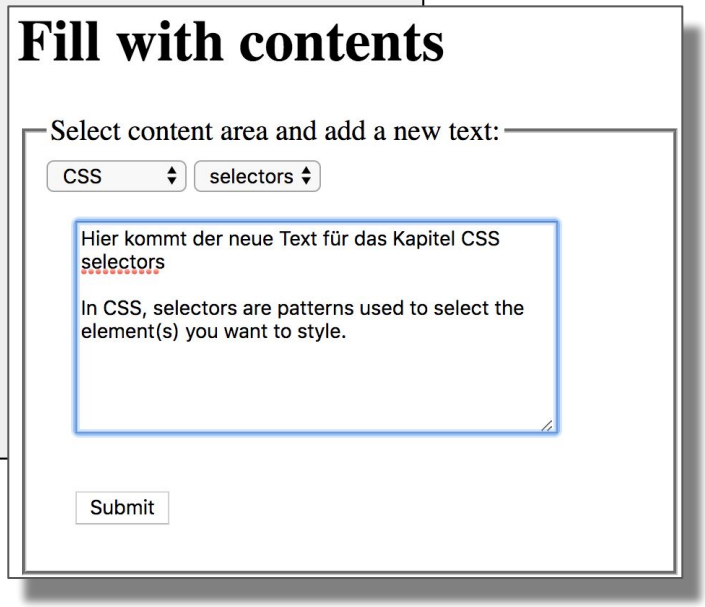
```
<!doctype html>
<h1>Fill with contents</h1>
<style>
  textarea { margin: 1rem; display: block; }
  input { margin: 1rem; }
</style>
<form method="post">
  <fieldset>
    <legend>Select content area and ac
    <select name="top_header">
      <option value="html">HTML</o
      <option value="css">CSS</opti
      <option value="javascript">Jav
    </select>
    <select name="sub_header">
    </select>
    <textarea name="content"></texta
    <input type="submit" value="Sub
  </fieldset>
</form>
```

```
<?PHP
  $file = './data.json';
  $contents = file_get_contents( $file );
  $json = json_decode( $contents, true );
?>
<script>
  let json = <?PHP echo json_encode( $json ) ?>;
  const top_header = document.querySelector('select[name="top_header"]');
  const sub_header = document.querySelector('select[name="sub_header"]');
  top_header.addEventListener('change', e => {
    Object.keys( json[ e.target.value ] ).forEach( key => {
      const option = document.createElement('option');
      option.value = key;
      option.innerText = key;
      sub_header.append( option );
    });
  });
</script>
```

Eingegebene Daten persistent auf dem Server speichern: Update von data.json

```php
<?PHP

  if ( isset($_POST[ 'top_header'  ]) && isset($_POST[ 'sub_header' ]) && isset($_POST[ 'content' ]) ){

      $top_header = $_POST[ 'top_header'  ];
      $sub_header = $_POST[ 'sub_header' ];
      $content    = $_POST[ 'content' ];

      $json[ $top_header ][ $sub_header ] = $content;

      if ( file_put_contents( $file, json_encode( $json, true )) ){
          echo "<script>alert('Content is entered successfully!')</script>";
      }

  }

?>
```

... noch unfertig ...



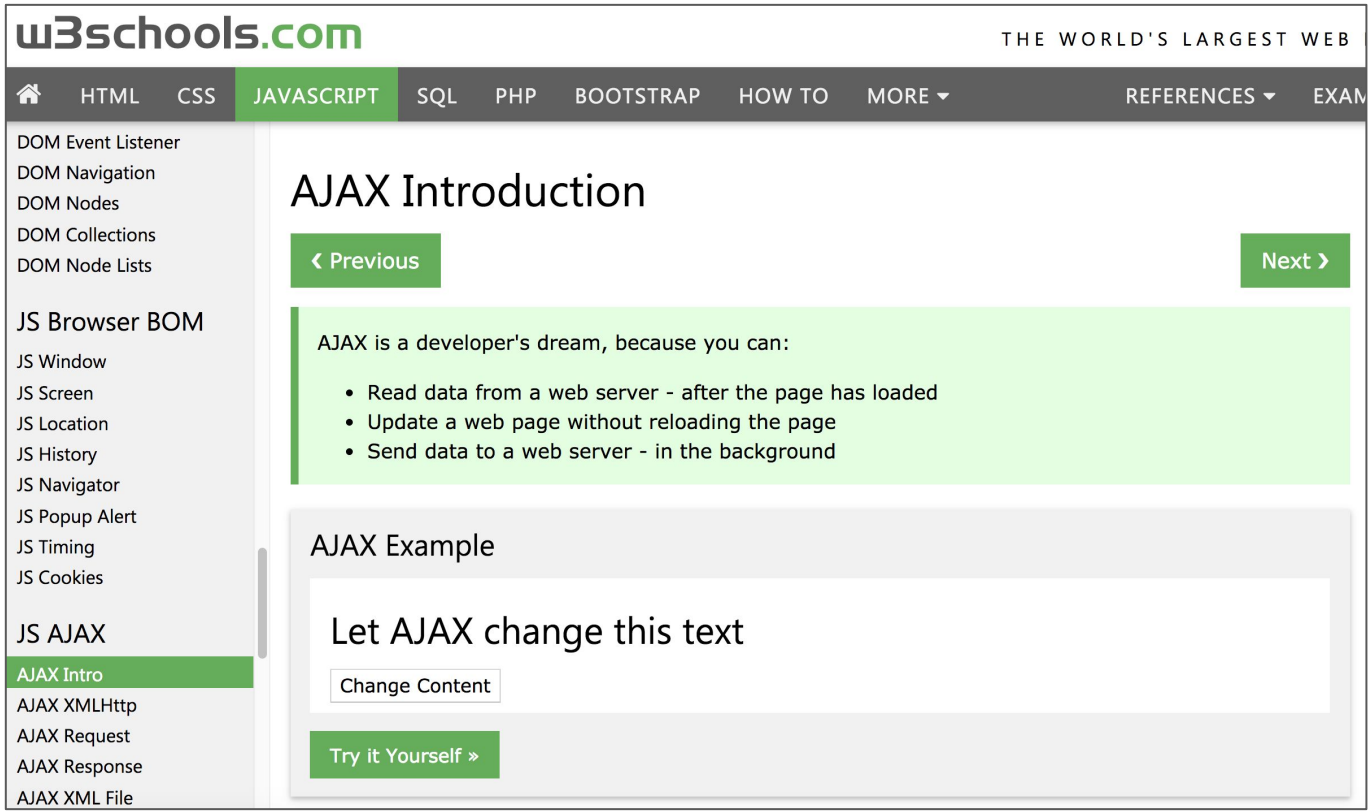
PHP passt gut zur Realität des WWW heute: Der Browser ist ein Multi-Client für viele Dienste auf verteilten Servern

Der Browser ist die Integrationsplattform.

