

Handbuch

im Fach

Webprogrammierung

SoSe 2021

Stand 01/03/2021

Inhaltsverzeichnis

1	Abbildungsverzeichnis	6
2	Grundlagen HTML	7
2.1	Zeilenumbrüche	7
2.2	Überschriften	7
2.3	Listen	8
2.3.1	Besonderheiten	10
2.4	Tabellen	10
2.5	CSS	11
2.5.1	Messeinheiten	11
2.5.2	Farbdefinitionen	11
2.5.3	Farbnamen:	12
2.5.4	Sonderzeichen	12
2.5.5	div	13
2.5.6	span	14
2.5.7	rahmen	15
2.5.8	Pseudo-Elemente	16
2.5.9	Padding und Margin	16
3	PHP	18
3.1	Was ist PHP?	18
3.2	Überblick über eine HTML-Abfrage	18
3.3	PHP-Abfrage	19
3.4	Eigenschaften von PHP in HTML	19
3.5	Erste Beispiel	19
4	PHP Grundlagen	21
4.1	Variablen	21
4.2	Einfache Datentypen (implizit)	21
4.2.1	Variablen Beispiele	21
4.3	Operatoren	21
4.3.1	Mathematische Ausdrücke:	21
4.3.2	Logische Verknüpfungen:	22
4.4	Abfragen	22
4.4.1	If, then, else	22
4.4.2	switch	22
4.5	Schleifen	23
4.5.1	For-Schleife	23
4.5.2	While-Schleife	23
4.5.3	Do-While-Schleife	23
4.5.4	Foreach-Schleife	24
4.6	Schleife à la Knotenliste	24
4.6.1	Beispiel	25
4.6.2	Ausgabe:	25
4.7	Ausgabe mit echo	26
4.8	Arrays	27
4.8.1	Einfache Arrays	27
4.8.2	Funktionen für Arrays	30

4.8.3	Multidimensionale Arrays	31
4.8.4	Assoziierte Arrays (HashTables)	33
4.9	Funktionen	35
4.10	String-Methoden	37
4.10.1	localeconv	49
4.10.2	nl_langinfo	50
4.10.3	Ländercode	52
4.11	Klassen	56
4.11.1	Defintion	56
4.11.2	Abstrakte Klassen	57
4.11.3	Interface	57
4.11.4	Einfaches Beispiel	57
4.11.5	Beispiel mit Konstruktor	58
4.11.6	Beispiel mit abgeleiteter Klasse	58
4.11.7	Eintragen in eine Liste (Beispiel Wertetabelle)	60
5	Formulare	62
5.1	Überblick	62
5.2	Formular-Struktur	62
5.2.1	Beispiel 1	63
5.3	GET / SET	64
5.3.1	Get	64
5.3.2	Post	64
5.4	Parameter der Datenübertragung	64
5.5	Formular-Elemente	65
5.5.1	HTML4-Elemente	65
5.5.2	Neue HTML5-Elemente	65
5.5.3	label	66
5.5.4	Text	66
5.5.5	Password	67
5.5.6	Checkbox	67
5.5.7	Radiobutton / Auswahlshalter	67
5.5.8	select	68
5.5.9	textarea	69
5.5.10	number	69
5.5.11	hidden	69
5.5.12	image	70
5.5.13	submit	70
5.5.14	reset	71
5.5.15	button	71
5.5.16	file	72
5.6	Sessions	72
5.6.1	Sessions-Abfrage auf Inhalt	72
5.6.2	Sessions-Variable setzen	73
5.6.3	Beispielprogramm	73
5.7	PHP-Formulare-Beispiele	74
5.7.1	Anzeige einer Liste von bis	75
5.7.2	Berechnen einer Addition oder einer Subtraktion	78
5.7.3	Eingabe mit einer Liste (Auswahl mehrerer Elemente)	81
5.7.4	Eingabe mit einer Liste von Checkboxes(Auswahl mehrerer Elemente)	84
6	Pattern	88
6.1	Beispiel Nachname	88
6.2	Beispiel Straße	89

6.3	Beispiel Geburtsdatum	89
6.4	Beispiel E-Mail	89
6.5	Beispiel Drei Großbuchstaben, dann drei Ziffern	89
6.5.1	Beispiel IPv4	89
6.6	Beispiel Paßwort:	90
7	Abfragen der Parameter in PHP	91
7.1	Abfrage eines String Wertes	91
7.2	Abfrage eines numerischen Wertes (int)	91
7.3	Abfrage eines numerischen Wertes (float)	92
8	Datenbank MySQL	94
8.1	Datenbank Abfrage	94
8.2	Initialisierung	94
8.3	Abfrage mit query	95
8.3.1	Quellcode	95
8.3.2	Connection	96
8.3.3	Sonderzeichen	96
8.3.4	Abfrage-Schleife	96
8.4	Sequence	97
8.5	Insert into	98
8.6	Update	98
8.7	Delete	99
8.8	Kompletter Rahmen für eigene Datenbanken	99
8.8.1	SQL-Datenbank	99
8.8.2	Verwendete Klasse:	100
8.8.3	Rahmen	100
9	JavaScript	104
9.1	Allgemeine Eigenschaften	104
9.2	Spracheigenschaften	104
9.3	Abschnitt definieren	104
9.4	Datentypen	105
9.4.1	Methoden	105
9.5	Arrays	105
9.5.1	Anlegen eines Arrays	105
9.5.2	Anzahl eines Arrays	105
9.5.3	Eintrag hinzufügen	105
9.5.4	Eintrag löschen	105
9.5.5	slice	105
9.5.6	valueOf	106
9.5.7	Mehrdimensionales Array	106
9.6	Operatoren	106
9.6.1	Mathematische Ausdrücke:	106
9.6.2	Logische Verknüpfungen:	106
9.7	Abfragen	106
9.7.1	If, then, else	106
9.7.2	switch	107
9.8	Schleifen	108

9.8.1	For-Schleife	108
9.8.2	While-Schleife	108
9.8.3	Do-While-Schleife	108
9.8.4	Foreach-Schleife	109
9.9	Funktionen	109
9.10	String-Methoden	110
9.11	Klassen	111
9.11.1	Klasse deklarieren	111
9.11.2	Vererbung	112
10	Ajax und JSON	113
10.1	Struktur von Ajax	113
10.2	Ajax-Struktur als Quellcode	114
10.3	Status-Code bei Ajax	115
10.4	JSON	122
10.4.1	Aufbau von JSon	122
10.4.2	JSon-Beispiele:	122
10.4.3	PHP und JSon	123
10.4.4	Javascript und JSon	123
10.5	Beispiele	124
10.5.1	Summe zweier Zahlen	124
10.5.2	Taschenrechner mit zwei Zahlen	128
10.5.3	Liste mit Zahlen	134
10.6	Probleme von Ajax	138
11	XAMMP-Installation	139
11.1	Windows	139
11.2	Starten vonm XAMPP	142
11.3	Port-Probleme	143
11.4	Apple	144
12	HTML-Editoren	150
12.1	HTML (Windows)	150
12.2	HTMLRtf (Windows)	150
12.2.1	iHTML (iOS, Linux)	150
12.3	iHTMLRtf (iOS, Linux)	150
12.4	iHTMLTab (iOS, Linux)	150
13	Stichwortverzeichnis	151

1 Abbildungsverzeichnis

Abbildung 1	Absätze in HTML (bsp05-Absatz.html)	7
Abbildung 2	Element header (bsp04-Header.html)	8
Abbildung 3	Listenbeispiel	9
Abbildung 4	Liste mit start und value-Attribut (bsp11-Listen.html).....	10
Abbildung 5	div-Beispiele mit horizontaler Ausrichtung (bsp21-div.xhtmll).....	14
Abbildung 6	Beispiel für das span-Element (bsp22-span.xhtmll)	15
Abbildung 7	Rahmen-Beispiel (bsp23-rahmen.xhtmll).....	16
Abbildung 8	Padding und Margin	17
Abbildung 9	3.2 Überblick über eine Anfrage einer HTML-Seite.....	18
Abbildung 10	Fehlerhafte Indizes.....	28
Abbildung 11	Ausgabe mit einer foreach-Schleife.....	28
Abbildung 12	Ausgabe mit einer for und foreach-Schleife	29
Abbildung 13	Anzeige eines zweidimensionalen Arrays	33
Abbildung 14	Wertetabelle mit Klassen	61
Abbildung 15	Wertetabelle mit Klassen und JSON	61
Abbildung 16	Überblick über den Verlauf einer Server-Anfrage	62
Abbildung 17	Erstes Beispiel eines einfachen Formulars	63
Abbildung 18	Erstes Beispiel eines einfachen Formulars (Serverseite).....	64
Abbildung 19	HTML-Element select (ComboBox)	68
Abbildung 20	HTML-Element select (Liste).....	69
Abbildung 21	Image mit einem Link.....	70
Abbildung 22	Anzeige des file-HTML-Elements	72
Abbildung 23	Anzeige der ausgewählten Datei	72
Abbildung 24	Struktur von Ajax	113
Abbildung 25	Erstes JAX-JSON-Beispiel.....	124
Abbildung 26	Zweites AJAX-JSON-Beispiel	128
Abbildung 27	Drittes AJAX-JSON-Beispiel.....	134
Abbildung 28	XAMMP-Downloadseite	139
Abbildung 29	Starten des Downloads	139
Abbildung 30	Frage nach dem AntiVirusprogramm	139
Abbildung 31	Rechteproblematik	140
Abbildung 32	Echtes Startfenster	140
Abbildung 33	Auswahl der Komponenten	141
Abbildung 34	Installationspfad.....	141
Abbildung 35	Auswahl im XAMPP-Dialog.....	142
Abbildung 36	Erfolgreiche Auswahl im XAMPP-Dialog.....	142
Abbildung 37	Eintragen des Ports für PHP	143
Abbildung 38	XAMMP-Downloadseite	144
Abbildung 39	Mit Drag&Drop XAMMP nach Application ziehen	144
Abbildung 40	Startdialog von XAMPP	145
Abbildung 41	IP-Adresse: 192.168.64.2	145
Abbildung 42	Port 8080 freischalten	147
Abbildung 43	Wechseln zum Register Volumes	147
Abbildung 44	Anklicken des Schalters "Mount"	148

2 Grundlagen HTML

Dieses Kapitels zeigt eine Übersicht über die wichtigsten HTML-Elemente und Eigenschaften.

2.1 Zeilenumbrüche

Mit „`
`“ wird nur ein Zeilenumbruch erzeugt. Mit der Kombination „`<p>`“ und „`</p>`“ wird jeweils eine Leerzeile vor und hinter dem Element eingefügt.

```
<html>
  <head>
    <title> Absatzdefinition </title>
  </head>
<body>
  Dies ist die erste Zeile des Tests, &#60;br /&#62;
  <br />
  Dies ist die zweite Zeile des Tests,&#60;br /&#62;
  <br />
  <p>
    Dies ist die dritte Zeile des Tests,&#60;p&#62;
  </p>
  Dies ist die vierte Zeile des Tests
</body>

</html>
```

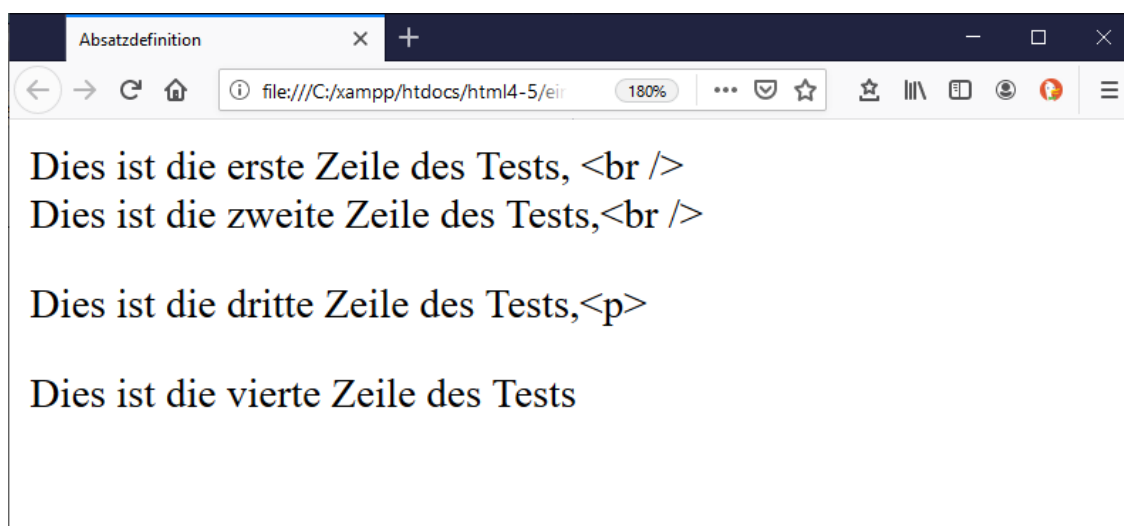


Abbildung 1 Absätze in HTML (bsp05-Absatz.html)

2.2 Überschriften

Insgesamt gibt es sechs Überschriften. Wie in einem Buch, sind die Hauptkapitel 1, 2, 3 etc. größer als die Kapitel der zweiten Stufe, 1.1, 1.2, 3.4).

```

<h1>Text der 1.  &#220;berschrift</h1>
<h2>Text der 2.  &#220;berschrift</h2>
<h3>Text der 3.  &#220;berschrift</h3>
<h4>Text der 4.  &#220;berschrift</h4>
<h5>Text der 5.  &#220;berschrift</h5>
<h6>Text der 6.  &#220;berschrift</h6>
HTML ist toll<br />

```

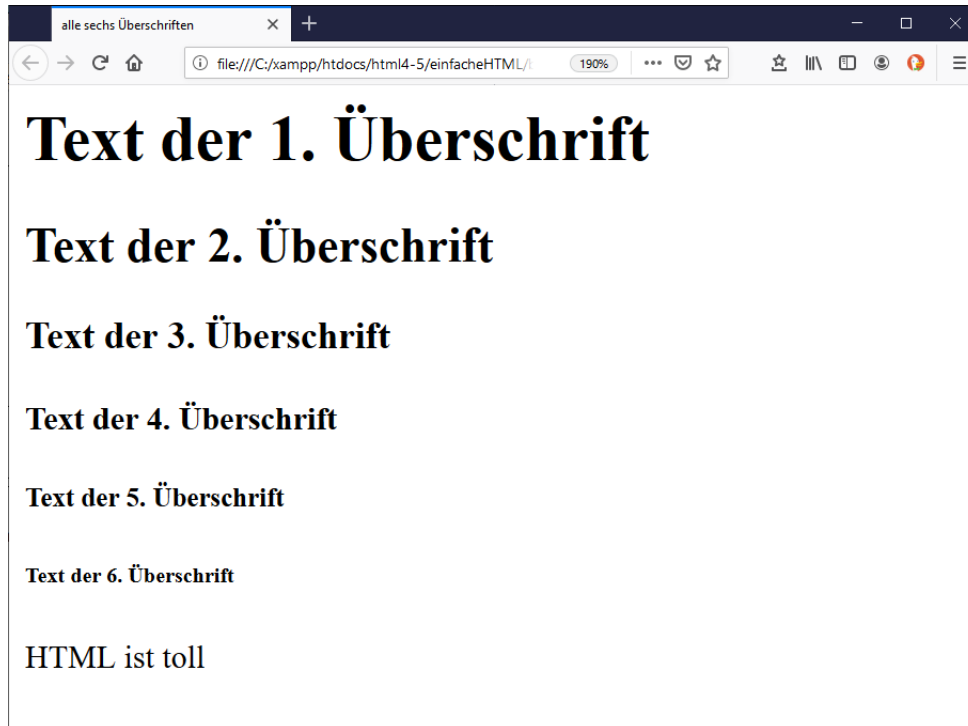


Abbildung 2 Element header (bsp04-Header.html)

2.3 Listen

Ein Autohaus braucht ein Programm zur Verwaltung der Fahrzeugdaten. Ein Student des ersten Semesters soll das programmieren.

- ul (unordered list)
 - none Kein Aufzählungszeichen
 - Circle Kreis, nur Rahmen
 - square Quadrat
 - disc Gefüllter Kreis
- ol (ordered list)
 - none Kein Aufzählungszeichen
 - decimal Dezimalzahlen (1. ,2. , 3. , ...)
 - lower-roman Kleine römische Zahlen (i. ,ii. ,iii. , ...)
 - upper-roman Grosse römische Zahlen (I. ,II. , III. , ...)
 - decimal-leading-zero Dezimalzahlen mit führender 0 (01. ,02. , 03. , ...)
 - lower-greek Kleine griechische Nummerierung alpha, beta, gamma,...
 - lower-latin Kleine Ascii-Zeichen (a. ,b. , c. , ...)
 - upper-latin Große Ascii-Zeichen (A. , B. ,C. , ...)
 - armenian Armenische Nummerierung
 - georgian Georgische Nummerierung

Quellcode

HTML


```

<h3> Liste mit Punkten </h3>
<!-- circle, disc, square -->
<ul type="disc" >
  <li> Text1aaaa </li>
  <li> Text2 </li>
</ul>

<ul type="circle">
  <li>Punkt 1 </li>
  <li>Punkt 2</li>
  <li>Punkt3</li>
  <li>Punkt 4</li>
</ul>

<h3> Liste mit Nummern </h3>
<ol type="1">
  <li>Punkt 1</li>
  <li>Punkt 2</li>
  <li>Punkt 3</li>
  <li>Punkt 4</li>
</ol>

<h3> Liste mit Römischen
Zahlen</h3>
<ol Type="I">
  <li>Punkt 1</li>
  <li>Punkt 2</li>
  <li>Punkt 3</li>
  <li>Punkt 4</li>
</ol>

<h3> Liste mit Römischen
Zahlen</h3>
<ol Type="I">
  <li>Punkt 1</li>
  <li>Punkt 2
    <ol type="a" start="100" >
      <li>Punkt 1</li>
      <li>Punkt 2 m hier ist eine
Testnachricht</li>
      <li>Punkt 3</li>
      <li>Punkt 4</li>
    </ol>
  </li>
  <li>Punkt 3</li>
  <li>Punkt 4</li>
</ol>

<h3>Liste mit Buchstaben </h3>
<ol type="A">
  <li>Punkt 1</li>
  <li>Punkt 2</li>
  <li>Punkt 3</li>
  <li>Punkt 4</li>
</ol>