Micro Services

in Java, PHP, JS, Ruby, Elixir, Erlang, Go, ...

Microservices benutzen AJAX



https://www.w3schools.com/js/js_ajax_intro.asp

Vermeide Page Refresh. Vermeide synchrone Operationen. Der Browser soll nie blockieren.

Server-Zugriff bei Click-Event

Asynchrones Nachladen ohne Page Refresh

```
<!doctype html>
<h1>Login</h1>
<form id="login-form">
<div>
  <span>Name: </span>
  <input type="text" name="name">
</div>
<div>
  <span>Passwd: </span>
  <input type="text" name="passwd">
</div>
  <div><button>Submit!</button></div>
</form>
```

Statt <input type="submit" value="Submit">

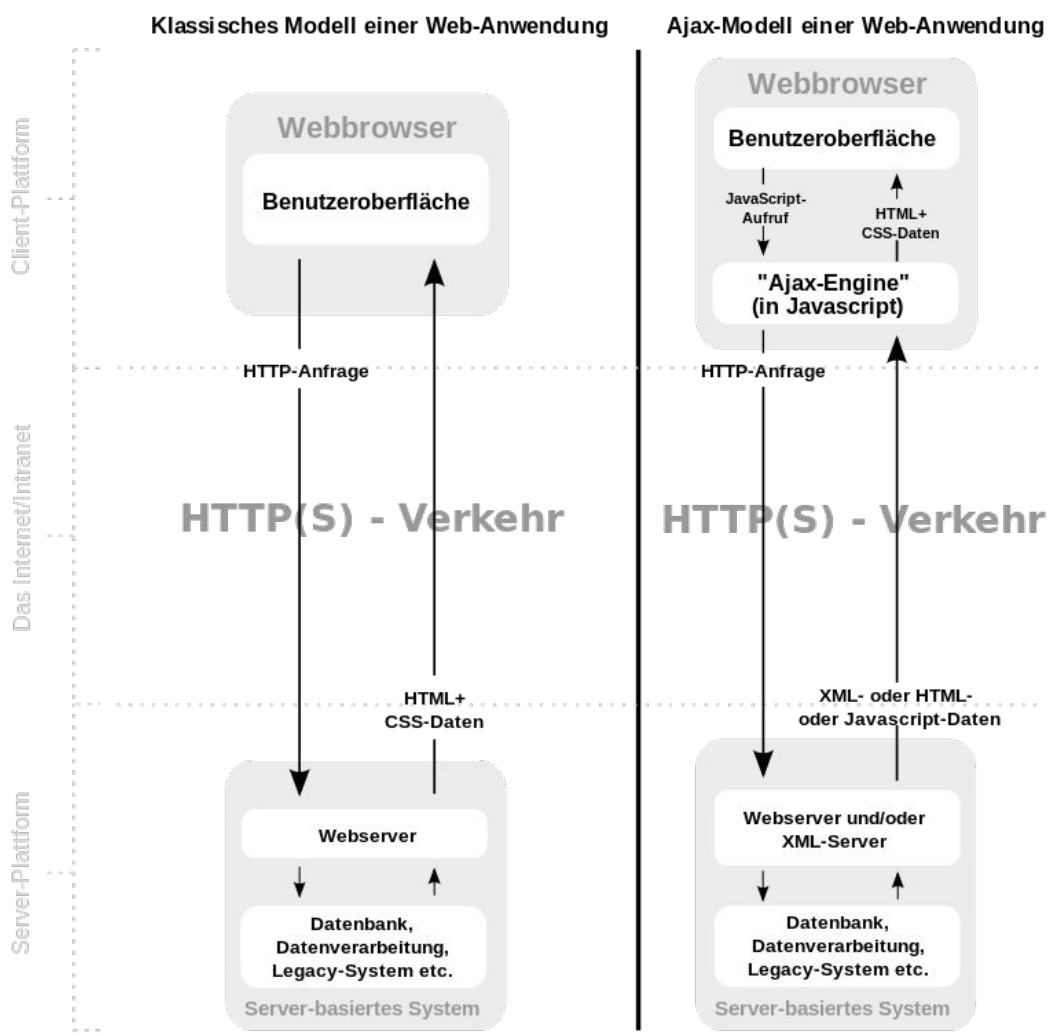
```
<script>
    document.querySelector('button').addEventListener('click', e => {
      fetch("./login.php", {
        method: "POST",
        body: new FormData(document.getElementById('login-form'))
      }).then(function(res) {
        if (res.ok) {
          alert("Perfect! Your settings are saved.");
        } else if (res.status === 401) {
          alert("Oops! You are not authorized.");
        }
      }, function(e) {
        alert("Error submitting form!");
      });
    });
</script>
```

... noch unfertig ...