```

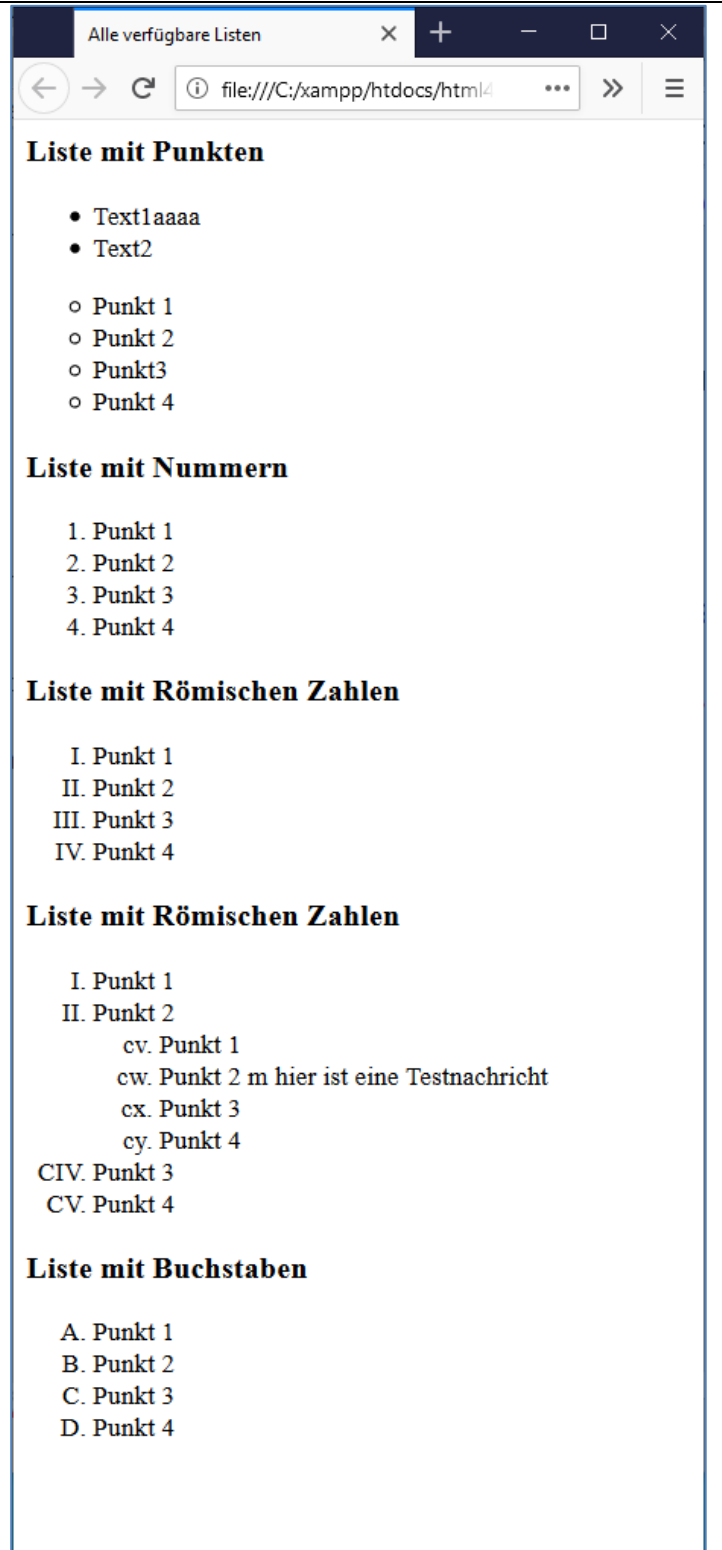


Abbildung 3 Listenbeispiel

2.3.1 Besonderheiten

start:

- Mit dem „start“-Attribut kann man die Anfangsnummerierung ändern.

value:

- Mit dem „value“-Attribut kann man die aktuelle Nummerierung ändern. Alle weitere Nummerierung beziehen sich dann auf die Nummerierung.

Beispiel:

```
<ol style="list-style-type:decimal;" start="4">
  <li>Punkt 1</li>
  <li>Punkt 2</li>
  <li value="8">Punkt 3</li>
  <li>Punkt 4</li>
  <li>Punkt 14</li>
  <li>Punkt 24</li>
</ol>
```

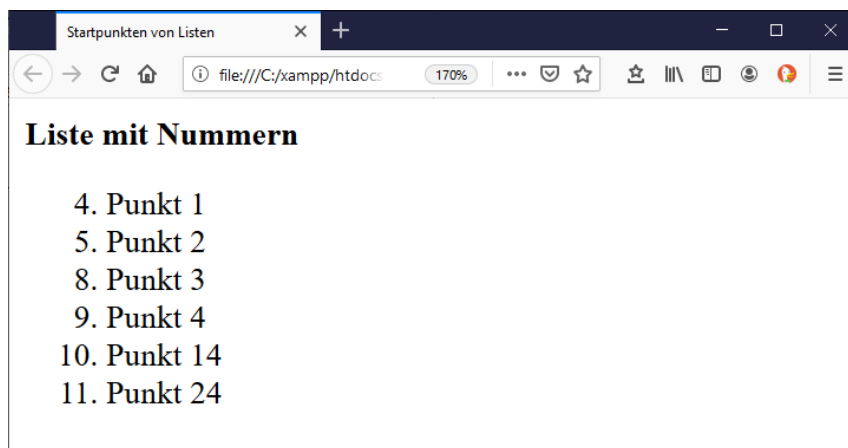


Abbildung 4 Liste mit start und value-Attribut (bsp11-Listen.html)

Liste in Liste:

- Eine Liste in einer Liste wird zwischen einem öffnenden und schließenden -Tag eingefügt.

```
<li>Punkt 2
  <ol>
    <li>Punkt 1</li>
    <li>Punkt 2 m hier ist eine Testnachricht</li>
    <li>Punkt 3</li>
    <li>Punkt 4</li>
  </ol>
</li>
```

2.4 Tabellen

Ein Autohaus braucht

2.5 CSS

- Ergänzung zu HTML
- Definition von Formateigenschaften von HTML-Elemente (Styles)
 - Überschrift
 - Abstand eines Absatzes
 - Tabellenkopf
 - Eigene Hintergrundfarbe pro Absatz
 - Eigener Rahmen
 - Einrückungen
 - Margin und Padding (Randabstände)
 - Border (Ränder)
 - Umfangreiche Styles und Maßeinheiten
 - Definition von Styles mittels Ort (Cascade)
 - Definition von Absätzen, auch übereinander (Sheets)
 - Akustische Wiedergabe von Texten (SMIL)
- Adresse: <http://de.selfhtml.org/css/intro.htm>
- <http://de.selfhtml.org/css/eigenschaften/index.htm>
- <http://www.w3.org/TR/REC-CSS2/>

2.5.1 Messeinheiten

- in inch
- cm centimeter
- mm millimeter
- em Höhe des aktuellen Fonts
- rem Höhe des aktuellen Fonts (root)
- ex Höhe des Buchstaben „x“ des aktuellen Fonts
- pt point, 1/72 in, entspricht 0,35278 mm
- pc pica, entspricht 12 points: 4,23 mm
- px Pixel

2.5.2 Farbdefinitionen

- #rrggbb Hexadezimale Darstellung
- rgb(x,y,z) Farbanteil pro Farbe, jeweils 0 bis 255
- rgb(x%,y%,z%) Farbanteil in Prozent pro Farbe, jeweils 0 bis 100%

Beispiele:

- #FF0000 rot
- #00FF00 grün
- #0000FF blau
- #FFFF00 gelb
- #FF00FF violett
- #00FFFF türkis
- #CCCCCC grau

2.5.3 Farbnamen:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- olive
- purple
- red
- silver
- teal
- white
- yellow

2.5.4 Sonderzeichen

- | | |
|-----------|------------------------------|
| • | Leerzeichen, |
| •   | Leerzeichen Breite n |
| •   | Leerzeichen Breite m |
| •   | Leerzeichen, schmal |
| • < | < |
| • > | > |
| • ≤ | <= |
| • ≥ | >= |
| • £ | £ Pfundzeichen |
| • € | € Eurozeichen |
| • ¢ | ¢ Centzeichen |
| • § | § Paragraphenzeichen |
| • © | © Copyright |
| • ® | ® eingetragene Marke |
| • ™ | ™ Trademark-Zeichen |
| • ³ | ³ hochgestellte 3 |
| • ² | ² hochgestellte 2 |
| • ¹ | ¹ hochgestellte 1 |
| • ½ | ½ |
| • ¼ | ¼ |
| • ¾ | ¾ |
| • μ | μ Microzeichen |
| • α | α alpha |
| • β | β beta |
| • λ | λ lambda (z.B. Wellenlänge) |
| • ω | ω omega (z.B. Kreisfrequenz) |
| • Ω | Ω Omega (z.B. Widerstand) |
| • π | π Kreiszahl |

- `Πi` Π
- `±` \pm plusminus
- `­` Man. Trennung: shy
- `¶` ¶ Absatz-Zeichen

- `ä` ä deutsche Umlaute
- `Ä` Ä
- `ö` ö
- `Ö` Ö
- `ü` ü
- `Ü` Ü
- `ß` ß

2.5.5 div

Das `<div>`-Tag ist ein Container für mehrere HTML-Elemente, denen durch die Kernattribute des `<div>`-Tags Stylesheet-Eigenschaften zugewiesen werden. `div`-Elemente sind Blockelemente, da das öffnende und das schließende `<div>`-Tag jeweils zu Zeilenumbrüchen – äquivalent zum `
`-Tag – führen. `<div>`-Tags lassen sich ineinander verschachteln und bilden einen leistungsfähigen Mechanismus. Man verwendet das `<div>`-Tag dazu, eine Menge von logisch zusammengehörigen HTML-Elementen mit der Hilfe von Cascading Stylesheets zu formatieren, positionieren oder mit JavaScript zu animieren. Dazu werden die Kernattribute `class`, `id` und `style` benutzt. Äquivalent zum `<div>`-Tag gibt es das ``-Tag, das benutzt wird, wenn einer Gruppe von HTML-Elementen Inlinestile – also Stile ohne Zeilenumbruch – zugewiesen werden sollen.

Beispiel:

```
<body>
  <h3> Die Texte k&ouml;nnen mit dem Attribut ALIGN innerhalb eines
  Blockes ausgerichtet werden. </h3>

  <div>
    Dieser Text wird linksb&uuml;ndig ausgegeben, da dieses die
    Defaulteinstellung ist.
  </div>
  <br />
  <div class="right">
    Die Texte k&ouml;nnen mit dem Attribut "text-align:right" innerhalb
    eines DIV-Blockes ausgerichtet werden.
  </div>
  <br />

  <div class="center">
    Dieser Text wird zentriert ausgegeben.
  </div>
</body>
```

CSS:

```
.center {
  text-align:center;
}

.left{
  text-align:left;
}

.right {
  text-align:right;
}
```



Abbildung 5 div-Beispiele mit horizontaler Ausrichtung (bsp21-div.xhtml)

2.5.6 span

<h3> Beispiel mit einem span-Abschnitt </h3>

<p>

Diese Zeile sollte in grün angezeigt werden

</p>

<div class="div1">

3. Zeile, im zweiten Beispiel, div

</div>



Abbildung 6 Beispiel für das span-Element (bsp22-span.xhtml)

Das Wort „zweiten“ hat das span-Element mit dem CSS-Wert „text-decoration: underline“.

2.5.7 rahmen

Ein Style kann man nun in die CSS-Datei eintragen, um spezielle Formate zu definieren, ohne in die HTML-Datei weitere Attribute einzufügen:

HTML:

```
<body>
  <h3>Rahmen</h3>
  <h1 class="rahmen">Eine h1-Überschrift mit Rahmen</h1>
  <h2 class="rahmen">Eine h2-Überschrift, allgm. Regel</h2>
  <p class="rahmen">Ein Absatz, p-Regel</p>
  <div class="rahmen">Ein Absatz, allgm. Regel</div>
</body>
```

CSS:

```
/* allgemeine Regel */
.rahmen {
  border:1px solid green;
  background-color:green;
}

/* gilt nur in einem p-Abschnitt */
p.rahmen {
  border:1px solid red;
  background-color:red;
}

/* gilt nur in einem h1-Abschnitt */
h1.rahmen {
  border:1px solid blue;
  background-color:blue;
}
```



Abbildung 7 Rahmen-Beispiel (bsp23-raahmen.xhtml)

2.5.8 Pseudo-Elemente

- :after Gilt nach einem Element
- :before Gilt Vor einem Element
- :first-letter Das erste Zeichen in einer Zeile
- :first-line Die erste Zeile in einem Absatz

- :first-child Das erste "Kind" des Abschnittes (body,ol)
- :last-child Das letzte "Kind" des Abschnittes (body,ol)
- :active benutzt in Hyperlink
- :focus wenn Element den Fokus erhält
- :hover Maus über Elemente, (Hyperlink)
- :lang Sprache
- :link Normaler Link (Hyperlink)
- :visited Besuchter Link

2.5.9 Padding und Margin

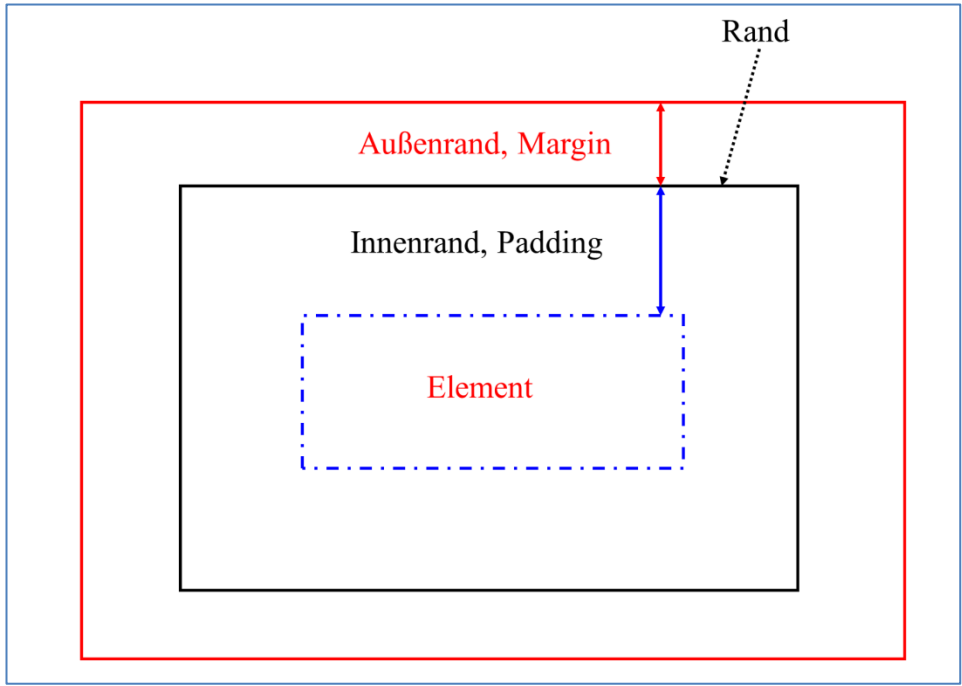


Abbildung 8 Padding und Margin

3 PHP

3.1 Was ist PHP?

- PHP: Hypertext Preprocessor
- Serverseitige Skriptsprache
- Syntax ähnlich C, Perl, Java, ...
- Eingebettet in HTML.
 - `<?php code ?>`
- Ermöglicht dynamische Webseiten.
- Anbindung an Datenbanken (z.B. MySQL)
- hat globale, statische und dynamische Variable
- hat Konstanten und Funktionen
- hat Klassen
- Die Kapselung geschieht durch include-Anweisungen
 - `include("meineDatei.php");`

3.2 Überblick über eine HTML-Abfrage

Clientseitige vs. Serverseitige Skriptsprache

– Clientseitig
(beispielsweise
Javascript):

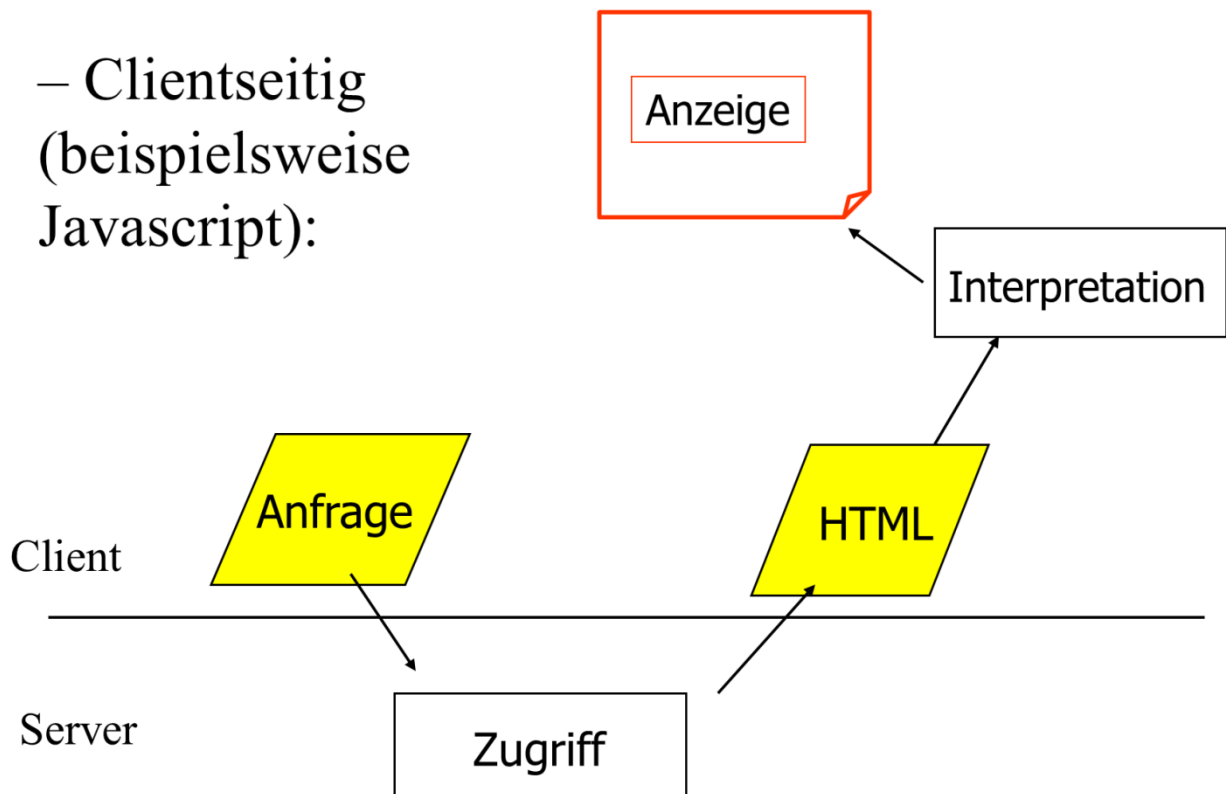
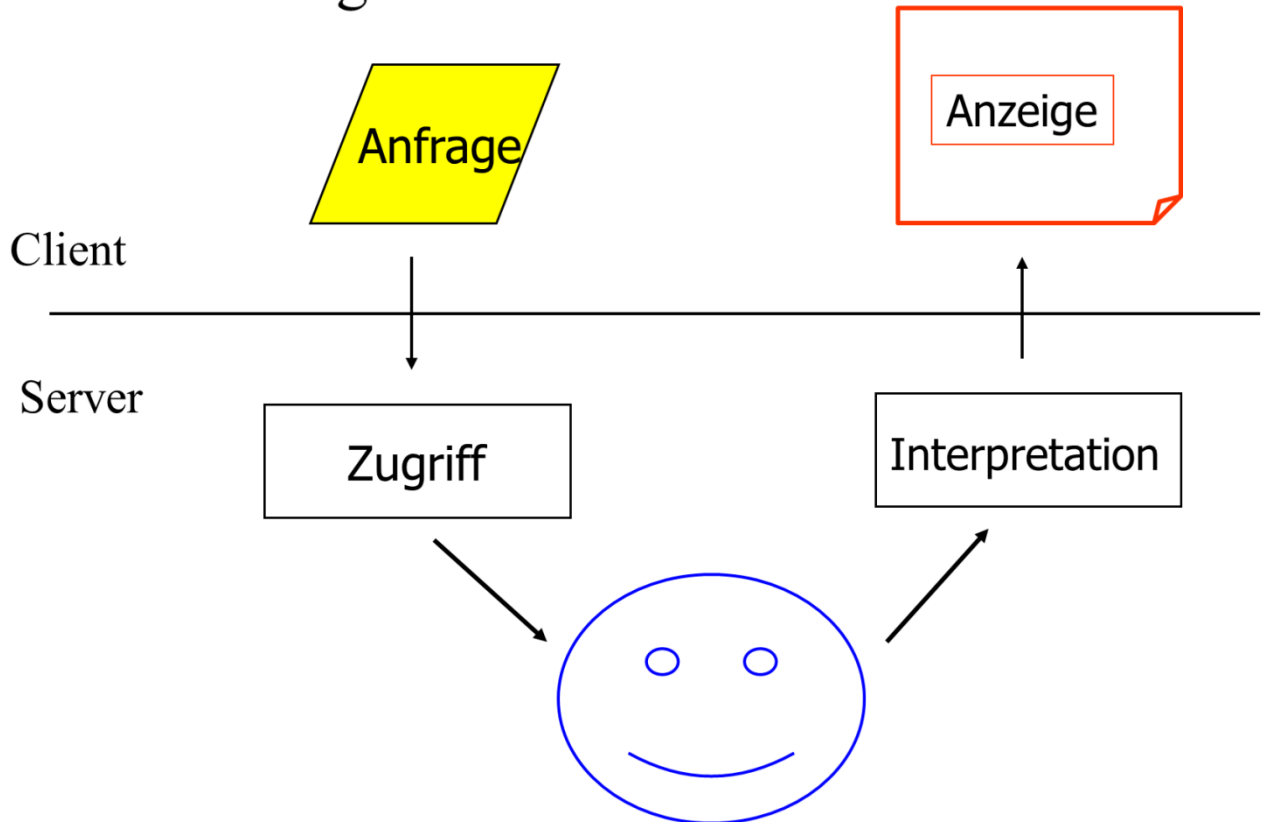