Ajax

- Ajax = "Asynchronous JavaScript and XML"
- Begriff def. von Jesse James Garret 2005

- Neuer Interaktionsstil:
  - Update ohne Page Reload



https://de.wikipedia.org/wiki/Ajax_(Programmierung)

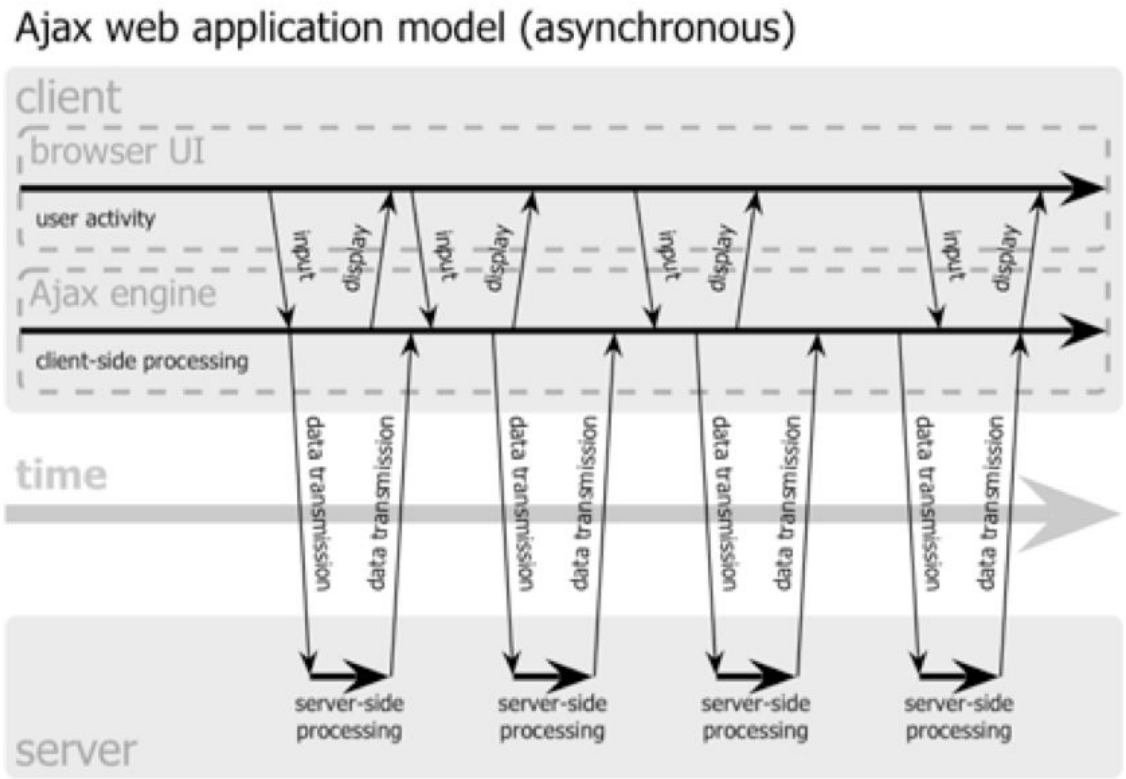
Durch AJAX wird das WWW wird asynchron

WWW war synchron:

- GET-Request
- Page Load

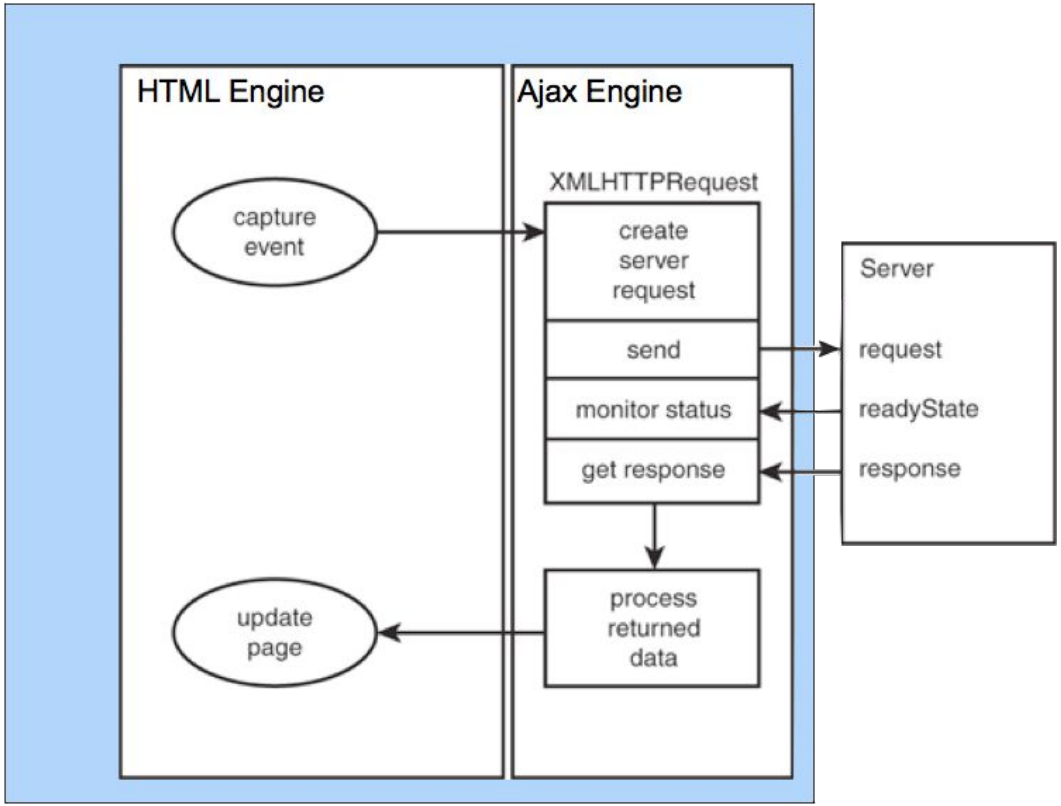
⇒ Problem:
Browser blockiert

Durch Ajax Page Refresh asynchron



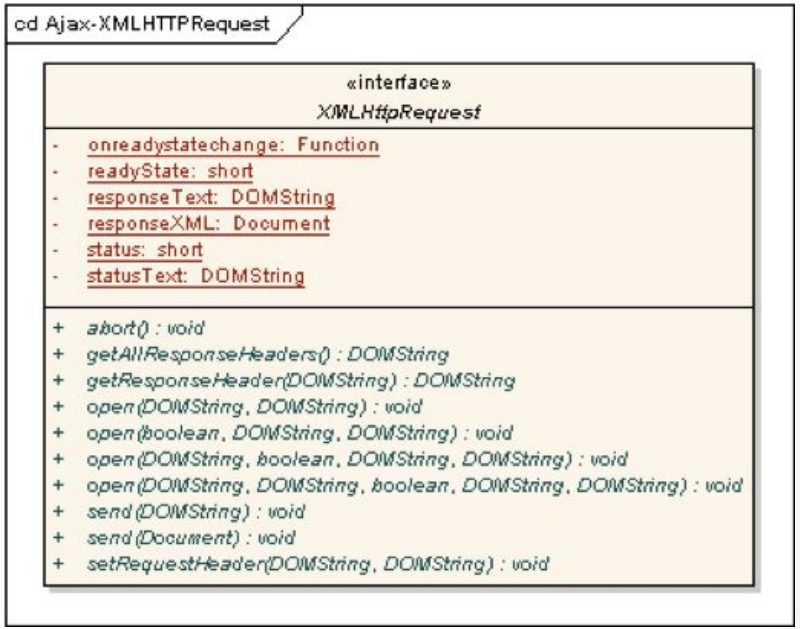
Quelle Jesse James Garrett 2005
http://www.adaptivepath.com/publications/essays/archives/000385.php