Abbildung 9 3.2 Überblick über eine Anfrage einer HTML-Seite

3.3 PHP-Abfrage

Clientseitige vs. Serverseitige Skriptsprache

– Serverseitig



3.4 Eigenschaften von PHP in HTML

- PHP-Code wird durch einen Interpreter auf dem Server ausgeführt.
- Variablen
- Kontrollstrukturen (if, case)
- Schleifen (for, while)
- Datentypen (Integer, Double, Bool, Strings)
- Arrays (einfache, multidimensionale, indizierte, assoziierte)
- Funktionen
- Bild-Erstellung
- OOP, ab Version 5.0
- Anbindung an eine Datenbank
- Dateioperationen
- Dateiendungen (php, phtml)

3.5 Erste Beispiel

```
<html>
<head>
  <title> Titel des Dokuments </title>
```

```
</head>
<body>
  <!-- http://mwilhelm.hs-harz.de/scripte/php/bsp01.php -->
  <h2> Erste PHP-Datei </h2>
  <?php
    echo("hallo Text");
  ?>
  <p>
    HTML-Kern des Dokuments
  </p>
</body>
</html>
```

Der PHP-Interpreter übersetzt den Befehl und fügt „hallo Text“ ein den HTML-Code ein.

4 PHP Grundlagen

4.1 Variablen

- Man muss und kann keine Datentypen deklarieren.
- Aus den Werten erschließt sich PHP den Datentyp.

4.2 Einfache Datentypen (implizit)

- Boolean (Werte: TRUE und FALSE)
- Integer (Ganze Zahlen)
- Float, Double (Fließkommazahlen)
- String (Zeichenketten)

4.2.1 Variablen Beispiele

- Speicherplatz für einen Wert
 - Kennzeichnung: "\$" + Name
 - Beispiel: \$vorname
 - Groß- / Kleinschreibung wird unterschieden
 - Zuweisung: \$i = 10;
 - Ausgabe: echo \$i;
 - \$a = 5; // legt einen Integer an, gettype() = int
 - \$a = 5.2; // legt eine Fließkommazahl an, gettype =double
 - \$a = "Heute ist Montag"; // gettype()=string
-
- <?php
 - \$zahl = 5;
 - \$str = "3.1415";
 - \$bool = true;
 - \$zahl = 5;
 - echo gettype(\$zahl);
 - ?>

4.3 Operatoren

Die Operatoren in PHP sind sehr ähnlich zu C, Perl oder Java

4.3.1 Mathematische Ausdrücke:

.	+	-	*	/	%	&		~		
+=	.	-=	*=	/=	%=	&=	=			
<	<=	==	===	=>	>	!=				
\$v++		++\$v		\$v--		--\$v				

4.3.2 Logische Verknüpfungen:

&&	And
	Or

"===" Identität

4.4 Abfragen

4.4.1 If, then, else

```
if (Bedingung) {  
    Anweisung(en)  
}  
else {  
    Anweisung(en)  
}  
  
if (Bedingung)  
    Anweisung1;  
else  
    Anweisung2;
```

Beispiel:

```
$zahl=5;  
if($zahl>4) {  
    echo "Größer als 4";  
}  
else {  
    echo "Kleiner gleich 4";  
}
```

4.4.2 switch

```
switch(Variable){  
    case Wert:  
        Anweisung(en);  
        break;  
    ...  
    case Wert:  
        Anweisungen;  
        break;  
    default:  
        Anweisungen  
}
```

Beispiel:

```
$ort="Stuttgart";  
switch($ort){
```

```

    case "Bottrop":
        echo "nicht schön!";
        break;
    case "Stuttgart":
        echo "sehr schön!";
        break;
    default:
        echo "naja!";
}

```

4.5 Schleifen

4.5.1 For-Schleife

```

for( Initialisierung(en); Bedingung; Anweisung(en) )
{
    Anweisung(en)
}

```

Beispiel:

```

for ($i=1; $i<100; $i++) {
    $k = $i * $i;
    echo "$i zum Quadrat ist $k <br />\n";
}

```

4.5.2 While-Schleife

Struktur:

```

while (Bedingung) {
    Anweisung(en)
}

```

Beispiel:

```

$i=1;
$k=1;
while($k<=100) {
    echo "$i zum Quadrat ist $k <br />";
    $i++;
    $k = $i * $i;
}

```

Beachten Sie auch die „break“ und „continue“-Anweisungen.

4.5.3 Do-While-Schleife

```

do {
    Anweisung(en)
}
while (Bedingung)

```

Beispiel:

```
$i=$k=1;
do {
    $k=$i*$i;
    echo "$i zum Quadrat ist $k<br />";
    $i++;
} while ($k<100);
```

Unterschied zur while-Schleife:

Sie wird mindestens einmal durchlaufen.

Beachten Sie auch die „break“ und „continue“-Anweisungen.

4.5.4 Foreach-Schleife

```
foreach(Array-Variable as Variable){
    Anweisungen
}
```

In den Anweisungen ist jeweils Zugriff auf die „Variable“ möglich

Beispiel:

```
$farbenliste=array('blau','rot','gelb');
foreach($farbenliste as $einzelfarbe){
    echo "Die Farbe ist: $einzelfarbe";
}
```

Für assoziative Arrays (Ausgabe Schlüssel möglich):

```
$farben = array('f1'=>'blau','f2'=>'rot','f3'=>'gelb');
foreach($farben as $schluessel=>$farbe) {
    echo "Farbe mit Schlüssel $schluessel ist:
    $farbe<br />";
}
```

4.6 Schleife à la Knotenliste

Es gibt in PHP auch die Möglichkeit, ein Array stückweise abzuarbeiten. Dazu gibt es folgende Funktionen:

- `current()`
 - Rückgabe des aktuellen Elementes ODER False
- `end()`
 - Ausgabe des letzten Elementes oder False, falls die Liste leer ist.
- `next()`
 - Setzt den internen Zeiger auf das nächste Element und gibt dieses aus.
- `prev()`
 - Setzt den internen Zeiger auf das vorherige Element und gibt dieses aus.
- `reset()`

- Setzt den internen Zeiger auf den Anfang und gibt das erste Element aus.
- each()
 - Gibt das aktuelle Element aus und wechselt zum nächsten Element ohne dieses auszugeben.

4.6.1 Beispiel

```
echo '1. Listenausgabe' . '<br />';
echo 'Anzahl: ' . count($liste) . '<br />';
foreach ($liste as $item) {
    echo $item . '<br />';
}
echo '<br />';
echo '<br />';
echo '2. Listenausgabe' . '<br />';
reset($liste);
echo 'Anzahl: ' . count($liste) . '<br />';

echo current($liste) . '<br />';
echo 'next1: ' . next($liste) . '<br />';
echo 'next2: ' . next($liste) . '<br />';
echo 'next3: ' . getType(next($liste)) . '<br />';

echo '<br />';
echo '<br />';
echo '3. Listenausgabe' . '<br />';
echo 'Anzahl: ' . count($liste) . '<br />';
reset($liste);
while (current($liste)) {
    echo '3. next: ' . current($liste) . '<br />';
    next($liste);
}

echo '<br />';
echo '<br />';
echo '4. Listenausgabe' . '<br />';
echo 'Anzahl: ' . count($liste) . '<br />';
echo '4. next: ' . reset($liste) . '<br />';
while ($item = next($liste)) {
    echo '4. next: ' . $item . '<br />';
}
```

4.6.2 Ausgabe:

Reset Array

1. Listenausgabe

Anzahl: 3

red

blue

green

2. Listenausgabe

Anzahl: 3

next1: red

```
next2: blue
next3: green
next4: boolean           // Ausgabe getType des letzten Elemente, hier false
```

3. Listenausgabe

```
Anzahl: 3
3. next: red
3. next: blue
3. next: green
```

4. Listenausgabe

```
Anzahl: 3
4. next: red
4. next: blue
4. next: green
```

4.7 Ausgabe mit echo

Zeichenketten werden nicht mit dem Pluszeichen, sondern mit dem Punkt verbunden.

Stringoperatoren zum Verbinden von Strings:

```
$a="Hallo " . "Welt!";
$a="Hallo "; $a .= "Welt";
```

Bei den Apostrophen gibt es zwei Varianten:

- Das einfache Apostroph
 - '
 - Variablen werden nicht durch ihren Wert ersetzt.
- Das doppelte Apostroph
 - ''
 - Variablen werden durch ihren Wert ersetzt.

Beispiel:

```
$b = 123
$a= "Dies ist der Wert b: $b";
    Ersetzung aller erkannten Variablen durch ihre Werte
Ausgabe: Dies ist der Wert b: 123

$a= 'Dies ist der Wert b: $b';
    Keine Ersetzung der Variablen durch ihre Werte
Ausgabe: Dies ist der Wert b: $b
```

Alternative:

```
$a= 'Dies ist der Wert b: ' . $b;
Ausgabe: Dies ist der Wert b: 123
```

4.8 Arrays

Eine Array-Variable enthält Speicherplatz für eine Liste von Werten. Dabei können in einem Array beliebige Datentypen gespeichert werden.

4.8.1 Einfache Arrays

Gespeichert und gelesen werden die Daten über einen Index à la Java, C#.

Besonderheiten:

- Es gibt keine Initialisierungsgrenze.
- Es kann Lücken geben.

4.8.1.1 Beispiel 1

```
$feld[0] = 7;  
$feld [3] = "Hallo";  
$feld[5] = true;  
$feld[] = "erstmal";  
echo $feld[6];
```

Im obigen Beispiel werden drei verschiedene Datentypen (integer, String, Boolean) eingefügt. Es gibt **vier** gültige Indizes:

- 0
- 3
- 5
- 6

Abfragen:

Eine normale for-Schleife ist für das obige Beispiel nicht möglich, da es Lücken enthält.

Abfrage mit einer For-Schleife.

```
echo 'Anzahl: ' . count($feld) . '<br />';  
echo $feld[6] . '<br />';  
  
for ($i=0; $i<count($feld); $i++) {  
    echo 'Index: ' . $i . ' ' . $feld[$i] . '<br />';  
}
```

Die Anzahl ist VIER, aber der höchste Index ist 6.

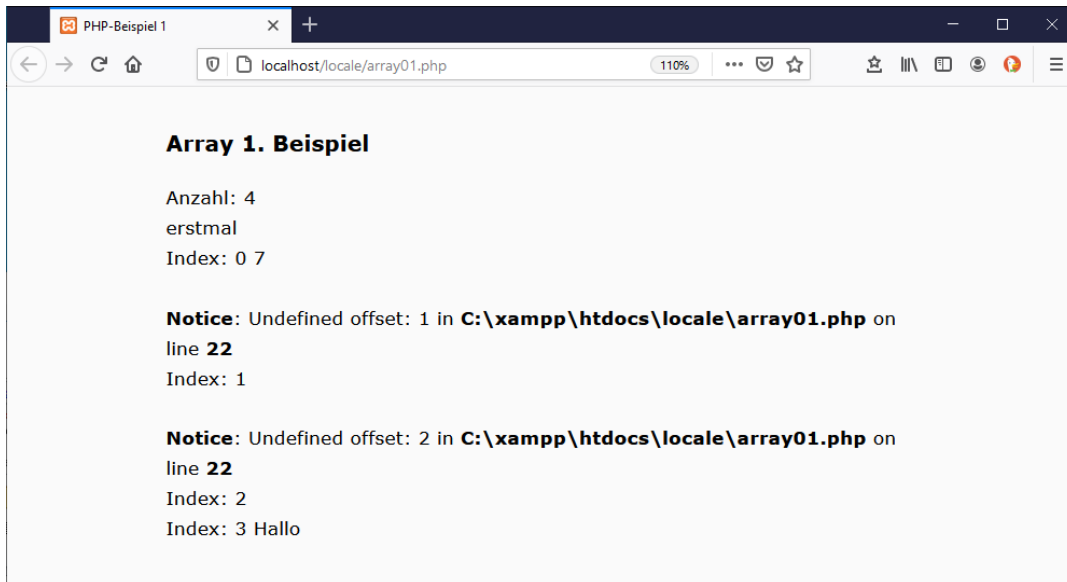


Abbildung 10 Fehlerhafte Indizes

4.8.1.2 Beispiel 2

Abfrage mit einer foreach-Schleife

```
$feld[0] = 7;
$feld[3] = "Hallo";
$feld[5] = true;
$feld[] = "erstmal";
echo 'Anzahl: ' . count($feld) . '<br />';

foreach ($feld as $item) {
    echo 'Items: ' . $item . '<br />';
}
```

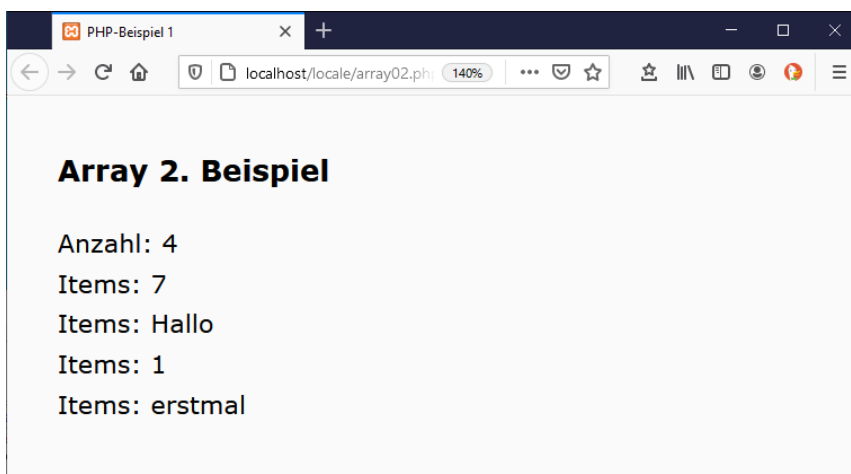


Abbildung 11 Ausgabe mit einer foreach-Schleife

4.8.1.3 Beispiel 3

Das Eintragen in eine Liste sollte nur mit der „Methode“ [] geschehen.

```
$feld = array(); // Anlegen eines neuen Arrays
```

```

$feld[] = 7;
$feld[] = "Hallo";
$feld[] = true;
$feld[] = "erstmal";
echo 'Anzahl: ' . count($feld) . '<br />';

echo $feld[3] . '<br />';
echo 'for-Schleife<br />';
for ($i=0; $i<count($feld); $i++) {
    echo 'Index: ' . $i . ' ' . $feld[$i] . '<br />';
}
echo '<br />';
echo '<br />';
echo 'foreach-Schleife<br />';
foreach ($feld as $item) {
    echo 'Items: ' . $item . '<br />';
}

```

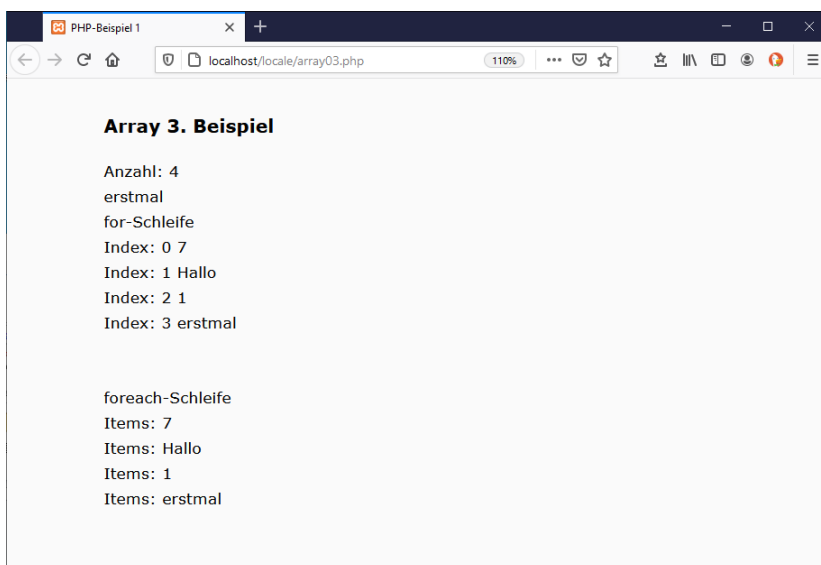


Abbildung 12 Ausgabe mit einer for und foreach-Schleife

4.8.1.4 Vordefinierte Arrays

Ein Array kann auch vordefiniert werden:

```
$feld = array("red", "blue", "green");
```

Danach kann man alle Methoden für Array benutzen.

4.8.1.5 Länge eines Arrays

Methode count

```

$feld = array("red", "blue", "green");
$n = count($feld);
echo $n;

```

4.8.2 Funktionen für Arrays

- [array_change_key_case](#) — Ändert die Groß- oder Kleinschreibung aller Schlüssel in einem Array
- [array_chunk](#) — Splittet ein Array in Teile auf
- [array_column](#) — Return the values from a single column in the input array
- [array_combine](#) — Erzeugt ein Array, indem es ein Array für die Schlüssel und ein anderes für die Werte verwendet
- [array_count_values](#) — Zählt die Werte eines Arrays
- [array_diff_assoc](#) — Berechnet den Unterschied zwischen Arrays mit zusätzlicher Indexprüfung
- [array_diff_key](#) — Berechnet den Unterschied zwischen Arrays, indem es die Schlüssel vergleicht
- [array_diff_uassoc](#) — Berechnet den Unterschied von Arrays mit zusätzlicher Indexprüfung, welche durch eine benutzerdefinierte Callback-Funktion vorgenommen wird
- [array_diff_ukey](#) — Berechnet den Unterschied zwischen Arrays mittels einer Callbackfunktion für den Vergleich der Schlüssel
- [array_diff](#) — Ermittelt die Unterschiede zwischen Arrays
- [array_fill_keys](#) — Befüllt ein Array mit Werten mit den übergebenen Schlüsseln
- [array_fill](#) — Füllt ein Array mit Werten
- [array_filter](#) — Filtert Elemente eines Arrays mittels einer Callback-Funktion
- [array_flip](#) — Vertauscht alle Schlüssel mit ihren zugehörigen Werten in einem Array
- [array_intersect_assoc](#) — Ermittelt die Schnittmenge von Arrays mit Indexprüfung
- [array_intersect_key](#) — Ermittelt die Schnittmenge von Arrays, indem es die Schlüssel vergleicht
- [array_intersect_uassoc](#) — Ermittelt die Schnittmenge von Arrays mit Indexprüfung; vergleicht Indizes mit einer Callbackfunktion
- [array_intersect_ukey](#) — Ermittelt die Schnittmenge zweier Arrays mittels eines durch eine Callbackfunktion durchgeführten Schlüsselvergleiches
- [array_intersect](#) — Ermittelt die Schnittmenge von Arrays
- [array_key_exists](#) — Prüft, ob ein Schlüssel in einem Array existiert
- [array_key_first](#) — Gets the first key of an array
- [array_key_last](#) — Gets the last key of an array
- [array_keys](#) — Liefert alle Schlüssel oder eine Teilmenge aller Schlüssel eines Arrays
- [array_map](#) — Wendet eine Callback-Funktion auf die Elemente von Arrays an
- [array_merge_recursive](#) — Führt ein oder mehrere Arrays rekursiv zusammen
- [array_merge](#) — Führt zwei oder mehr Arrays zusammen
- [array_multisort](#) — Sortiert mehrere oder multidimensionale Arrays
- [array_pad](#) — Füllt ein Array bis auf die angegebene Länge mit einem Wert auf
- [array_pop](#) — **Liefert und entfernt das letzte Element eines Arrays**
- [array_product](#) — Ermittelt das Produkt von Werten in einem Array
- [array_push](#) — **Fügt ein oder mehr Elemente an das Ende eines Arrays an**
- [array_rand](#) — Liefert einen oder mehrere zufällige Schlüssel eines Arrays
- [array_reduce](#) — Iterative Reduktion eines Arrays zu einem Wert mittels einer Callbackfunktion
- [array_replace_recursive](#) — Replaces elements from passed arrays into the first array recursively
- [array_replace](#) — Ersetzt Elemente von übergebenen Arrays im ersten Array
- [array_reverse](#) — Liefert ein Array mit umgekehrter Reihenfolge der Elemente
- [array_search](#) — Durchsucht ein Array nach einem Wert und liefert bei Erfolg den zugehörigen Schlüssel
- [array_shift](#) — **Liefert und entfernt das erste Element eines Arrays**
- [array_slice](#) — Extrahiert einen Ausschnitt eines Arrays
- [array_splice](#) — Entfernt einen Teil eines Arrays und ersetzt ihn durch etwas anderes
- [array_sum](#) — **Liefert die Summe der Werte in einem Array**
- [array_udiff_assoc](#) — Ermittelt den Unterschied zwischen Arrays mit zusätzlicher Indexprüfung, vergleicht mittels einer Callbackfunktion
- [array_udiff_uassoc](#) — Ermittelt den Unterschied zwischen Arrays mit zusätzlicher Indexprüfung, vergleicht Daten und Indizes mittels einer Callbackfunktion

- [array_udiff](#) — Ermittelt den Unterschied zwischen Arrays mittels einer Callbackfunktion für den Datenvergleich
- [array_uintersect_assoc](#) — Ermittelt die Schnittmenge von Arrays mit zusätzlicher Indexprüfung, vergleicht Daten mittels einer Callbackfunktion
- [array_uintersect_uassoc](#) — Ermittelt die Schnittmenge von Arrays mit zusätzlicher Indexprüfung, vergleicht Daten und Schlüssel mittels separaten Callbackfunktionen
- [array_uintersect](#) — Ermittelt die Schnittmenge von Arrays, vergleicht Daten mittels einer Callbackfunktion
- [array_unique](#) — **Entfernt doppelte Werte aus einem Array**
- [array_unshift](#) — Fügt ein oder mehr Elemente am Anfang eines Arrays ein
- [array_values](#) — Liefert alle Werte eines Arrays
- [array_walk_recursive](#) — Wendet eine Benutzerfunktion rekursiv auf jedes Element eines Arrays an
- [array_walk](#) — Wendet eine vom Benutzer gelieferte Funktion auf jedes Element eines Arrays an
- [array](#) — Erstellt ein Array
- [arsort](#) — Sortiert ein Array in umgekehrter Reihenfolge und erhält die Index-Assoziation
- [asort](#) — Sortiert ein Array und erhält die Index-Assoziation
- [compact](#) — Erstellt ein Array mit Variablen und deren Werten
- [count](#) — **Zählt alle Elemente eines Arrays oder etwas in einem Objekt**
- [current](#) — Liefert das aktuelle Element eines Arrays
- [each](#) — Liefert das aktuelle Paar (Schlüssel und Wert) eines Arrays und rückt den Arrayzeiger vor
- [end](#) — Positioniert den internen Zeiger eines Arrays auf dessen letztes Element
- [extract](#) — Importiert Variablen eines Arrays in die aktuelle Symboltabelle
- [in_array](#) — Prüft, ob ein Wert in einem Array existiert
- [key_exists](#) — Alias von `array_key_exists`
- [key](#) — Liefert einen Schlüssel eines Arrays
- [ksort](#) — Sortiert ein Array nach Schlüsseln in umgekehrter Reihenfolge
- [krsort](#) — Sortiert ein Array nach Schlüsseln
- [list](#) — Weist Variablen zu, als wären sie ein Array
- [natcasesort](#) — Sortiert ein Array in "natürlicher Reihenfolge", Groß/Kleinschreibung wird ignoriert
- [natsort](#) — Sortiert ein Array in "natürlicher Reihenfolge"
- [next](#) — Rückt den internen Zeiger eines Arrays vor
- [pos](#) — Alias von `current`
- [prev](#) — Setzt den internen Zeiger eines Arrays um ein Element zurück
- [range](#) — Erstellt ein Array mit einem Bereich von Elementen
- [reset](#) — Setzt den internen Zeiger eines Arrays auf sein erstes Element
- [rsort](#) — Sortiert ein Array in umgekehrter Reihenfolge
- [shuffle](#) — Mischt die Elemente eines Arrays
- [sizeof](#) — Alias von `count`
- [sort](#) — **Sortiert ein Array**
- [uasort](#) — Sortiert ein Array mittels einer benutzerdefinierten Vergleichsfunktion und behält Indexassoziationen bei
- [uksort](#) — Sortiert ein Array nach Schlüsseln mittels einer benutzerdefinierten Vergleichsfunktion
- [usort](#) — Sortiert ein Array nach Werten mittels einer benutzerdefinierten Vergleichsfunktion

Link:

- <https://www.php.net/manual/de/ref.array.php>

4.8.3 Multidimensionale Arrays

4.8.3.1 Beispiel 1

```
$colors = array (
    array('green',0,0,255),
    array('red',255,0,0),
    array('lightgray',200,200,200),
    array('orange',255,128,0)
);
```

Das obige Beispiel definiert ein zwei-dimensionales Arrays. Besser wäre aber ein Array mit einer Klasse „Color“.

4.8.3.2 Beispiel 2

```
$stage = array (
// Index 0
Array (
    'Montag',
    'Dienstag',
    'Mittwoch',
    'Donnerstag',
    'Freitag'
),

// Index 1
array
(
    'morgen',
    'vormittag',
    'mittag',
    'nachmittag',
    'abend',
    'nacht',
)
);

// Ausgabe ergibt Montagabend
echo $stage[0][0] . '<br />';
echo $stage[1][4] . '<br />';
```

4.8.3.3 Beispiel 5

Im Beispiel 5 ist eine Funktion definiert, die eine Matrix à la Java erstellt. Als Parameter werden die Anzahl der Zeilen, n, und die Anzahl der Spalten, m, übergeben. Initialisiert werden die einzelnen „Zellen“ mit 0.

```
function createMatrix($n, $m) {
    $mat = array ();
    for ($r=0; $r<$n; $r++) {
        $row = array();
        for ($c=0; $c<$m; $c++) {
            //$row[] = 0; // Normalfall
            $row[] = ($r+1)*100 + ($c+1); // Als Demo
        }
    }
}
```



```

    }
    $mat[] = $row;
}
return $mat;
}

$n=4;
$m=5;
$matrix = createMatrix($n, $m);
foreach ($matrix as $row) {
    echo 'Count(row)' . count($row) . ' <br />';
    foreach ($row as $item) {
        echo $item. ' &#160;&#160;&#160;&#160;&#160;&#160;';
    }
    echo '<br />';
}
echo '<br />';
echo '<br />';
for ($i=0; $i<$n; $i++) {
    for ($j=0; $j<$m; $j++) {
        echo "m[$i][$j] = $item &#160;&#160;";
    }
    echo '<br />';
}
}

```

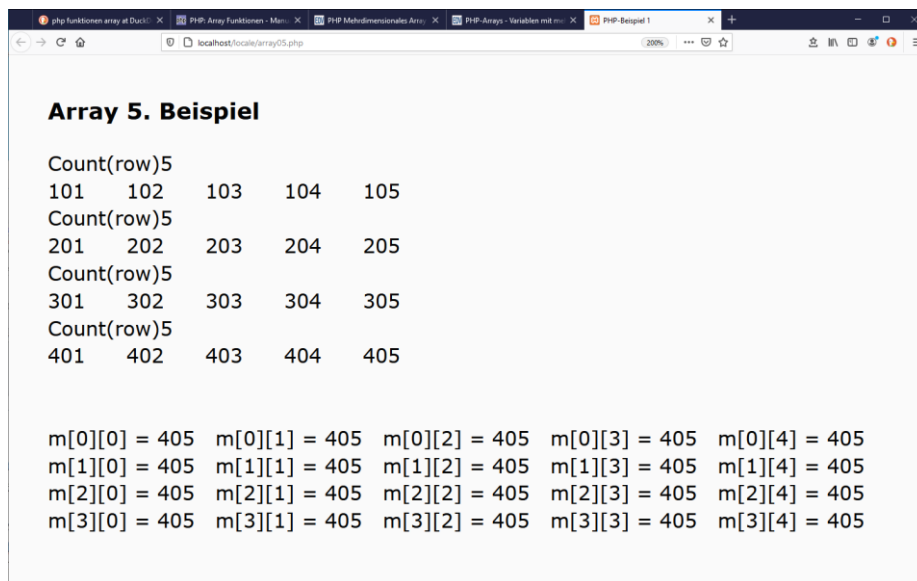


Abbildung 13 Anzeige eines zweidimensionalen Arrays

4.8.4 Assoziierte Arrays (HashTables)

Assoziierte Arrays haben keinen numerischen Index, sondern einen Key, meistens als String, der bestimmt, an welcher Stelle in einem Array ein Wert gespeichert werden soll. Diese Technik ist auch als Hashtable bekannt. Die Parameterübergabe bei Formularen arbeitet nach dem gleichen Prinzip.

Beispiel:

Erstellen:

```
$std1 = array(
```

```

"Math"=>1.3,
"Physik"=>2.7,
"Chemie"=>3.3,
"Englisch"=>2.3,
"Computer"=>5.0
);

$std2 = array();          // nicht unbedingt notwendig
$std2["Math"] = 1.0;
$std2["Physik"] = 2.0;
$std2["Chemie "] = 3.0;
$std2["Englisch"] = 4.0;
$std2["Computer"] = 5.0;

```

4.8.4.1 Abfragen der Elemente:

```

$n1 = $std1['Math'];
$n2 = $std1["Math"];
// echo "Maths: $std1['Maths']<br />";
echo 'Math: ' . $std1['Math'] . '<br />';
echo 'Englisch: ' . $std1['Englisch'] . '<br />';
echo 'Physik: ' . $std1['Physik'] . '<br />';
echo '<br />';

echo 'Math: ' . $std2['Math'] . '<br />';
echo 'Englisch: ' . $std2['Englisch'] . '<br />';
echo 'Physik: ' . $std2['Physik'] . '<br />';

```

Bitte beachten:

- Die untere Anweisung funktioniert leider nicht. Es gibt ein Parserfehler mit den Hochkommas.
 - echo "Maths: \$std1['Maths']
";**

4.8.4.2 Abfragen in einer Schleife:

1. Variante:

Die einfachste Variante ist die foreach-Schleife. Als „Variable“ wird hier aber ein Tupel à la Python angegeben. Das heißt, dass man zwei Variablen pro Durchlauf hat.

```

foreach($std1 as $key=>$note) {
    echo "Key: $key    Value: $note<br>";
}

```

2. Variante:

In dieser Variante werden als erstes alle Schlüssel aus der Hashtable geholt und in der Variablen \$keys gespeichert. Danach wird eine normale for-Schleife benutzt.

```

$length = count($std1);
// holen der Keys
$keys = array_keys($std1);

```

```
// for loop to traverse through the associative array
for($i=0; $i<$length; $i++) {
    $s = $std1[$keys[$i]];
    // echo "    Key: $keys[$i]    Value: $std1[$keys[$i]]<br />";
    echo "    Key: $keys[$i]    Value: " . $std1[$keys[$i]] . '<br />';
}
```

Bitte beachten:

- Die untere Anweisung funktioniert leider nicht. Es gibt ein Parserfehler mit den Klammern.
 - o echo " Key: \$keys[\$i] Value: \$std1[\$keys[\$i]]
";

4.8.4.3 try / catch

Auch in PHP gibt es die moderne try/catch-Fehlerbehandlung:

Syntax:

```
try {
    // Sourcecode
}
catch (Exception e) {
    echo e->getMessage();
}
```

4.9 Funktionen

- Funktionen können rekursiv aufgerufen werden
- Die Argumente werden normalerweise "per value" übergeben
- Sie können jedoch auch als Referenz übergeben werden, wenn dem Argumentnamen ein "&" vorangestellt wurde.
- Das Keyword "static" dient zur Deklaration von lokalen Funktionsvariablen, die zwischen Funktionsaufrufen ihren Wert behalten.

Beispiele:

```
// Rekursion ist erlaubt
function fak($n) {
    if ($n>1)
        return $n*fak($n-1);
    else
        return 1;
}
```

```
$a=2;
$b=3;
$c=addieren($a,$b);
echo ( "Summe von $a und " . $b . " ist " . $c );
function addieren ($summand1, $summand2) {
```

```
    return $summand1 + $summand2;
}
```

Beispiel: call by Reference

```
$a=2;
$b=3;
addieren($a,$b,$c);
echo ( "Summe von $a und " . $b . ": " . $c );

function addieren ($summand1, $summand2, &$serg) {
    $serg = $summand1 + $summand2;
}
```

```
function add ($a, $b, &$c) {
    $c=$a+$b;
}

$x=1;
$y=2;
$z=1;
add($x, $y, $z);
echo $z;
```

```
function add (&$c) {
    $c .= 'a²+b²=c²';
}

$sStr='Pythagoras: ';
add($sStr);
echo $sStr;
```

Im folgenden Beispiel benutzt die Funktion „addieren“ die globale Variable \$a

```
$a=2;
$b=3;
echo "Hier die Summe: " . addieren();
```

// \$a ist NICHT global, also ist \$b null.

```
function addieren() {
    $b = 7;
    return $a + $b;
}
```

Im folgenden Beispiel benutzt die Funktion „addieren“ die echte globale Variable \$a

```
$a=2;
$b=3;
echo "Hier die Summe: " . addieren();

// $a ist nun global, also ist $b null.
function addieren() {
    global $a; // Nachteil der Nichtdeklaration
    $b = 7;
```

```

return $a + $b;
}

```

4.10 String-Methoden

addslashes	Fügt eines Backslash VOR einem bestimmten Zeichens \$str = addslashes("Hello World and Welt!","W"); Ergebnis: Hello \World and \Welt!
addslashes	Fügt vor jedem " ein Backslash. \$str = addslashes("What does "yolo" mean?"); Ergebnis: What does \"yolo\" mean?
bin2hex	Konvertiert eines String in die jeweiligen hexadezimalen ASCII-Zeichen. Verwendet zum Beispiel in MIME-Texten. \$str = bin2hex("Hello World!"); Ergebnis: 48656c6c6f20576f726c6421
chop	Löscht beliebige Texte mittels eines Suchstrings. \$str = "Hello World!" ; echo chop(\$str,"World!"); Ergebnis: Hello
chr	Ausgabe des Zeichens durch einen ASCII-Code chr(65) Ergebnis A chr(97) Ergebnis a chr(080) Ergebnis A chr(0x41) Ergebnis A
chunk_split	Auftrennen eines Strings in Teile nach jedem n-ten Zeichen. Wird häufig benutzt , um einen Text auf eine maximale Breite zu bringen. chunk_split(string,length,end) \$str = "Hello world!"; echo chunk_split(\$str,1,"."); Ergebnis: H.e.l.l.o. .w.o.r.l.d.!.! \$str = "Hello world!"; echo chunk_split(\$str,5,"."); Ergebnis: Hello. world!.!
convert_cyr_string	Konvertiert Kyrillische Zeichen.
convert_uudecode	Dekodiert einen Uni-Code String \$str = ",2&5L;&\@=V]R;&0A `"; echo convert_uudecode(\$str); Ergebnis: Hello world!
convert_uencode	Kodiert einen String in Unicode. \$str = "Hello world!"; echo convert_uencode(\$str); Ergebnis: ,2&5L;&\@=V]R;&0A `
count_chars	Gibt Informationen über den String zurück. count_chars(String, mode) Optional. Specifies the return modes. 0 is default. 0: an array with the ASCII value as key and number of occurrences as value (Häufigkeit der Zeichen) 1: an array with the ASCII value as key and number of occurrences as value, only lists occurrences greater than zero 2: an array with the ASCII value as key and number of occurrences as value,