XMLHttpRequest

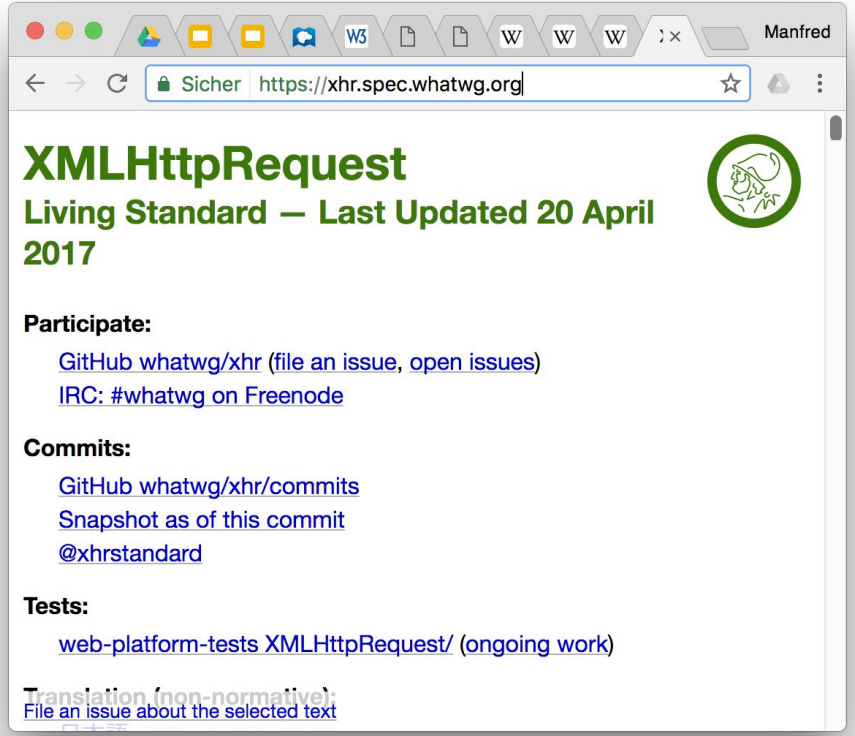


- XMLHttpRequest ist das Interface des Browsers das die Ajax-API zur Verfügung stellt
- Aufruf:
  ```
  var xhr = new XMLHttpRequest();
  ```
- synchron / asynchron
- nicht nur XML

Spezifikation durch W3C & WHATWG



Quelle: http://de.wikipedia.org/wiki/XMLHttpRequest



Spezifikation durch WHATWG

Ajax-API

https://xhr.spec.whatwg.org/

4 Interface XMLHttpRequest

1. 4.1 Constructors
2. 4.2 Garbage collection
3. 4.3 Event handlers
4. 4.4 States
5. 4.5 Request
    1. 4.5.1 The open() method
    2. 4.5.2 The setRequestHeader() method
    3. 4.5.3 The timeout attribute
    4. 4.5.4 The withCredentials attribute
    5. 4.5.5 The upload attribute
    6. 4.5.6 The send() method
    7. 4.5.7 The abort() method
6. 4.6 Response
    1. 4.6.1 The responseURL attribute
    2. 4.6.2 The status attribute
    3. 4.6.3 The statusText attribute
    4. 4.6.4 The getResponseHeader() method
    5. 4.6.5 The getAllResponseHeaders() method
    6. 4.6.6 Response body
    7. 4.6.7 The overrideMimeType() method
    8. 4.6.8 The responseType attribute
    9. 4.6.9 The response attribute
    10. 4.6.10 The responseText attribute
    11. 4.6.11 The responseXML attribute
7. 4.7 Events summary

Ajax-Beispiel

http://origin-server.de/index.html

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Ajax01</title>
</head>
<body>
<div id="id01"></div>
<script>
 var xhr = new XMLHttpRequest();

 xhr.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("id01").innerHTML = this.responseText;
  } else {
   console.log( this.statusText );
  }
 };
 xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.json", true);
 xhr.send();
</script>
</body>
</html>
```

Container-DIV, welches die Daten empfangen soll

Erzeugung eines neuen xhr-Objektes.

EventHandler: Was soll passieren, wenn die Antwort vollständig eingetroffen ist?

Ajax-Request cross domain

true = asynchron

⇒ Cross Domain-Anfrage

⇒ Fehlermeldung:

```
Quellübergreifende (Cross-Origin) Anfrage blockiert: Die Gleiche-Quelle-Regel verbietet (unbekannt)
das Lesen der externen Ressource auf http://kaul.inf.h-brs.de/data/wem/www.json. (Grund:
CORS-Kopfzeile 'Access-Control-Allow-Origin' fehlt).
```

SOP und CORS

SOP = Same Origin Policy

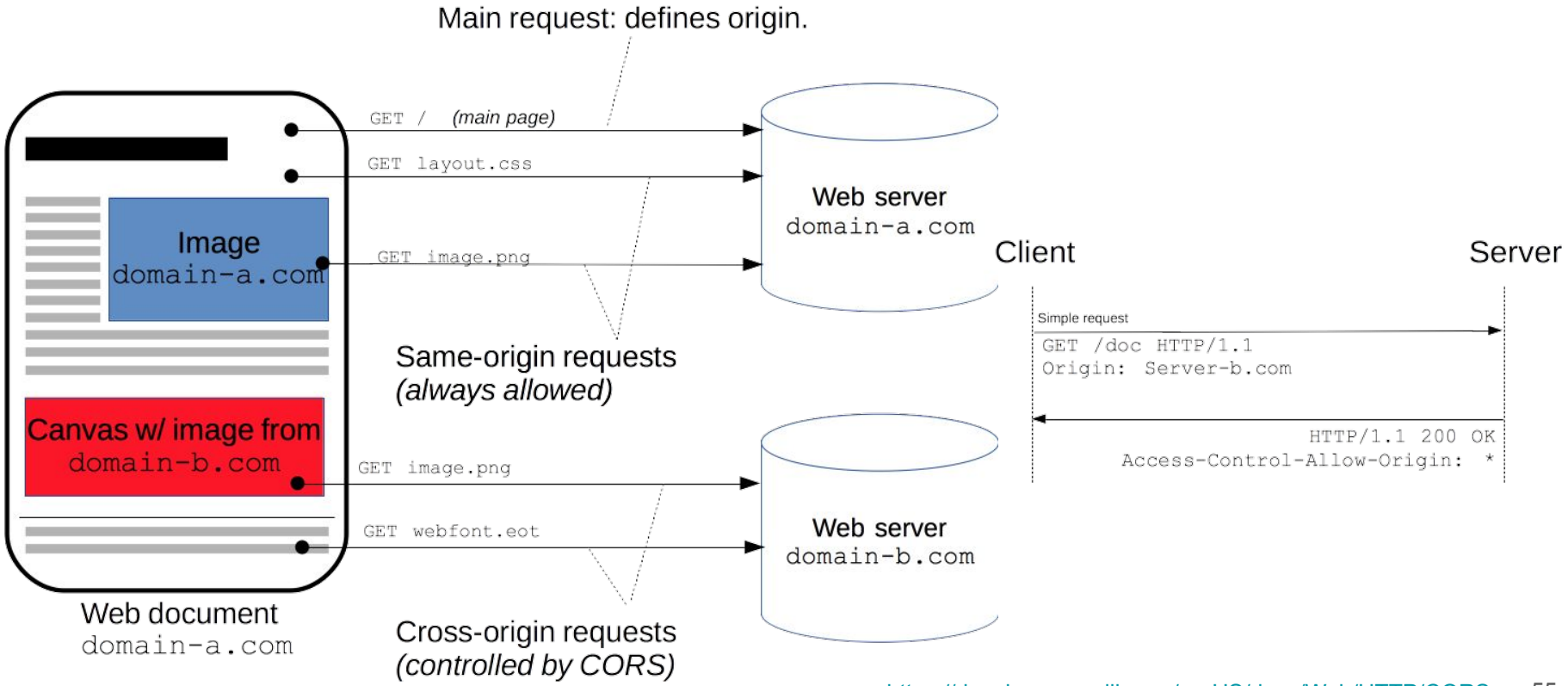
Cross-Origin Resource Sharing

In **http://www.example.com/dir/page.html** eingebettetes Skript versucht, auf ein Element in den folgenden Seiten zuzugreifen:

angesprochene URL	Ergebnis	Grund
http://www.example.com/dir/page2.html	✓ **Ja**	selbes Protokoll und Host
http://www.example.com/dir2/other.html	✓ **Ja**	selbes Protokoll und Host
http://www.example.com:**81**/dir/other.html	✗ **Nein**	selbes Protokoll und Host, aber anderer Port
https://www.example.com/dir/other.html	✗ **Nein**	anderes Protokoll
http://**en.example.com**/dir/other.html	✗ **Nein**	anderer Host
http://**example.com**/dir/other.html	✗ **Nein**	anderer Host (genaue Übereinstimmung benötigt)
http://**v2.www.example.com**/dir/other.html	✗ **Nein**	anderer Host (genaue Übereinstimmung benötigt)
http://www.example.com:**80**/dir/other.html	N/A	Port eindeutig. Hängt von der Implementierung des Browsers ab.

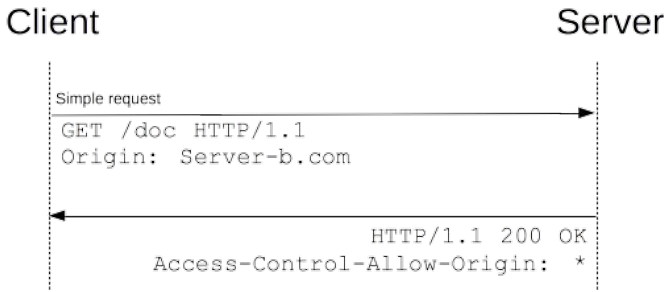
https://de.wikipedia.org/wiki/Same-Origin-Policy

Cross-Origin Resource Sharing (CORS)



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Lösung mit CORS



HTTP-CORS-Header erzeugen mit .htaccess

Server 1	**/path/ajax01.html** *... xhr*.open(**"GET"**, **"http://kaul.inf.h-brs.de/data/wem/www.json"**, **true**); ...
Server 2	**/data/wem/www.json**

Lösung mit .htaccess auf Server 2

.htaccess

```
Header set Access-Control-Allow-Origin "*"
Header set Access-Control-Allow-Methods "GET"
Header set Access-Control-Allow-Headers "content-type"
```

oder "**http://localhost:63342**"

https://de.wikipedia.org/wiki/Cross-Origin_Resource_Sharing

CORS-Lösung mit PHP

wenn der Server nicht für CORS konfiguriert ist, (z.B. das CORS-Modul in Apache nicht installiert), kann man sich mit PHP behelfen wie folgt:

Man kann in PHP auch HTTP-Header senden

Server 1	/path/ajax01.html `xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.json.php", true);`
Server 2	**/data/wem/www.json.php**

/data/wem/www.json.php

oder "*"