	<p>only lists occurrences equal to zero are listed 3: a string with all the different characters used 4: a string with all the unused characters</p> <p>\$str = "Hello World!"; echo count_chars(\$str,3); Ergebnis: !HWdelor</p>
crc32	Berechnet einen CRC-Wert eines Strings
crypt	Berechnet einen Hashwert eines Strings.
echo	Ausgabe eines Strings.
explode	Zerlegen eines Strings an Hand eines Trennzeichen, Ergebnis Array
fprintf	<p>Ausgabe eines formatierten Strings in eine Datei. \$number = 9; \$str = "Beijing"; \$file = fopen("test.txt","w"); echo fprintf(\$file,"There are %u million bicycles in %s.", \$number,\$str);</p>
get_html_translation_table	<p>Ausgabe der Konvertierungstabelle für Spezialzeichen von HTML. get_html_translation_table(function,flags,character-set) print_r (get_html_translation_table()); // HTML_SPECIALCHARS . Array ([''] => " [&] => & [<] => < [>] => >) Array ([''] => &quot; [&] => &amp; [<] => &lt; [>] => &gt;) print_r (get_html_translation_table(HTML_ENTITIES)); Array ([''] => &quot; [&] => &amp; [<] => &lt; [>] => &gt; ... [♦] => &diamonds;) </p>
hex2bin	<p>Umwandlung eines hexadezimalen „Strings“ in eine String. Umkehrfunktion der Methode „bin2hex“. echo hex2bin("48656c6c6f20576f726c6421"); Ergebnis: Hello World!</p>
html_entity_decode	<p>Konvertiert in einem String HTML-Zeichen (<>) in echte HTML-Zeichen. html_entity_decode(string,flags,character-set) \$str = '&lt;a href=&quot;https://www.w3schools.com&quot;&gt;w3schools.com&lt;/a&gt;'; echo html_entity_decode(\$str); Ergebnis: w3schools.com</p>
htmlentities	<p>Konvertiert einen HTML-String in einen normalen Text (< in &lt;) \$str = 'Go to w3schools.com'; echo htmlentities(\$str); HTML: &lt;a href=&quot;https://www.w3schools.com&quot;&gt;Go to</p>

	w3schools.com<lt;/a>> Anzeige im Browser: Go to w3schools.com
htmlspecialchars	Konvertiert die spitzen Klammern in <lt; und <gt; Dient zur Ausgabe von Quellcode-Beispielen. \$str = "This is some bold text."; echo htmlspecialchars(\$str); Ergebnis: This is some <lt;b>bold text. Browser: This is some bold text.
implode	Umwandlung eines Array-String in einen Gesamtstring. implode(separator,array) \$arr = array('Hello','World!','Beautiful','Day!'); echo implode(";", \$arr); Ergebnis: Hello;World!;Beautiful;Day!
join	Alias von implode
lcfirst	Konvertiert das erste Zeichen in Kleinbuchstaben (wenn A-Z) echo lcfirst("Hello World!"); Ergebnis: hello World echo lcfirst("ÄHello World!"); Ergebnis: ÄHello World
levenshtein	Rückgabe des levenshtein-Abstandes. Also, wieviele Zeichen man löschen, ersetzen oder einfügen muss, um vom String1 nach String2 zu kommen. levenshtein(string1,string2) levenshtein(string1,string2,insert,replace,delete) Die drei Integer-Parameter sind Gewichtungen. echo levenshtein("Hello World", "ello XWird"); Ergebnis: 3
localeconv	Die Methode „localeconv“ gibt die nationalen Besonderheiten für die Zahlendarstellung als Hashtable zurück (siehe das Kapitel unten).
ltrim	Löscht Leerzeichen und Zeilentrenner am Anfang
md5	Berechnet den MD5-Werte eines Strings \$str = "Hello"; echo md5(\$str); Ergebnis: 8b1a9953c4611296a827abf8c47804d7
metaphone	Berechnet die phonetische Aussprache eines Wortes. echo metaphone("World"); WRLT echo metaphone("Harz"); HRS echo metaphone("Number"); NMR
money_format	Gibt die aktuelle Geldformatierung aus. \$number = 1234.56; setlocale(LC_MONETARY,"en_US"); echo money_format("The price is %i", \$number); Ergebnis: The price is USD 1,234.56
nl_langinfo	Informationen über weitere Länderspezifische Informationen nl_langinfo(element)
nl2br	Ersetzt die Zeilentrennung durch ein echo nl2br("Erste Zeile\nZweite Zeile"); Ergebnis: Erste Zeile Zweite Zeile

number_format	<p>Formatiert eine Zahl in lokales Format (Komma, Punkt).</p> <pre>echo number_format("1000000")."
"; echo number_format("1000000",2)."
"; echo number_format("1000000",2,".",",");</pre> <p>Ergebnis:</p> <p>1,000,000 1,000,000.00 1.000.000,00</p>
ord	<p>Gibt den ASCII-Code eines Zeichens zurück.</p> <pre>echo ord("A");</pre> <p>Ergebnis: 65</p>
parse_str	<p>Ein Parameterstring wird in Variablen geparkt!</p> <pre>parse_str("name=Peter&age=43"); echo \$name echo \$age;</pre>
printf	<p>Ausgabe eines Strings mit Parametern</p> <pre>\$nr1 = 9; \$nr2 = -9; \$nr3 = -9; \$str = "Harz"; printf("There are 1: %u 2: %u 3: %d million students in %s.", \$nr1, \$nr2, \$nr3, \$str);</pre> <p>Ergebnis: Erste Zeile Zweite Zeile There are 1: 9 2: 4294967287 3: -9 million students in Harz. Die zweite Zahl ist fehlerhaft!</p> <p>%u Vorzeichenlose Zahl %d Vorzeichenbehaftete Zahl %s String</p>
quoted_printable_decode	<p>Dekodiert einen „quoted“String in einen 8-Bit-ASCII String</p> <pre>\$str = "Hello=0Aworld."; echo quoted_printable_decode(\$str);</pre> <p>Ergebnis: Hello world.</p>
quoted_printable_encode	<p>Kodiert einen 8-Bit-ASCII String in einen „quoted“String</p> <pre>echo quoted_printable_encode('Hello=0World');</pre> <p>Ergebnis: Hello=3D0World</p>
quotemeta	<p>Fügt ein Backslash vor allen Sonderzeichen. Damit kann man es als Pattern benutzen.</p> <pre>\$str = "Hello world. (can you hear me?); echo quotemeta(\$str);</pre> <p>Ergebnis: Hello world\.\(can you hear me\?)</p>
rtrim	<p>Löscht Leerzeichen und Zeilentrenner am Ende</p>
setlocale	<p>Setzt den länderspezifischen Code.</p> <ul style="list-style-type: none"> • LC_ALL - All of the below • LC_COLLATE - Sort order • LC_CTYPE - Character classification and conversion (e.g. all characters should be lower or upper-case) • LC_MESSAGES - System message formatting • LC_MONETARY - Monetary/currency formatting • LC_NUMERIC - Numeric formatting • LC_TIME - Date and time formatting

sha1	Berechnet den SHA-1 Hashwert eines Strings. \$str = "Hello"; echo sha1(\$str); Ergebnis: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0
similar_text	Rückgabe des String-Abstandes. Also, wieviele Zeichen man löschen, ersetzen oder einfügen muss, um vom String1 nach String2 zu kommen. Diese Methode ist etwas genauer als die Methode levenshtein() . similar_text (<i>string1,string2</i>) echo similar_text ("Hello World","ello XWird"); Ergebnis: 9
soundex	Berechnet die phonetische Aussprache eines Wortes. echo soundex ("World"); W643 echo soundex ("Harz"); H620 echo soundex ("Number"); N516 echo soundex ("Nummer"); N560 echo soundex ("Ball"); B400 echo soundex ("Boll"); N400
sprintf	Wie printf, aber der Text wird in eine Variable gespeichert.
sscanf	Scannt einen Text nach Variablen. \$str = "alter:30 gewicht1:60.2kg gewicht2:60kg"; sscanf(\$str,"alter:%d gewicht1:%fkg gewicht2:%dkg",\$alter,\$gew1, \$gew2); // show types and values var_dump(\$alter,\$gew1, \$gew2); Ergebnis: int(30) float(60.2) int(60) Formate: <ul style="list-style-type: none">• %% - Returns a percent sign• %c - The character according to the ASCII value• %d - Signed decimal number (negative, zero or positive)• %e - Scientific notation using a lowercase (e.g. 1.2e+2)• %u - Unsigned decimal number (equal to or greater than zero)• %f - Floating-point number• %o - Octal number• %s - String• %x - Hexadecimal number (lowercase letters)• %X - Hexadecimal number (uppercase letters) Additional format values. These are placed between the % and the letter (example %.2f): <ul style="list-style-type: none">• + (Forces both + and - in front of numbers. By default, only negative numbers are marked)• ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding)• - (Left-justifies the variable value)• [0-9] (Specifies the minimum width held of to the variable value)• .[0-9] (Specifies the number of decimal digits or maximum string length)
str_getcsv	Analysiert eine CSV-Zeile und fügt die Werte in Variablen (Array). str_getcsv(<i>string,separator,enclosure,escape</i>)

	<p>separator Optional. A character that specifies the field separator. Default is comma (,)</p> <p>enclosure Optional. A character that specifies the field enclosure character. Default is "</p> <p>escape Optional. A character that specifies the escape character. Default is backslash (\)</p> <pre>\$str = 'Max,Mustermann,123456789,Musterhausen'; \$csv_array = str_getcsv (\$str); // Trennung mit , \$str = 'Max;Mustermann;123456789;Musterhausen'; \$csv_array = str_getcsv (\$str, ';');</pre> <pre>\$str = '"Mustermann, Max"',123456789,Musterhausen'; \$csv_array = str_getcsv (\$str, ',', '"'); // Trennung mit , aber mit " Mustermann, Max 123456789 Musterhausen</pre>
str_ireplace	<p>Umbenennen in einem String (Case-Insensitive)</p> <pre>echo str_ireplace("WORLD", "Peter", "Hello world!");</pre>
str_pad	<p>Fügt rechts vom Text zusätzliche Füllzeichen ein. str_pad(string,length,pad_string,pad_type) pad_string Optional. Specifies the string to use for padding. Default is whitespace pad_type Optional. Specifies what side to pad the string.</p> <ul style="list-style-type: none"> • STR_PAD_BOTH - Pad to both sides of the string. If not an even number, the right side gets the extra padding • STR_PAD_LEFT - Pad to the left side of the string • STR_PAD_RIGHT - Pad to the right side of the string. This is default <pre>\$str = "Hello World"; echo str_pad(\$str,20,"."); echo str_pad(\$str,20,".",STR_PAD_LEFT); Ergebnis: Hello World.....Hello World</pre>
str_repeat	<p>Wiederholt eine String n-Mal.</p> <pre>echo str_repeat("Wow",13);</pre>
str_replace	<p>Umbenennen in einem String (Case-Insensitive)</p> <pre>echo str_ireplace("WORLD", "Peter", "Hello world!");</pre> <p>str_replace(find,replace,string,count) Umbenennen in einem String (Case-Sensitive)</p> <pre>echo str_replace("WORLD", "Peter", "Hello world!"); echo str_replace("world", "Peter", "Hello world!"); // okay</pre>
str_rot13	<p>Verschiebt die Buchstaben um 13 Buchstaben</p> <pre>echo str_rot13("Hello World"); Ergebnis: Uryyb Jbeyq echo str_rot13("Uryyb Jbeyq"); Ergebnis: Hello World</pre>
str_shuffle	<p>Mischt die Buchstaben per Zufall.</p> <pre>echo str_shuffle("Hello World"); Ergebnis: WoedlrllHo eordolHWll</pre>

str_split	<p>Splittet einen Text nach den Buchstaben in ein Charakter-Array. <pre>print_r(str_split("Hello")); \$feld = str_split("Hello"); foreach(\$feld as \$ch) { echo \$ch . '
'; }</pre> </p> <p>Ergebnis: Array ([0] => H [1] => e [2] => l [3] => l [4] => o) H e l l o</p>
str_word_count	<p>Zählt die Anzahl der Wörter in einem String. Es trennt die Wörter nach Sonderzeichen und Zahlen. <pre>echo str_word_count("Hello world! abcd1 abc1de");</pre> </p> <p>Ergebnis: 5</p>
strcasecmp	<p>Vergleicht zwei String (case insensitive) Returnwert: 0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</p> <p><code>echo</code> <code>strcasecmp("Hello world!","HELLO WORLD!");</code></p>
strchr	<p>Sucht einen Text in einem String und gibt den Rest aus. <pre>strchr(string,search,before_search);</pre> <i>before_search false: Ausgabe nach dem Text</i> <i>before_search true: Ausgabe vor dem Text</i></p>
strcmp	<p>Vergleicht zwei String (case sensitive) Returnwert: 0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</p> <p><code>echo</code> <code>strcmp("Hello world!","HELLO WORLD!");</code></p>
strcoll	<p>Vergleicht zwei String (case sensitive) Der Vergleich kann hier geändert werden (A<a or A>a) Returnwert: 0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</p> <p><code>echo</code> <code>strcoll ("Hello world!","HELLO WORLD!");</code></p>
strcspn	<p>Anzeige der Zeichen, bis man das Suchenzeichen zum erstenmal gefunden hat. <pre>strcspn(string,char,start,length)</pre></p> <p><code>echo</code> <code>strcspn("Hello with world!","w");</code> Ergebnis: 6</p> <p><code>echo</code> <code>strcspn("Hello with world!","wo");</code> Ergebnis: 4 // Ergebnis unklar</p>

	<pre>echo strspn("Hello wither world!", "w", 7);</pre> <p>Ergebnis: 6</p>
strip_tags	<p>Trennt aus einem String HTML oder XML-Tags.</p> <pre>echo strip_tags("Hello world!");</pre> <p>Hello world!</p>
stripslashes	<p>Entfernt die Escape-Sequenzen (Backslash) eines Strings.</p> <pre>echo stripslashes("Hello \World! \.Welt\\Welt");</pre> <p>Ergebnis: Hello World! .WeltWelt</p> <p>Die beiden Backslash am Ende werden komplett entfernt.</p>
stripslashes	<p>Entfernt die Escape-Sequenzen (Backslash) eines Strings.</p> <pre>echo stripslashes("Hello \World! \.Welt\\Welt");</pre> <p>Ergebnis: Hello World! .WeltWelt</p> <p>Die beiden Backslash am Ende werden komplett entfernt.</p>
stripos	<p>Sucht das erste Vorkommen des Suchstrings in einem Text. (case insensitive)</p> <pre>echo stripos("I love php, I love php too!", "PHP");</pre> <p>Ergebnis: 7</p>
strstr	<p>Sucht einen Text in einem String und gibt den Rest aus.</p> <p>case-insensitive</p> <pre>strstr(string, search, before_search);</pre> <p><i>before_search false: Ausgabe nach dem Text</i> <i>before_search true: Ausgabe vor dem Text</i></p>
strlen	<p>Bestimmt die Länge eines Strings</p>
strnatcasecmp	<p>Vergleicht zwei String (case insensitive).</p> <p>Hier werden aber die Nummer korrekt verglichen:</p> <pre>1<2 -1 2<10 -1</pre> <p>Returnwert:</p> <pre>0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</pre> <pre>echo strnatcasecmp("2Hello world!", "10Hello WORLD!"); echo strnatcasecmp("10Hello world!", "2Hello WORLD!");</pre> <p>Ergebnis:</p> <pre>-1 +1</pre>
strnatcmp	<p>Vergleicht zwei String (case sensitive).</p> <p>Hier werden aber die Nummer korrekt verglichen:</p> <pre>1<2 -1 2<10 -1</pre> <p>Returnwert:</p> <pre>0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</pre> <pre>echo strnatcasecmp("2Hello world!", "10Hello WORLD!");</pre>

	<pre>echo strnatcasecmp("10Hello world!","2Hello WORLD!");</pre> <p>Ergebnis:</p> <pre>-1 +1</pre>
strncmp	<p>Vergleicht zwei String, aber nur bis zu einer Länge. <code>strncmp(string1,string2,length)</code></p> <p>Returnwert:</p> <pre>0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</pre> <pre>echo strncmp("Hello world!","Hello earth!",6);</pre> <p>Ergebnis: 0</p> <pre>echo strncmp("Hello world!","Hello earth!",7);</pre> <p>Ergebnis: 4608 1. String ist größer !</p>
strpbrk	<p>Sucht in einem String nach einem Zeichen aus der Liste. Gibt dann die restlichen Zeichen zurück. <code>strpbrk(string,charlist)</code> <pre>echo strpbrk("Hello world!","oe");</pre> <p>Ergebnis: ello world!</p> </p>
strpos	<p>Sucht eine Text in einem String und gibt die Position zurück oder true <code>strpos(string,find,start)</code> start Optional. Specifies where to begin the search. If start is a negative number, it counts from the end of the string.</p> <pre>echo strpos("I love php, I love php too!","php");</pre> <pre>echo '
';</pre> <pre>\$b = strpos("I love php, I love php too!","ppp");</pre> <pre>if (\$b) {</pre> <pre> echo 'Nicht gefunden
';</pre> <pre>}</pre> <pre>echo gettype(strpos("I love php, I love php too!","ppp"))</pre> <p>Ergebnis:</p> <pre>7</pre> <p>Nicht gefunden</p> <p>boolean</p> <p>Related functions:</p> <ul style="list-style-type: none"> • <code>strrpos()</code> - Finds the position of the last occurrence of a string inside another string (case-sensitive) • <code>stripos()</code> - Finds the position of the first occurrence of a string inside another string (case-insensitive) • <code>strripos()</code> - Finds the position of the last occurrence of a string inside another string (case-insensitive)
strrchr	<p>Sucht eine Text in einem String und gibt alle Zeichen danach aus. <pre>echo strrchr("Hello world!","world");</pre> <p>Ergebnis: world!</p> </p>
strstr	<p>Sucht einen Text in einem String und gibt den Rest aus. <code>strstr(string,search,before_search);</code> before_search false: Ausgabe nach dem Text</p>

	<i>before_search true: Ausgabe vor dem Text</i>
strrev	Kehrt den String um.
stripos	Sucht den letzten Text in einem String (case insensitive). echo stripos("I love php, I love php too!", "PHP"); Ergebnis: 19
strrpos	Sucht den letzten Text in einem String (case sensitive). echo strrpos("I love php, I love php too!", "php"); echo ' '; echo strrpos("I love php, I love php too!", "PHP"); echo ' '; \$b = strrpos("I love php, I love php too!", "PHP"); echo gettype(\$b) . ' '; if (!\$b) { echo 'Nicht gefunden '; }
strspn	Sucht alle Zeichen aus der Suchliste in dem String. strspn(string, charlist, start, length) start Optional. Specifies where in the string to start length Optional. Defines the length of the string echo strspn("Hello world!", "kHlleo"); Ergebnis: 5
strstr	Sucht einen Text in einem String und gibt den Rest zurück. echo strstr("Hello world!", "world"); Ergebnis: world!
strtok	Die Methode spaltet den String nach einem Zeichen. Ähnlich „explode“, aber die Rückgabe ist immer nur ein Teilstring. \$string = "Hello world. Beautiful day today."; \$token = strtok(\$string, " "); while (\$token !== false) { echo "\$token "; \$token = strtok(" "); } Ergebnis: Hello world. Beautiful day today.
strtolower	Umwandeln eines String in Kleinbuchstaben
strtoupper	Umwandeln eines String in Großbuchstaben
strtr	Ersetzt in einem Text ein Zeichen durch ein anderes. strtr(string, fromListe, toListe) echo strtr("Hilla Winter Banane Warld", "ia", "eo");