```
<?php
// Allow from any origin
if (isset($_SERVER['HTTP_ORIGIN'])) {
   header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
   header('Access-Control-Allow-Credentials: true');
   header('Access-Control-Max-Age: 86400');    // cache for 1 day
}
// Access-Control headers are received during OPTIONS requests
if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {

   if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
      header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

   if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
      header("Access-Control-Allow-Headers:
{$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");
}
echo "... JSON data ";
?>
```

http://stackoverflow.com/questions/4950331/secure-php-file-upload-script

Erlaubte "Cross Origin Resources"

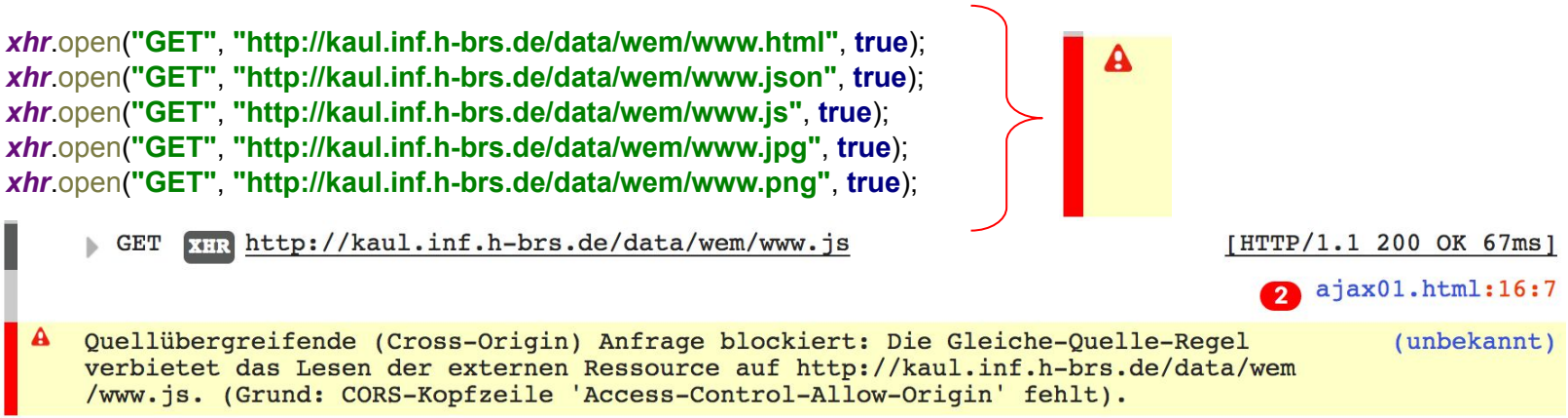
Here are some examples of resources which may be embedded cross-origin:

- JavaScript with `<script src="..."></script>`. Error messages for syntax errors are only available for same-origin scripts.
- CSS with `<link rel="stylesheet" href="...">`. Due to the relaxed syntax rules of CSS, cross-origin CSS requires a correct `Content-Type` header. Restrictions vary by browser: IE, Firefox, Chrome, Safari (scroll down to CVE-2010-0051) and Opera.
- Images with `<img>`. Supported image formats include PNG, JPEG, GIF, BMP, SVG, ...
- Media files with `<video>` and `<audio>`.
- Plug-ins with `<object>`, `<embed>` and `<applet>`.
- Fonts with `@font-face`. Some browsers allow cross-origin fonts, others require same-origin fonts.
- Anything with `<frame>` and `<iframe>`. A site can use the `X-Frame-Options` header to prevent this form of cross-origin interaction.

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Sicherheitseinschränkung wg. SOP: Verbotene "Cross Origin Resources"

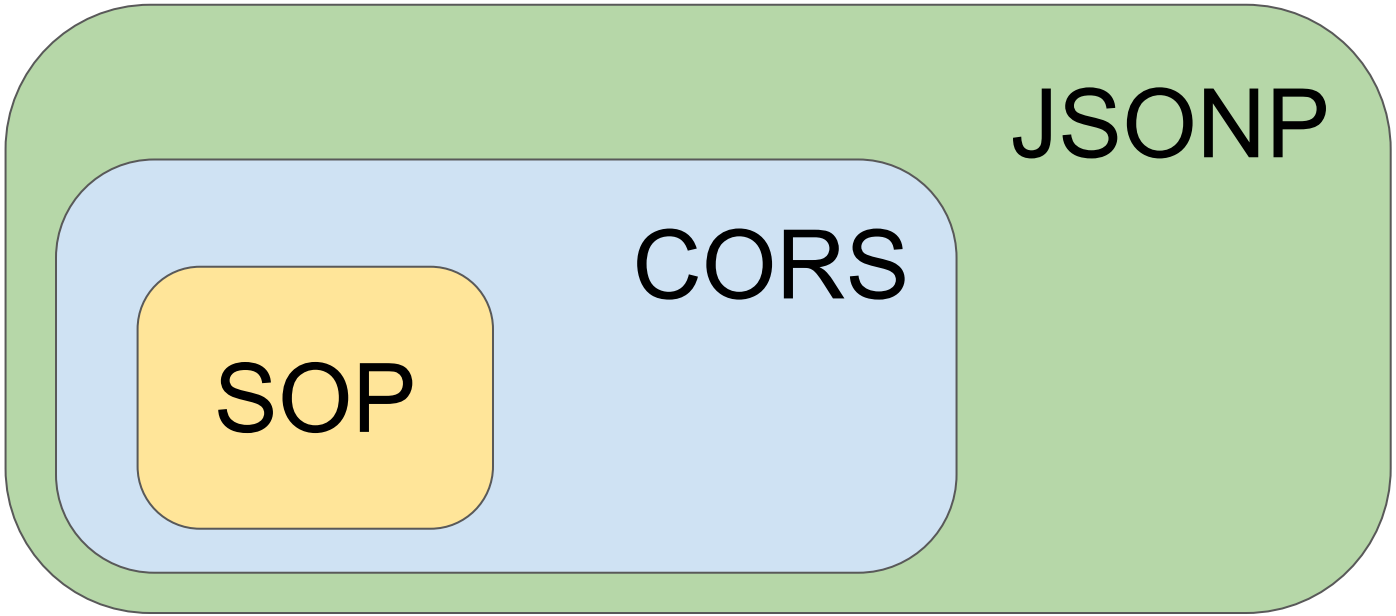
```
xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.html", true);
xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.json", true);
xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.js", true);
xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.jpg", true);
xhr.open("GET", "http://kaul.inf.h-brs.de/data/wem/www.png", true);
```



Sandbox-Prinzip für

- Cookies, authorization headers or TLS client certificates
- <iframe>, <img>, <audio> und <video> können zwar cross origin geladen und angehört bzw. angeschaut werden, aber **nicht von JavaScript aus** gelesen, bearbeitet oder manipuliert werden.

Sandboxing mit SOP und darüber hinaus



https://de.wikipedia.org/wiki/JavaScript_Object_Notation#JSONP

JSONP = JSON with Padding

- JSONP = Cross Origin Daten via Skript laden und ausführen ohne CORS

http://kaul.inf.h-brs.de/data/wem/data.js

```
showPrice({symbol: 'IBM', price: 91.42});
```

1. Daten

2. Verpackung

3. Nutzung

http://localhost:63342/jsonp.html

```
<!DOCTYPE html>
<script>
 // This is our function to be called with JSON data
 function showPrice(data) {
  alert("Symbol:"+data.symbol+",Price:"+data.price);
 }
 var url = "http://kaul.inf.h-brs.de/data/wem/data.js"; // remote script

 // dynamic script insertion follows
 var script = document.createElement('script');
 script.setAttribute('src', url);

 // load and execute the script
 document.getElementsByTagName('head')[0].appendChild(script);
</script>
```

Skript zum Erzeugen eines Skript-Elementes

http://www.ibm.com/developerworks/library/wa-aj-jsonp1/

Anwendung JSONP bei Flickr

```
<!DOCTYPE html>
<script src="https://code.jquery.com/jquery-2.2.3.js"></script>
<h1>Include Flickr via JSONP</h1>
<div id="images"></div>
<script>
 $.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?jsoncallback=?",
   {
     tags:"cat",
     tagmode:"any",
     format:"json"
   },
   function (data) {
     $.each(data.items, function (i, item) {
       $("<img/>").attr("src", item.media.m).appendTo("#images");
       if (i == 3) return false;
     });
   });
</script>
```

http://www.ibm.com/developerworks/library/wa-aj-jsonp1/

jQuery-Funktionen für Ajax

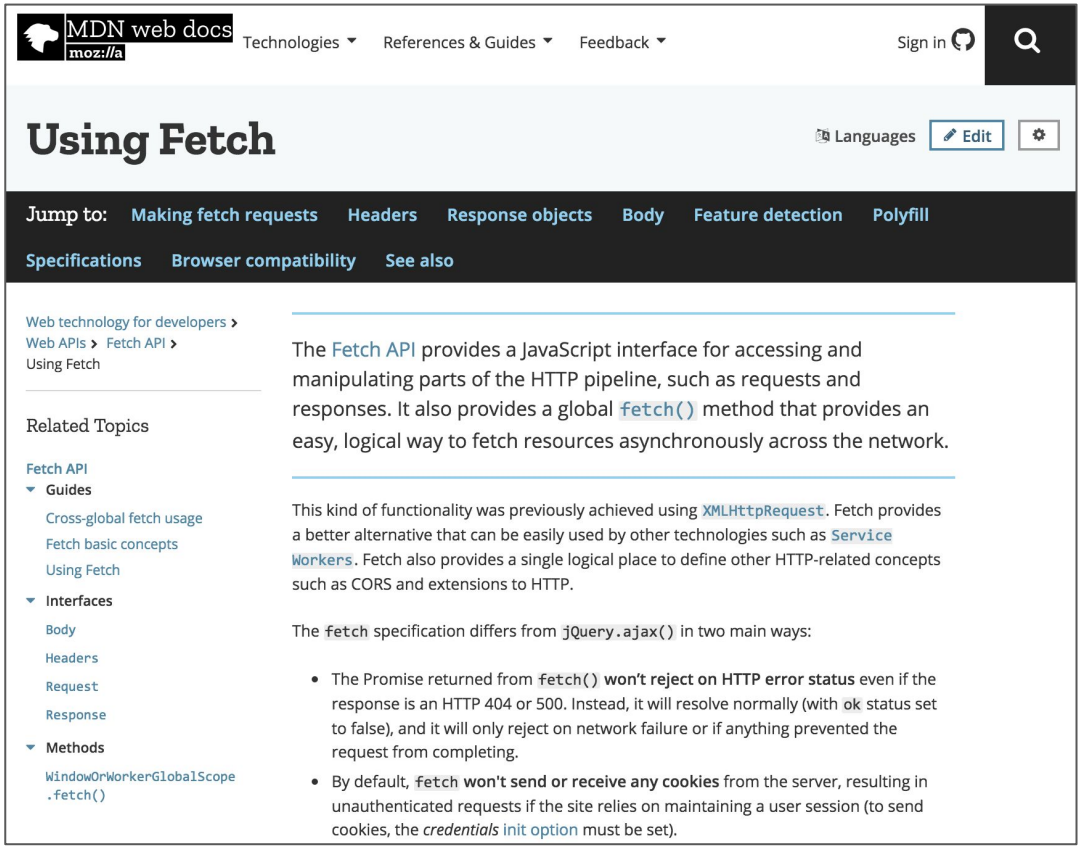
https://api.jquery.com/category/ajax/shorthand-methods/

jQuery function	API	GET	POST
load()	load(url, data, callback)	YES	YES
$.getJSON()	$.getJSON(url, data, callback)	YES	NO
$.getScript()	$.getScript(url, callback)	NO	NO
$.get()	$.get(url, data, callback, type)	YES	NO
$.post()	$.post(url, data, callback, type)	NO	YES
$.ajax()	$.ajax(options)	YES	YES

http://net.tutsplus.com/tutorials/javascript-ajax/5-ways-to-make-ajax-calls-with-jquery/

Die fetch-API als Nachfolger von XMLHttpRequest

Dann braucht man kein XMLHttpRequest und kein jQuery mehr ...



https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Mit fetch Formulardaten auf dem Server abspeichern