	<p>Ergebnis: Hello Wenter Bonone World</p> <p>i ersetzt durch e a ersetzt durch o</p>
substr	<p>Gib einen Teilstring eines Strings zurück. substr(<i>string</i>,<i>start</i>,<i>length</i>) echo substr("Hello world",6); Ergebnis: world</p>
substr_compare	<p>Vergleicht zwei String, aber nur bis zu einer Länge. strncmp(<i>string1</i>,<i>string2</i>,<i>start</i>, <i>length</i>, <i>case</i>) length und case sind optional. case: false case-sensitive case: true case-insensitive Returnwert: 0 Beide String sind gleich <0: String1 ist kleiner als String2 >0: String1 ist größer als String2</p> <p>echo substr_compare("Hello world","Hello world",0); 0 echo substr_compare("HEllo world","Hello world",0); -8192 echo substr_compare ("HelLo world!","Hello earth!",2,3, false); 36 unklar echo substr_compare ("HelLo world!","Hello earth!",2,3, true); 4 unklar echo substr_compare ("Hello world!","Hello earth!",6); 47 echo substr_compare ("Hello world!","Hello earth!",7); 39</p>
substr_count	<p>Zählt die Anzahl der Treffer eines Suchstring in einem Text. substr_count(<i>string</i>,<i>substring</i>,<i>start</i>,<i>length</i>) start Optional. Specifies where in the string to start length Optional. Defines the length of the string</p> <p>echo substr_count("Hello world. The world is nice","world"); Ergebnis: 2</p>
substr_replace	<p>Ersetzt den ersten String durch den zweiten String. substr_replace (<i>string</i>,<i>substring</i>,<i>start</i>,<i>length</i>) start Optional. Specifies where in the string to start length Optional. Defines the length of the string</p>
trim	<p>Löscht Leerzeichen und Zeilentrenner am Anfang und am Ende</p>
ucfirst	<p>Konvertiert das erste Zeichen in einen Großbuchstaben, wenn es ein Buchstabe ist. echo ucfirst("hello world!"); Hello world! echo ucfirst("123hello world!"); 123hello world!</p>

	<pre>echo ucfirst("123 hello world!"); 123 hello world!</pre>
ucwords	<p>Konvertiert das erste Zeichen eines Wortes in einen Großbuchstaben, wenn es ein Buchstabe ist.</p> <pre>echo ucwords ("hello world!"); Hello World! echo ucwords ("123hello world!"); 123hello World! echo ucwords ("123 hello world!"); 123 Hello World!</pre>
vprintf	<p>Ausgabe eines Strings mit Parametern. Die Werte werden hier aber als Array übergeben.</p> <pre>\$nr1 = 9; \$nr2 = -9; \$nr3 = -12.59; \$str = "Harz"; vprintf ("There are 1: %u 2: %d 3: %f million students in %s.", array(\$nr1, \$nr2, \$nr3, \$str));</pre> <p>Ergebnis: There are 1: 9 2: -9 3: -12.590000 million students in Harz.</p> <p>%u Vorzeichenlose Zahl %d Vorzeichenbehaftete Zahl %f Fließkommazahl %s String</p>
vsprintf	<p>Ausgabe eines Strings mit Parametern in eine Variable. Die Werte werden hier aber als Array übergeben.</p> <pre>\$nr1 = 9; \$nr2 = -9; \$nr3 = -12.59; \$str = "Harz"; \$text = vsprintf ("There are 1: %u 2: %d 3: %f million students in %s.", array(\$nr1, \$nr2, \$nr3, \$str)); echo \$text;</pre> <p>Ergebnis: There are 1: 9 2: -9 3: -12.590000 million students in Harz.</p> <p>%u Vorzeichenlose Zahl %d Vorzeichenbehaftete Zahl %f Fließkommazahl %s String</p>
wordwrap	<p>Bricht einen Text mit einer angegebenen Breite um.</p> <pre>\$str = "An example of a long word is: Supercalifragulistic"; echo wordwrap(\$str,15,"
\n");</pre> <p>Ergebnis: An example of a long word is: Supercalifragulistic</p>

4.10.1 localeconv

Die Methode „localeconv“ gibt die nationalen Besonderheiten für die Zahlendarstellung als Hashtable zurück.

The localeconv() function will return the following array elements:

- [decimal_point] - Decimal point character
- [thousands_sep] - Thousands separator
- [int_curr_symbol] - Currency symbol (example: USD)
- [currency_symbol] - Currency symbol (example: \$)
- [mon_decimal_point] - Monetary decimal point character
- [mon_thousands_sep] - Monetary thousands separator
- [positive_sign] - Positive value character
- [negative_sign] - Negative value character
- [int_frac_digits] - International fractional digits
- [frac_digits] - Local fractional digits
- [p_cs_precedes] - True (1) if currency symbol is placed in front of a positive value, False (0) if it is placed behind
- [p_sep_by_space] - True (1) if there is a spaces between the currency symbol and a positive value, False (0) otherwise
- [n_cs_precedes] - True (1) if currency symbol is placed in front of a negative value, False (0) if it is placed behind
- [n_sep_by_space] - True (1) if there is a spaces between the currency symbol and a negative value, False (0) otherwise
- [p_sign_posn] - Formatting options:
 - 0 - Parentheses surround the quantity and currency symbol
 - 1 - The + sign is placed in front of the quantity and currency symbol
 - 2 - The + sign is placed after the quantity and currency symbol
 - 3 - The + sign is placed immediately in front of the currency symbol
 - 4- The + sign is placed immediately after the currency symbol
- [n_sign_posn] - Formatting options:
 - 0 - Parentheses surround the quantity and currency symbol
 - 1 - The - sign is placed in front of the quantity and currency symbol
 - 2 - The - sign is placed after the quantity and currency symbol
 - 3 - The - sign is placed immediately in front of the currency symbol
 - 4 - The - sign is placed immediately after the currency symbol
- [grouping] - Array displaying how numbers are grouped (example: 3 indicates 1 000 000)
- [mon_grouping] - Array displaying how monetary numbers are grouped (example: 2 indicates 1 00 00 00)

Beispiel:

```
setlocale(LC_ALL, "de");
$loc = localeconv();
print_r($loc) . '<br /><br /><br />';
echo $loc['decimal_point'] . '<br />';
echo $loc['thousands_sep'] . '<br />';
echo $loc['mon_decimal_point'] . '<br />';
echo $loc['frac_digits'] . '<br />';
$grp = $loc['grouping'];
echo 'grp[0]: ' . $grp[0] . '<br />';
echo '<br />';
```

```
echo $loc['currency_symbol'] . '<br />';
```

Formatierte Ausgabe:

```
Array (
  [decimal_point] => ,
  [thousands_sep] => .
  [int_curr_symbol] => EUR
  [currency_symbol] => €
  [mon_decimal_point] => ,
  [mon_thousands_sep] => .
  [positive_sign] =>
  [negative_sign] => -
  [int_frac_digits] => 2
  [frac_digits] => 2
  [p_cs_precedes] => 0
  [p_sep_by_space] => 1
  [n_cs_precedes] => 0
  [n_sep_by_space] => 1
  [p_sign_posn] => 1
  [n_sign_posn] => 1
  [grouping] => Array ([0] => 3 )
  [mon_grouping] => Array ( [0] => 3 ) ) ,

.
,
2
grp[0]: 3
€
```

Tip: To define locale settings, see the `setlocale()` function.

4.10.2 nl_langinfo

`nl_langinfo(element)`

Parameter	Description
<i>element</i>	<p>Required. Specifies which element to return. Must be one of the following elements:</p> <p>Time and Calendar:</p> <ul style="list-style-type: none">ABDAY_(1-7) - Abbreviated name of the numbered day of the weekDAY_(1-7) - Name of the numbered day of the week (DAY_1 = Sunday)ABMON_(1-12) - Abbreviated name of the numbered month of the yearMON_(1-12) - Name of the numbered month of the yearAM_STR - String for Ante meridianPM_STR - String for Post meridianD_T_FMT - String that can be used as the format string for <code>strftime()</code> to represent time and dateD_FMT - String that can be used as the format string for <code>strftime()</code> to represent dateT_FMT - String that can be used as the format string for <code>strftime()</code> to represent timeT_FMT_AMPM - String that can be used as the format string for <code>strftime()</code> to represent time in 12-hour format with ante/post meridian

- ERA - Alternate era
- ERA_YEAR - Year in alternate era format
- ERA_D_T_FMT - Date and time in alternate era format (string can be used in strftime())
- ERA_D_FMT - Date in alternate era format (string can be used in strftime())
- ERA_T_FMT - Time in alternate era format (string can be used in strftime())

Monetary Category:

- INT_CURR_SYMBOL - Currency symbol (example: USD)
- CURRENCY_SYMBOL - Currency symbol (example: \$)
- CRNCYSTR - Same as CURRENCY_SYMBOL
- MON_DECIMAL_POINT - Monetary decimal point character
- MON_THOUSANDS_SEP - Monetary thousands separator
- POSITIVE_SIGN - Positive value character
- NEGATIVE_SIGN - Negative value character
- MON_GROUPING - Array displaying how monetary numbers are grouped (example: 1 000 000)
- INT_FRAC_DIGITS - International fractional digits
- FRAC_DIGITS - Local fractional digits
- P_CS_PRECEDES - True (1) if currency symbol is placed in front of a positive value, False (0) if it is placed behind
- P_SEP_BY_SPACE - True (1) if there is a spaces between the currency symbol and a positive value, False (0) otherwise
- N_CS_PRECEDES - True (1) if currency symbol is placed in front of a negative value, False (0) if it is placed behind
- N_SEP_BY_SPACE - True (1) if there is a spaces between the currency symbol and a negative value, False (0) otherwise
- P_SIGN_POSN - Formatting setting. Possible return values:
 - 0 - Parentheses surround the quantity and currency symbol
 - 1 - The sign string is placed in front of the quantity and currency symbol
 - 2 - The sign string is placed after the quantity and currency symbol
 - 3 - The sign string is placed immediately in front of the currency symbol
 - 4 - The sign string is placed immediately after the currency symbol
- N_SIGN_POSN - Formatting setting. Possible return values:
 - 0 - Parentheses surround the quantity and currency symbol
 - 1 - The sign string is placed in front of the quantity and currency symbol
 - 2 - The sign string is placed after the quantity and currency symbol
 - 3 - The sign string is placed immediately in front of the currency symbol
 - 4 - The sign string is placed immediately after the currency symbol

Numeric Category:

- DECIMAL_POINT - Decimal point character
- RADIXCHAR - Same as DECIMAL_POINT
- THOUSANDS_SEP - Separator character for thousands
- THOUSEP - Same as THOUSANDS_SEP
- GROUPING - Array displaying how numbers are grouped (example: 1 000

	000) Messaging Category: <ul style="list-style-type: none"> • YESEXPR - Regex string for matching 'yes' input • NOEXPR - Regex string for matching 'no' input • YESSTR - Output string for 'yes' • NOSTR - Output string for 'no' Code set Category: <ul style="list-style-type: none"> • CODESET Return a string with the name of the character encoding.
--	--

4.10.3 Ländercode

Land	ISO Code
Abkhazian	ab
Afar	aa
Afrikaans	af
Akan	ak
Albanian	sq
Amharic	am
Arabic	ar
Aragonese	an
Armenian	hy
Assamese	as
Avaric	av
Avestan	ae
Aymara	ay
Azerbaijani	az
Bambara	bm
Bashkir	ba
Basque	eu
Belarusian	be
Bengali (Bangla)	bn
Bihari	bh
Bislama	bi
Bosnian	bs
Breton	br
Bulgarian	bg
Burmese	my
Catalan	ca
Chamorro	ch
Chechen	ce
Chichewa, Chewa, Nyanja	ny
Chinese	zh
Chinese (Simplified)	zh-Hans
Chinese (Traditional)	zh-Hant
Chuvash	cv

Cornish	kw
Corsican	co
Cree	cr
Croatian	hr
Czech	cs
Danish	da
Divehi, Dhivehi, Maldivian	dv
Dutch	nl
Dzongkha	dz
English	en
Esperanto	eo
Estonian	et
Ewe	ee
Faroese	fo
Fijian	fj
Finnish	fi
French	fr
Fula, Fulah, Pulaar, Pular	ff
Galician	gl
Gaelic (Scottish)	gd
Gaelic (Manx)	gv
Georgian	ka
German	de
Greek	el
Greenlandic	kl
Guarani	gn
Gujarati	gu
Haitian Creole	ht
Hausa	ha
Hebrew	he
Herero	hz
Hindi	hi
Hiri Motu	ho
Hungarian	hu
Icelandic	is
Ido	io
Igbo	ig
Indonesian	id, in
Interlingua	ia
Interlingue	ie
Inuktitut	iu
Inupiak	ik
Irish	ga
Italian	it
Japanese	ja
Javanese	jv
Kalaallisut, Greenlandic	kl
Kannada	kn
Kanuri	kr
Kashmiri	ks
Kazakh	kk
Khmer	km

Kikuyu	ki
Kinyarwanda (Rwanda)	rw
Kirundi	rn
Kyrgyz	ky
Komi	kv
Kongo	kg
Korean	ko
Kurdish	ku
Kwanyama	kj
Lao	lo
Latin	la
Latvian (Lettish)	lv
Limburgish (Limburger)	li
Lingala	ln
Lithuanian	lt
Luga-Katanga	lu
Luganda, Ganda	lg
Luxembourgish	lb
Manx	gv
Macedonian	mk
Malagasy	mg
Malay	ms
Malayalam	ml
Maltese	mt
Maori	mi
Marathi	mr
Marshallese	mh
Moldavian	mo
Mongolian	mn
Nauru	na
Navajo	nv
Ndonga	ng
Northern Ndebele	nd
Nepali	ne
Norwegian	no
Norwegian bokmål	nb
Norwegian nynorsk	nn
Nuosu	ii
Occitan	oc
Ojibwe	oj
Old Church Slavonic, Old Bulgarian	cu
Oriya	or
Oromo (Afaan Oromo)	om
Ossetian	os
Pāli	pi
Pashto, Pushto	ps
Persian (Farsi)	fa
Polish	pl
Portuguese	pt
Punjabi (Eastern)	pa
Quechua	qu

Romansh	rm
Romanian	ro
Russian	ru
Sami	se
Samoan	sm
Sango	sg
Sanskrit	sa
Serbian	sr
Serbo-Croatian	sh
Sesotho	st
Setswana	tn
Shona	sn
Sichuan Yi	ii
Sindhi	sd
Sinhalese	si
Siswati	ss
Slovak	sk
Slovenian	sl
Somali	so
Southern Ndebele	nr
Spanish	es
Sundanese	su
Swahili (Kiswahili)	sw
Swati	ss
Swedish	sv
Tagalog	tl
Tahitian	ty
Tajik	tg
Tamil	ta
Tatar	tt
Telugu	te
Thai	th
Tibetan	bo
Tigrinya	ti
Tonga	to
Tsonga	ts
Turkish	tr
Turkmen	tk
Twi	tw
Uyghur	ug
Ukrainian	uk
Urdu	ur
Uzbek	uz
Venda	ve
Vietnamese	vi
Volapük	vo
Wallon	wa
Welsh	cy
Wolof	wo
Western Frisian	fy
Xhosa	xh
Yiddish	yi, ji

Yoruba	yo
Zhuang, Chuang	za
Zulu	zu

4.11 Klassen

Ab Version 5.3 ist in PHP auch OOP möglich. Die Syntax ist sehr ähnlich der von Java. Der Konstruktor, der Dekonstruktor und der Aufruf des „super“- Konstruktor sind anders.

4.11.1 Defintion

- Attribute (private,protected,public)
- Setter (Memberfunktionen)
- Getter (Memberfunktionen)
- Allgemeine Methoden (private,protected,public)
- Ableitung mit „extends“
- Konstruktor
- Aufruf des super-Konstruktor
- Dekonstruktor
- Überladen innerhalb einer Klasse
- Polymorphismus (Überschreiben mit mehreren Klassen)


```

class myPoint extends BaseClass{
    const requiredMargin = 1.7;

    private/protected/public $attrib2=1;

    public function setter/getter {
    }

    function __construct( $attrib1, $attrib2) {
        $this-> $$attrib2 = $attrib2;
        BaseClass::BaseClass($attrib1);    // super($attrib1);
        parent::__construct($attrib1);      // Alternative
    }

    function __destruct( ) {
        // aufräumen, Datei schließen
        parent::__destruct();
    }

    public function print() {
    }

}

```

4.11.2 Abstrakte Klassen

```

abstract class MyAbstractClass {

    abstract function myAbstractFunction() {
    }

}

```

4.11.3 Interface

```

interface ICSV {
    public function export2CSV(); // keine Rückgabe definiert!
}

// Implementiere das Interface
Class MyClass implements ICSV {
    private $attrib1 = 1;

    public function export2CSV() {
        return $attrib1 . ';';
    }
}

```

4.11.4 Einfaches Beispiel

```

class myPoint {

```

```

private $x=1;
private $y=1;

public function getX() {
    return $x;
}
public function getY() {
    return $y;
}

public function setX($value) {
    $x=$value;
}
public function setY($value) {
    $y=$value;
}
} // myPoint

```

Aufruf:

```

define( "lf", "<br />" );
$p1 = new myPoint;
$p1->setX(100);
echo 'p1(x): ' . $p1->getX() .lf ;

```

4.11.5 Beispiel mit Konstruktor

```

class myPoint {
    private $x=1;
    private $y=1;

    public function __construct($px, $py) {
        $x=$px;
        this->y=$py;
    }

    public function __deconstruct() {
        $x=0;
        $y=0;
    }

    // Aufruf
    define( "lf", "<br />" );
    $p1 = new myPoint(12,44);
    $p1->setX(100);
    echo "p1(x): " . $p1->getX() .lf ;

```

4.11.6 Beispiel mit abgeleiteter Klasse

```

class myPoint2 {
    private $x=1;
    private $y=1;

```

```

    public function setX($value) {
        $this->x=$value;
    }
    public function getX() {
        return $this->x;
    }
    public function setY($value) {
        $this->y=$value;
    }
    public function getY() {
        return $this->y;
    }

    public function __construct($px, $py) {
        $this->x=$px;
        $this->y=$py;
    }
    public function __destruct() {
        $this->x=0;
        $this->y=0;
    }
} // myPoint2
class myPoint3 extends myPoint2 {
    private $z=3;

    public function setZ($value) {
        $this->z=$value;
    }
    public function getZ() {
        return $this->z;
    }

    public function __construct($px, $py, $pz) {
        parent::__construct($px, $py);
        $this->z=$pz;
    }
    public function __destruct() {
        parent::__destruct();
        $this->z=0;
    }
} // myPoint3

```

Aufruf:

```

define( "lf", "<br />" );
$p1 = new myPoint2(100,200);
echo ("p1(x): " . $p1->getX() .lf );
echo ("p1(y): " . $p1->getY() .lf );

$p2 = new myPoint3(10,20,30);
echo "p2(x): " . $p2->getX() .lf;
echo "p2(y): " . $p2->getY() .lf;
echo "p2(z): " . $p2->getZ() .lf;

```

4.11.7 Eintragen in eine Liste (Beispiel Wertetabelle)

Dieses Kapitel zeigt die Verwendung einer Klasse, deren Instanzen in eine Liste eingetragen werden. Am Schluss wird aus der Liste dann ein JSON-String erzeugt.

4.11.7.1 Klasse Point

Die Klasse dient zur Speicherung der X/Y-Koordinaten. Sie hat öffentliche Attribute, also braucht sie keine setter/getter-Methoden. Damit ist auch der Export nach JSON einfacher.

```
class myPoint {
    public $x=1;
    public $y=1;

    public function __construct($px, $py) {
        $this->x=$px;
        $this->y=$py;
    }
} // myPoint
```

Gegeben sind:

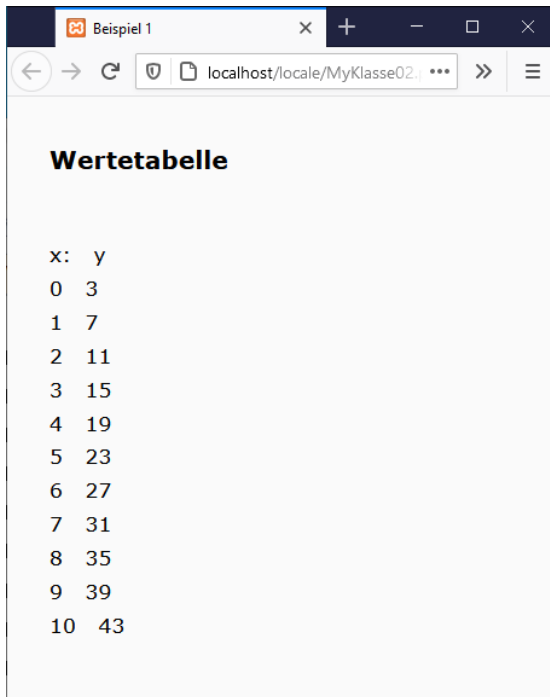
- ist eine feste Funktion: $y=2 \cdot x^2 + 3$
- xa (Anfangswert)
- xsw (Schrittweite)
- xe (Endwert)

1. Variante:

```
$xa=0;
$xsw=1.0;
$xe=10;
$x=$xa;

$feld = array();
while ($x<=$xe+$xsw*0.1) {
    $y = fx($x);
    $feld[] = new myPoint($x,$y);
    $x+=$xsw;
}

define( "lf", "<br />" );
echo '<br />';
echo 'x:&#160;&#160;&#160;y' . lf;
foreach ($feld as $p) {
    echo "$p->x&#160;&#160;&#160;$p->y" . lf;
}
```



Beispiel 1

localhost/locale/MyKlasse02...

Wertetabelle

x:	y
0	3
1	7
2	11
3	15
4	19
5	23
6	27
7	31
8	35
9	39
10	43

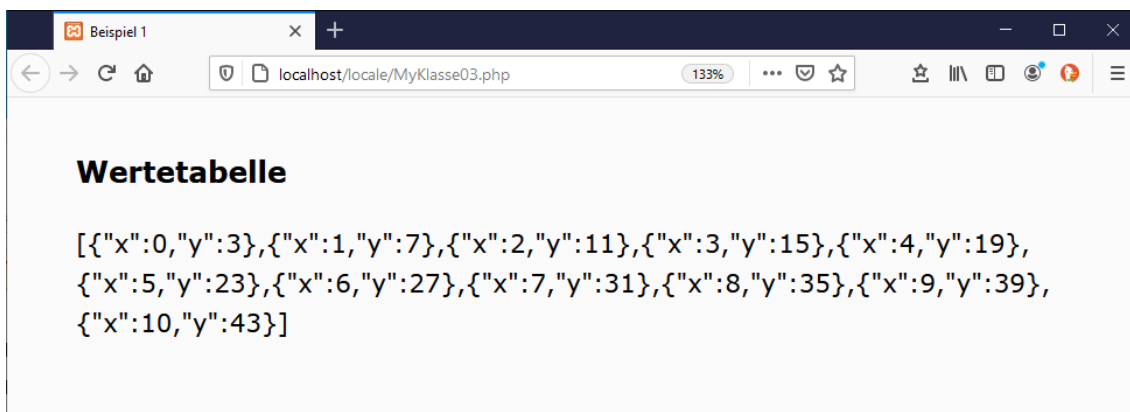
Abbildung 14 Wertetabelle mit Klassen

2. Variante (JSON):

```
$xa=0;
$sw=1.0;
$xe=10;
$x=$xa;

$feld = array();
while ($x<=$xe+$sw*0.1) {
    $y = fx($x);
    $feld[] = new myPoint($x,$y);
    $x+=$sw;
}

$s = json_encode($feld);
echo $s;
```



Beispiel 1

localhost/locale/MyKlasse03.php

Wertetabelle

[{"x":0,"y":3}, {"x":1,"y":7}, {"x":2,"y":11}, {"x":3,"y":15}, {"x":4,"y":19}, {"x":5,"y":23}, {"x":6,"y":27}, {"x":7,"y":31}, {"x":8,"y":35}, {"x":9,"y":39}, {"x":10,"y":43}]

Abbildung 15 Wertetabelle mit Klassen und JSON

5 Formulare

Dynamische Webseiten müssen auf Eingaben des Nutzers reagieren können. Dazu benötigt man aktive HTML-Elemente. Die Daten in den Eingabeelementen werden dann per GET oder POST an den Server geschickt. Dieser wertet die Daten aus und bestimmt die Ausgabe, die an den Nutzer resp. Browser geschickt werden.

5.1 Überblick

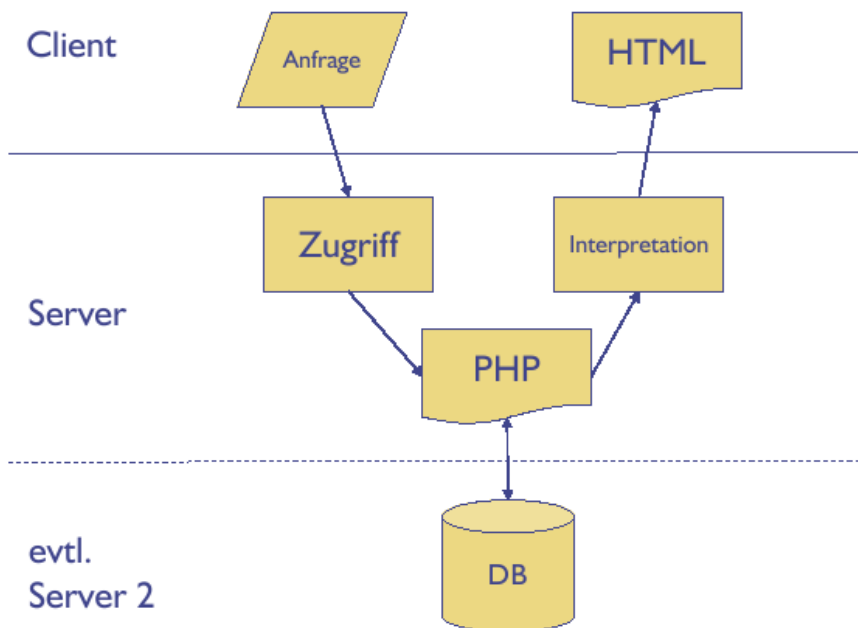


Abbildung 16 Überblick über den Verlauf einer Server-Anfrage

Der Nutzer gibt im Browser Auswahldaten ein, zum Beispiel den gewünschten Fachbereich. Diese Eingabe wird dann vom Server in einem PHP-Script entgegengenommen, geprüft und ausgewertet. Es wird beispielsweise eine Datenbankabfrage gestartet und die Ergebnisse der Abfrage mittels JSON an den Browser zurückgeschickt. Dieser stellt dann, beispielsweise die Studiengänge des gewählten Fachbereich als Liste dar.

5.2 Formular-Struktur

```
<form method="post/get" action="/scripts/process.php">
  <input type="text" name="fb" size="8" value="AI"/>
  <input type="submit" value="Send" />
</form>
```

Jedes Formular hat mindestens einen Form-Abschnitt. Das Attribut „method“ muss nicht angegeben werden, da die get-Methode als Standard gesetzt ist. Das „action“-Attribut aber muss gesetzt werden. Wenn es nicht gesetzt ist, wird die aktuelle Seite / Datei, mit den Eingabewerten aufgerufen!

5.2.1 Beispiel 1

Quellcode der ersten Seite: form1a.html

```
<body>
  <h3>1. Beispiel</h3>
  <form action="form1b.php" >
    <input type="text" name="fb" size="8" value="AI"/>
    <input type="submit" value="Send" />
  </form>

</body>
```

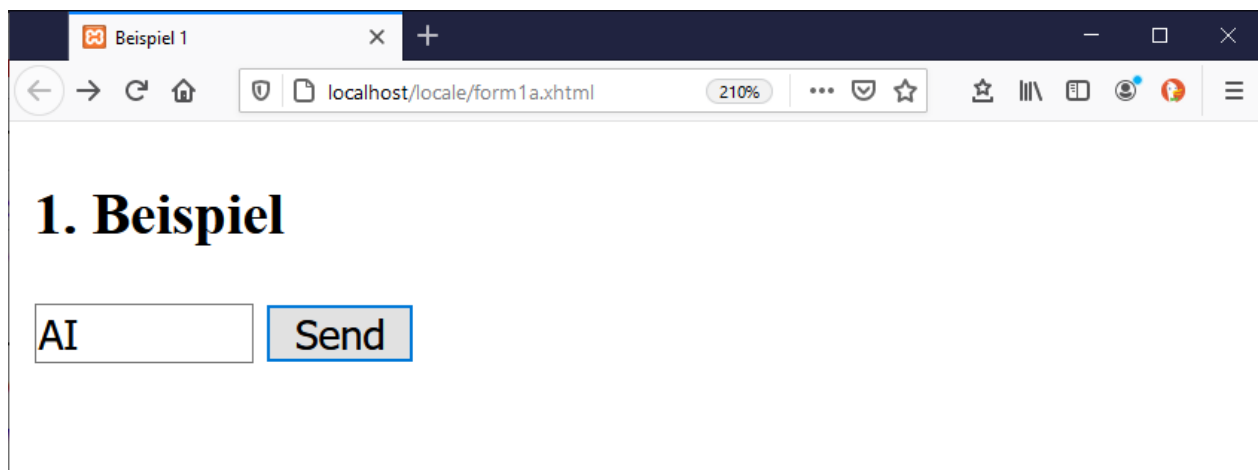


Abbildung 17 Erstes Beispiel eines einfachen Formulars

Die Serverseite muss natürlich eine dynamische Seite sein, also eine:

- PHP-Seite
- JSP-Seite
- Perl-Seite
- Node-js-Seite
- C#-Seite

Quellcode der ersten Seite: form1b.php

```
<body>
  <h3>Mein erstes PHP-Beispiel</h3>
  <?php
    echo 'hallo Text';
  ?>
</body>
```

Im obigen Beispiel werden noch keine Parameter abgefragt, aber der Text in der echo-Anweisung wird ausgegeben.

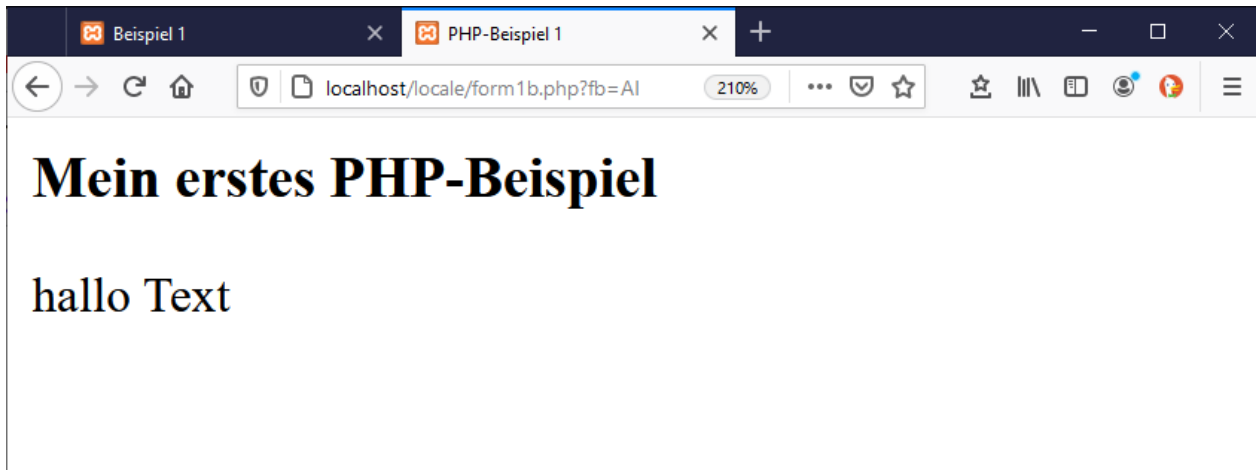


Abbildung 18 Erstes Beispiel eines einfachen Formulars (Serverseite)

Parameter-Übergabe: `http://localhost/locale/form1b.php?fb=AI`

5.3 GET / SET

5.3.1 Get

Bei der Methode „Get“ werden die Daten als zusätzlichen Text an den neuen Skriptnamen angehängt. Dies geschieht mittels Fragezeichen. Alle Parameter können dann im Skript als Variable abgerufen werden.

5.3.2 Post

Die Methode „Post“ übermittelt die Daten direkt in speziellen Variablen. Diese Variante ist sinnvoll bei der Übertragung größerer Datenmengen.

5.4 Parameter der Datenübertragung

- Das Fragezeichen starten den Parameterabschnitt.
- Daten werden durch ein Ampersand (&) voneinander getrennt.
- Jedes UI-Element hat einen Abschnitt mit Key und Value
 - Name = Datenwert
- Leerzeichen werden ersetzt durch ein Pluszeichen (+)
- Alle Sonderzeichen werden hexadezimal dargestellt (%20)
- Alle Zeichen, die in diesen Regeln als Steuerzeichen vorkommen (also &, +, = und %) werden ebenfalls hexadezimal umschrieben, und zwar genau so wie Zeichen außerhalb von ASCII.

Beispiel mit den Daten „Ute Müller“:

`http://mwilhelm.de/php/test.php?userid=Ute+M%FCller&password=geheim`

in neueren Browser werden die deutschen Umlaute korrekt dargestellt.

`http://mwilhelm.de/php/test.php?userid=Ute+Müller&password=geheim`

+ Leerzeichen
%FC ü ASCII-Zeichen 252

5.5 Formular-Elemente

5.5.1 HTML4-Elemente

Type	Bedeutung
label	Hilfselement, nicht sichtbar, Zuordnung zum Element
text	Texteingabefeld (einzeilig)
password	Texteingabefeld mit verdeckter Eingabe
checkbox	Anwahlschalter
radiobutton	Auswahlschalter
select	ComboBox oder mehrzeilige Liste
hidden	verstecktes Textelement,(für alle Elemente, Session)
textarea	mehrzeiliger Editor
image	anwählbares Bild in der Zeile, Ersatz für einen Schalter
submit	Schalter zum Abschieken
reset	Schalter zum Löschen der Eingaben
button	Normaler Schalter
file	Eingabefeld mit Schalter, Dateiauswahl

5.5.2 Neue HTML5-Elemente

Type	Bedeutung
nav	Navigator-Element
datetime	Inhalt ist ein Datum
summary	Summary ist die Zusammenfassung
details	Mit „details“ erhält man erweiterte Blöcke
email	Eintragen einer E-Mailadresse
url	Eintragen einer URL
number	Eintragen einer ganzzahligen Zahl
range	Slider ohne Angabe des aktuellen Wertes
color	Farbauswahl
search	Sucheingabe
day	Eingabe für eine Tag
month	Eingabe für einen Monat
year	Eingabe für ein Jahr
datetime	Eingabe für eine Datum und Zeiteingabe
date	Eingabe für ein Datum
week	Eingabe für eine Woche
time	Eingabe für eine Uhrzeit

5.5.3 label

Ein Labelelement umfasst ein echtes UI-Element mit dem Ziel, dass es, wenn das Label-Element angeklickt wird, das UI-Element aktiviert wird. Dabei sind zwei Techniken zu unterscheiden:

a) UI-Element ist innerhalb des Labelblocks

```
<label> Username  
  <input type="text" name="fb" size="8" value="Ute Müller"/>  
</label>
```

b) UI-Element ist außerhalb des Labelblocks.

Diese Fall kann auftreten, wenn man mit einem Grid- oder Flexlayout arbeitet. Dann sind die einzelnen UI-Elemente mit einem div-Block versehen und können so nicht adressiert werden.

```
<label for="userName2" >  
  Username  
</label>  
  
<input type="text" id="userName2" name="fb" value="Ute Müller"/>
```

Der Trick liegt nun darin, dass man im Label das for-Attribut mit einem ID versieht. Das dazu gehörige UI-Element muss dann die gleiche ID erhalten.

5.5.4 Text

Minimalform:

```
<input type="text" name="nachname" />
```

In der Minimalform muss man das Attribut „type“ und „name“ angeben.

Weitere Attribute:

- value="Vorgabetext" Vorgabewert
- placeholder="Vorgabetext(placeholder)" Text, der angezeigt wird, wenn das Element leer ist.
- size="12" Größe in „Buchstaben“ (ungefähr)
- autofocus="autofocus"
- required="required"
- pattern="[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}"

Mit required-Attribut:

```
<input type="text"  
  name="name1"  
  placeholder="Vorgabetext(placeholder)"  
  value="Vorgabetext"  
  size="12"  
  autofocus="autofocus"  
  required="required"  
>
```

Mit required- und pattern-Attribut:

```
<input type="text"
  name="name1"
  placeholder="Vorgabetext (placeholder)"
  pattern="[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}"
  value="Vorgabetext" size="12"
  autofocus="autofocus"
  required="required"
/>
```

Weitere Pattern-Beispiel im Kapitel 6, Seite 88.

CSS:

```
input:required {
  border: solid 2px black;
}
input:invalid {
  border: solid 2px red;
}
```

5.5.5 Password

Zeigt die eingegebenen Zeichen mit Punkten an.

5.5.6 Checkbox

Eine CheckBox zeigt meistens eine optionale Auswahl an.

WICHTIG:

- **Nur gesetzte Checkboxes werden übertragen**

Beispiel:

```
<input type="checkbox" name="AutoCAD"/> AutoCAD
```

5.5.7 Radiobutton / Auswahlhalter

Mit mehreren RadioButton kann aus einer Menge von Optionen ausgewählt werden. Im Gegensatz zur CheckBox muss man eine RadioButton auswählen. Sinnvoll ist es also, am Anfang mindestens eine zu setzen.