```
<h1>POST Form</h1>
<div>
  <label for="name">Name:</label>
  <input type="text" id="name" class="form">
</div>
<div>
  <label for="password">Password:</label>
  <input type="password" id="password" class="form">
</div>
<input type="submit" id="submit">
```

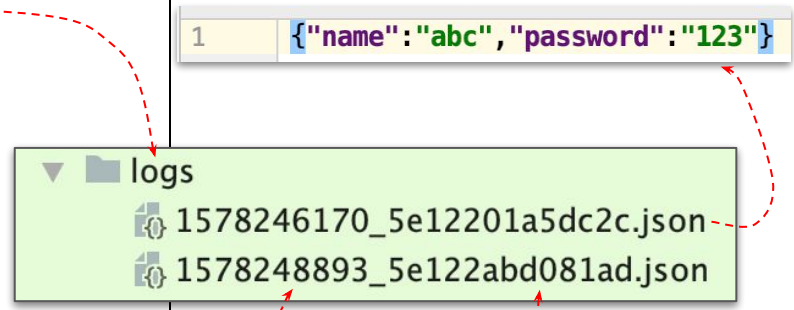
```
<script>
  const submit=document.getElementById('submit');
  submit.addEventListener('click', function( event ){
    fetch( new Request( "./server.php" ), {
      method: 'POST',
      mode: 'cors',
      cache: 'no-store',
      body: JSON.stringify( Array.from( document.querySelectorAll('.form') )
          .reduce( ( json, input_field )=> {
              json[ input_field.id ] = input_field.value;
              return json }, {} ) ),
      headers:{
        'Content-Type': 'application/json'
      }
    });
  });
</script>
```

https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/fetch

server.php: (1.) HTTP-POST-Request

```php
<?PHP
 include("./response.php");
 include("./cors.php");
 // single server script for handling both:
 // POSTing data to the server AND GETting all data from the server

 // Constant LOG_DIR - where to store log files
 // security: locate directory of log files outside of WWW folders
 define("LOG_DIR", "../logs/");

 if ( $_SERVER['REQUEST_METHOD'] == 'POST' ) { // HTTP POST request

    // get raw POST data string
    $raw_data = file_get_contents('php://input');

    // create unique filename with time stamp and unique ID
    $filename = LOG_DIR . time() . '_' . uniqid() . ".json";

    // write raw data to the new file
    file_put_contents( $filename, $raw_data );

  } else if ($_SERVER['REQUEST_METHOD'] == 'GET') { // HTTP GET request ...   }

?>
```

Die Daten jedes Post-Requests werden in einer separaten Datei gespeichert. ⇒ Nebenläufigkeits-artefakte werden dadurch ausgeschlossen.



http://www.php.net/manual/en/function.file-put-contents.php

server.php: (2.) HTTP-GET-Request

```php
<?PHP
 include("./response.php");
 include("./cors.php");
 define("LOG_DIR", "../logs/");

if ( $_SERVER['REQUEST_METHOD'] == 'POST' ) { // HTTP POST request …
} else if ($_SERVER['REQUEST_METHOD'] == 'GET') { // HTTP GET request

   // collect all log files from log directory
   // decode JSON into PHP arrays

   // collect all files in the directory "LOG_DIR" except '.', '..'
   $files = array_diff( scandir(LOG_DIR), array('.', '..') );

   // push all of them into a single large array "$all"
   $all = array();
   foreach( $files as $file ){
     // array push
     $all[] = json_decode( file_get_contents( LOG_DIR . $file ), false, 512, JSON_UNESCAPED_UNICODE );
   }

   // encode the large array and send it back to the client
   response( $all );
 }

?>
```

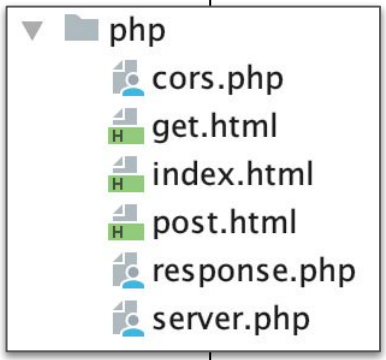
http://www.php.net/manual/en/function.file-get-contents.php

response.php: JSON-Daten für fetch & JSONP

```php
<?PHP
 /**
  * finish php script and send message to client
  * in AJAX calls e.g. fetch requests
  * e.g. JSONP requests
  * @param $msg message
  */
 function response( $msg ){
  // encode message to JSON
  $msg = json_encode( $msg, JSON_UNESCAPED_UNICODE );

  // filter received dynamic function name (jsonp)
  $callback = filter_input( INPUT_GET, 'callback', FILTER_SANITIZE_STRING );

  // is cross domain request?
  if ( isset( $callback ) ) {
   // padding message in function call (jsonp)
   $msg = $callback.'('.$msg.');';
  }

  // send message to client
  die( $msg );
 }
?>
```

cors.php: Cross-Origin-Requests erlauben

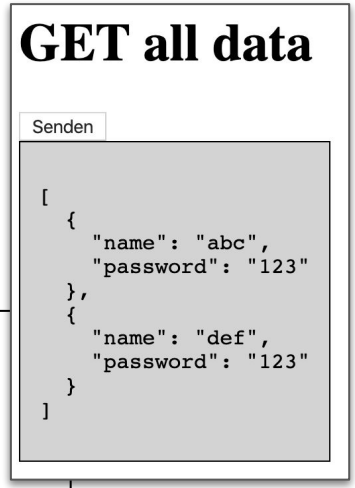
```php
<?PHP

 // Allow cross origin = CORS
 if (isset($_SERVER['HTTP_ORIGIN'])) {
    header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
    header('Access-Control-Allow-Credentials: true');
    header('Access-Control-Max-Age: 86400');    // cache for 1 day
 }
 // Access-Control headers are received during OPTIONS requests
 if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {

    if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
        header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

    if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
        header("Access-Control-Allow-Headers: {$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");
 }

?>
```

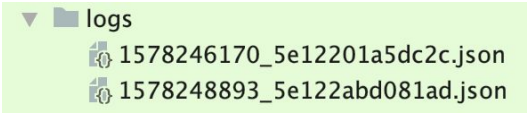


Mit fetch alle Daten vom Server holen

```
<h1>GET all data</h1>
<input type="submit" id="submit">
<div id="result"></div>
```

```
<script>
  const submit=document.getElementById('submit');
  submit.addEventListener('click', async function( event ){
    const dataset = (await (await fetch( new Request( "./server.php" ), {
      method: 'GET',
      mode: 'cors',
      cache: 'no-store'
    })).json());
    document.getElementById('result').innerHTML = `<pre><code>${JSON.stringify(dataset,null,2)}</code></pre>`;
  });
</script>
```





https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/fetch