WICHTIG:

- **Die Namen der Radiobutton müssen alle gleich sein.**
- **Der name ist die Gruppe der Radiobutton**

Beispiel:

```

☒

```

Das Attribut „value“ ist wichtig, da dieser Wert dem Server übergeben wird.

5.5.8 select

Diese UI-Element fungiert entweder als ComboBox oder mehrzeilige Liste. Hier ist es wichtig, das man in der Definition den Value-Eintrag nicht vergisst.

Beispiel:

```

<p>Wählen Sie eines der Objekte aus: </p>
<select name="geomkoerper">
  <option value="idk" selected="selected"> Kegel </option>
  <option value="idz" > Zylinder </option>
  <option value="idq" > Quader </option>
</select>

```

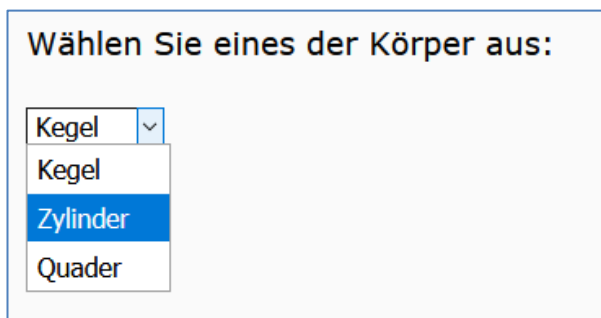


Abbildung 19 HTML-Element select (ComboBox)

Mit dem Setzen der unteren Attribute erweitert sich die ComboBox als Liste.

- size="6"
- multiple="multiple"

Beispiel:

```

<p>Wählen Sie eine oder mehrere der Flächen aus: </p>
<select name="geomkoerper" size="6" multiple="multiple" >
  <option value="flq" selected="selected"> Quadrat</option>
  <option value="flk" > Kreis </option>
  <option value="flr"> Rechteck </option>
  <option value="flrt"> Raute </option>
  <option value="fle"> Ellipse</option>
  <option value="flv"> Viereck </option>
  <option value="fl8"> 8-eck </option>
  <option value="fl12"> 12-eck </option>
</select>

```

Wählen Sie eine oder mehrere der Flächen aus:



Abbildung 20 HTML-Element select (Liste)

5.5.9 textarea

Eine Textarea speichert mehrzeiligen Text. Folgende Attribute existieren:

- rows="10"
- cols="50"
- wrap="soft"
 - hard
 - off

Beispiel:

```
<textarea name="editor" rows="10" cols="50" wrap="soft">
  Eingabe eines Textes in einem Editor
</textarea>
```

5.5.10 number

Das Number-Element vereinfacht die Eingabe von ganzzahligen Zahlen.

Folgende Attribute existieren:

- name
- value
- min
- max

Beispiel:

```
Zahl <input type="number" name="zahl" min="0" max="100" value ="12" />
```

5.5.11 hidden

Ein hidden-Element dient dazu, Werte zu speichern, die der Anwender nicht sehen soll. Das Problem in der Webprogrammierung ist ja, dass es keine globalen Variablen gibt, so dass der Transport von Werten, zum Beispiel eine Kundennummer, zur nächsten Seite nur über Hidden-Elemente oder Session-Variablen gehen kann.

Für allgemeine Daten reicht die hidden-Variante;

```
<input type="hidden" name="geheim" value="passwort" size="12"/>
```

Falls man Texte mit Zeilenumbruch speichern will, kann man nicht das Hiddenfeld nehmen, da dort die Zeilenumbrüche entfernt werden. Deshalb hier die Variante mit einer Textarea:

```
<textarea name="hiddeneditor" rows="10" cols="50" style="display:none;" wrap="off"> Hidden</textarea>
```

5.5.12 image

Dieses Element zeigt ein Bild an. Häufig kann man eine Link auf dieses Bild setzen. Mit Image-Coords kann man sogar unterschiedliche Bereiche innerhalb des Bildes als Link setzen. Man sollte aber den title setzen, damit Analyse-Programme den Inhalt erkennen können.

Wichtig:

- Man sollte entweder die Höhe oder die Breite setzen. Niemals beide zusammen!

Beispiel:

```


<a href="handyproperty.xhtml">
  
</a>
```

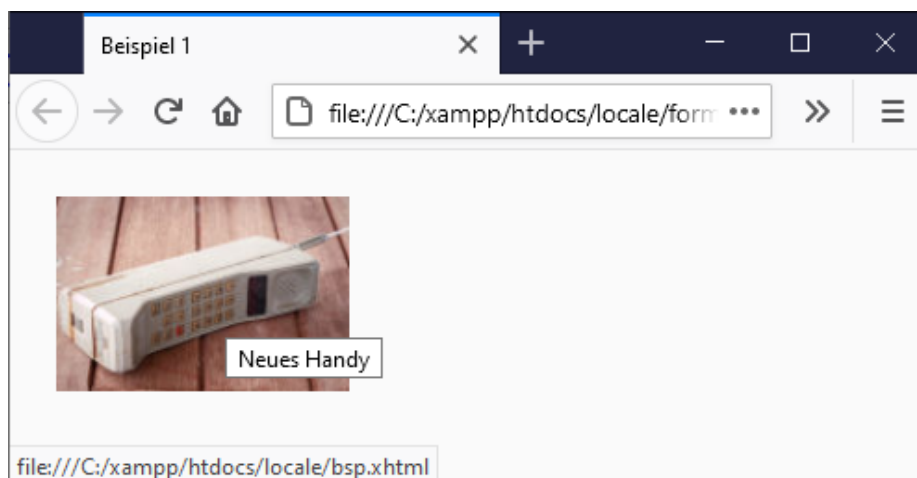


Abbildung 21 Image mit einme Linkl

5.5.13 submit

Mit diesem Schalter ruft die Seite aufgerufen, die im form-Abschnitt definiert wurde. Ist dort keine definiert, so wird die aktuelle Seite aufgerufen.

```
<input type="submit" value="Send" />
```

Mit dem „required“-Attribut in den text-Elementen, kann man recht gut die Plausibilität der Eingaben prüfen. Benötigt man zusätzliche Prüfungen, so kann ein spezielles Java-Script aufrufen:

```

<script type="text/javascript">
  // 
    "use strict";

    function validation( ) {
      "use strict";
      let form = document.forms[0];
      let kurzname = form.kurzname.value;
      alert(kurzname );
      if (kurzname == 'Abc') {
        alert('Bitte nicht "Abc" verwenden');
        return false;
      }
      else {
        return true;
      }

    } // validation
  // ]]&gt;
&lt;/script&gt;

&lt;body&gt;
  &lt;form method="get" onsubmit="return validation()"
action="form1a.xhtml" &gt;
    Kurzname
    &lt;input type="text" name="kurzname" placeholder="Kurzname"
    pattern="[A-Z][a-z]{1,3}" value="Abc" size="12"
    autofocus="autofocus" required="required"/&gt;
    &lt;br /&gt;
    &lt;input type="submit" value="Send" /&gt;
  &lt;/form&gt;
&lt;/body&gt;
</pre>
</div>
<div data-bbox="67 558 923 607" data-label="Text">
<p>Mit der Anweisung „onsubmit="return validation()" in der form-Definition wird die Javascript-Methode aufgerufen. Gibt diese Methode true zurück, wird die neue Seite aufgerufen. Ist der Rückgabewert false, so ändert sich nichts.</p>
</div>
<div data-bbox="67 671 197 688" data-label="Section-Header">
<h4>5.5.14 reset</h4>
</div>
<div data-bbox="67 708 583 726" data-label="Text">
<p>Dieser Schalter setzt alle Eingaben auf den Stand beim Starten.</p>
</div>
<div data-bbox="91 743 496 760" data-label="Text">
<pre>
&lt;input type="reset" value="Reset" /&gt;
</pre>
</div>
<div data-bbox="67 792 212 809" data-label="Section-Header">
<h4>5.5.15 button</h4>
</div>
<div data-bbox="67 828 274 845" data-label="Text">
<p>Der button-Schalter ruft</p>
</div>
<div data-bbox="905 936 938 955" data-label="Page-Footer">
<p>71</p>
</div>
```

5.5.16 file

Dieses HTML-Element erlaubt den Transfer von Dateien zum Server. Dazu muss man aber dann mit “Post“ arbeiten.

Beispiel:

```
<input type="file" name="filename" size="12" />
```



Abbildung 22 Anzeige des file-HTML-Elements

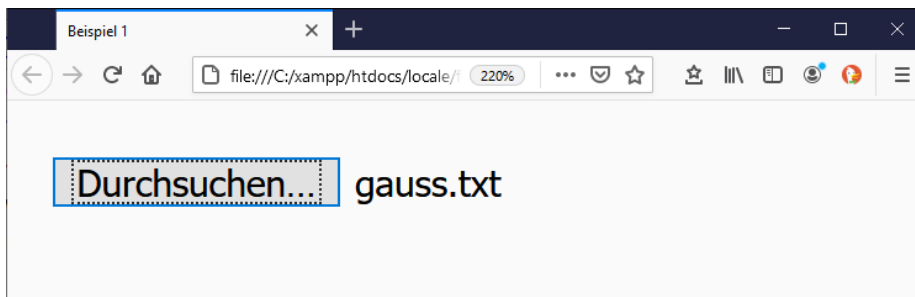


Abbildung 23 Anzeige der ausgewählten Datei

5.6 Sessions

Mit Hilfe der Session-Variablen kann man Informationen wie Kundennummer, Warenkorb pro Browser-Sitzung auf dem Server speichern. Dazu wird auf dem Server ein Eintrag in die globale SESSION-Liste eingetragen. Diese Variable ist zwar global, speichert aber die einzelnen Werte pro Aufruf!

Wichtig:

- **Auf jeder PHP-Seite muss ein PHP-Abschnitt mit „ session_start();“ ganz oben eingetragen werden!**

5.6.1 Sessions-Abfrage auf Inhalt

```
if(isset($_SESSION['username'])) {  
    $name = $_SESSION['username'];  
}  
else {  
    $name = 'unbekannt';  
}
```


5.6.2 Sessions-Variable setzen

```
$_SESSION['username'] = $name;
```

5.6.3 Beispielprogramm

Startprogramm:

```
<html>
  <head>
    <title> Startformular für eine Session-Beispiel </title>
  </head>

  <!-- http://mwilhelm.hs-harz.de/scripte/php/session_formular.html -->
<body>
  <form action="session_1.php" method="get">
    <h3> Startformular für eine Session-Beispiel</h3>

    Name: <input type="Text" name="name" value="?" size="12">
    <br />
    <br />
    <input type="submit" value="Senden" />
    <input type="reset" value="Löschen"/><br />

  </form>
</body>
</html>
```

1. PHP-Seite (session_1.php):

```
<?php
  session_start(); //Ganz wichtig
?>

<html>
  <head>
    <title> Seite der 1. PHP-Seite </title>
  </head>

  <body>
    <h2>Seite der 1. PHP-Seite</h2>

    <?php
      // bestimmen, ob schon registriert
      $name = $_GET['name'];

      if(!isset($name)) {
        $name = "Gast";
      }

      //Sessionsvariable registrieren
      $_SESSION['username'] = $name;
```

```

//Text ausgeben
echo "Sie heißen $name <br>\n";
echo "<a href=\"session_2.php\">Weiter zur 2. Seite</a>\n";
?>
</body>
</html>

```

2. PHP-Seite (session_2.php):

```

<?php
    session_start(); //Ganz wichtig
?>

<html>
<head>
    <title> Seite der 2. PHP-Seite </title>
</head>

<body>
    <h2>Seite der 2. PHP-Seite</h2>

    <?php
        //In $name den Wert der Sessionvariablen speichern
        $name = $_SESSION['username'];

        //Text ausgeben
        echo "Sie heißen immer noch: $name\n";
    ?>

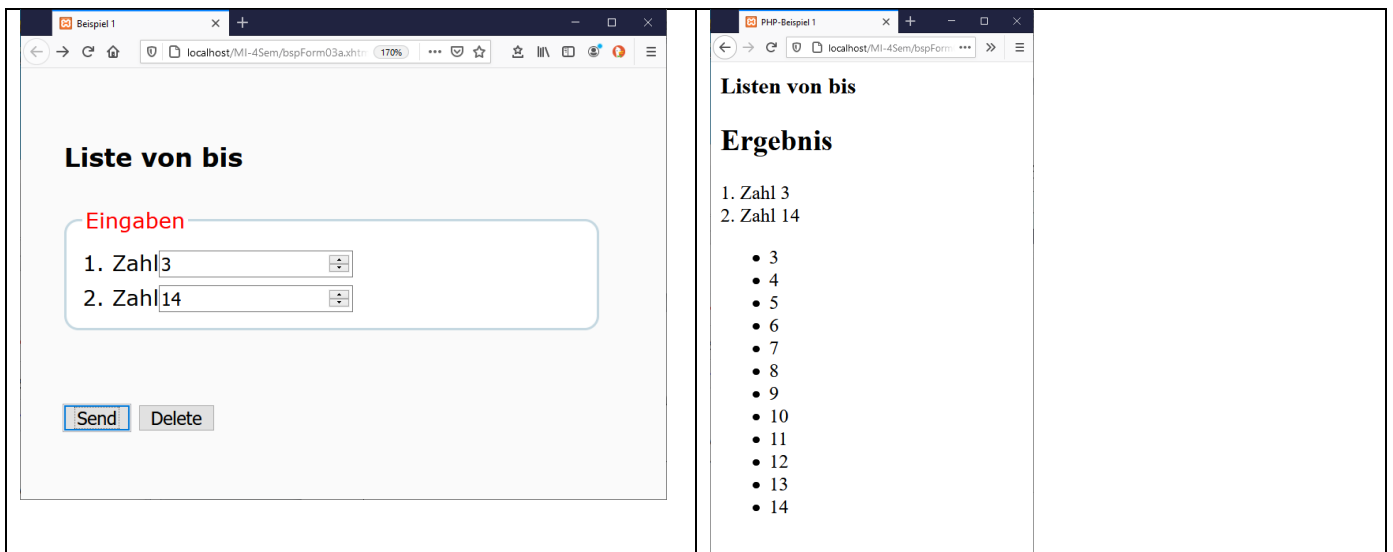
</body>
</html>

```

5.7 PHP-Formulare-Beispiele

In diesem Kapitel werden einige einfache Beispiele mit PHP aufgezeigt. Es gibt immer zwei Seiten. Auf der ersten Seite können die Daten in Formularen eingegeben werden. Diese Daten werden an die zweite Seite per „GET“ gesendet und dann dort verarbeitet.

5.7.1 Anzeige einer Liste von bis



5.7.1.1 Quellcode der Formular-Seite

```
<body>
  <h3>Liste von bis</h3>
  <form method="get" action="bspForm03b.php" >
    <fieldset>
      <legend>Eingaben</legend>
      1. Zahl
      <input type="number" name="zahl1" min="1" max="100" value ="3" />
      <br />
      2. Zahl
      <input type="number" name="zahl2" min="1" max="100" value ="14" />
    </fieldset>
    <br />
    <br />
    <input type="submit" value="Send" />
    <input type="reset" value="Delete"/>
  </form>
</body>
```

Eingegeben werden zwei Zahlen, die den Server übergeben werden und dann daraus eine Liste erstellt wird.

Link zur zweiten Seiten:

<http://localhost/MI-4Sem/bspForm03b.php?zahl1=3&zahl2=14>

5.7.1.2 Quellcode der Server-Seite

```
<body>
  <h3>Listen von bis</h3>
  <?php
    // Variablen
    $ok = true;
```

```

$zahl1 = 0;
$zahl2 = 0;
$error = '';

// Check zahl1
if ( isset($_GET['zahl1']) ){ // Ist gesetzt?
    $str_zahl1 = $_GET['zahl1'];
    if (is_numeric($str_zahl1)) {
        $zahl1 = intval ( $str_zahl1, 10 );
        if ($zahl1<0) {
            $ok = false;
            $error.="Error: the variable 'zahl1' must greater zero<br />";
        }
    }
    else {
        $ok = false;
        $error.="Error in number conversion: the variable 'zahl1' is not any
number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl1' are not passed<br />";
}

// Check zahl2
if ( isset($_GET['zahl2']) ){ // Ist gesetzt?
    $str_zahl2 = $_GET['zahl2'];
    if (is_numeric($str_zahl2)) {
        $zahl2 = intval ( $str_zahl2, 10 );
        if ($zahl2<0) {
            $ok = false;
            $error.="Error: the variable 'zahl2' must greater zero<br />";
        }
    }
    else {
        $ok = false;
        $error.="Error: the variable 'zahl2' is not any number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl2' are not passed<br />";
}

if ($ok) {
    // falsche Reihenfolge?
    if ($zahl1>$zahl2) {
        $dummy = $zahl1;
        $zahl1 = $zahl2;
        $zahl2 = $dummy;
    }
} // if

if ($ok) {
    echo '<h2>Ergebnis</h2>';
    echo "1. Zahl $zahl1<br />";
    echo "2. Zahl $zahl2<br />";
}

```

```
$s='<ul>';
for ($i=$zahl1; $i<=$zahl2; $i++) {
    $s.='<li>' . $i . '</li>';
}
$s.='</ul>';
echo $s;
}
else {
    echo '<h2>Fehler</h2>';
    echo $error;
}

?>
</body>
```

5.7.2 Berechnen einer Addition oder einer Subtraktion

The image shows two browser windows side-by-side. The left window, titled 'Beispiel 1', displays a web form titled 'Addition und Subtraktion'. The form has two sections: 'Eingaben' (Inputs) and 'Operation'. In the 'Eingaben' section, there are two text input fields: '1. Zahl' with the value '12.5' and '2. Zahl' with the value '33.5'. In the 'Operation' section, there are two radio buttons: 'Addition' (which is selected) and 'Subtraktion'. At the bottom of the form are two buttons: 'Send' and 'Delete'. The right window, titled 'PHP-Beispiel 1', shows the result of the form submission. It displays the title 'Addition / Subtraktion' followed by the input values: '1. Zahl 12.5', '2. Zahl 33.5', and the calculated result: 'Ergebnis: 46'.

5.7.2.1 Quellcode der Formular-Seite

```
<body>
  <h3>Addition und Subtraktion</h3>
  <form method="get" action="bspForm04b.php" >
    <fieldset>
      <legend>Eingaben</legend>
      1. Zahl
      <input type="text" name="zahl1" value="12.5" size="12" />
      <br />
      2. Zahl<input type="text" name="zahl2" value="33.5" size="12" />
    </fieldset>
    <br />
    <fieldset>
      <legend>Operation</legend>
      <!-- RadioButton: name ist gleich GRUPPE -->
      <input type="radio" name="rbop" value="add" checked="checked" />
      Addition<br />
      <input type="radio" name="rbop" value="sub" /> Subtraktion
    </fieldset>
    <br />
    <input type="submit" value="Send" />
    <input type="reset" value="Delete"/><br />
  </form>
</body>
```

Eingegeben werden zwei Zahlen und eine Auswahl der mathematischen Operation, die den Server übergeben werden. Die Daten werden überprüft und dann wird das Ergebnis ausgerechnet und ausgegeben.

Link zur zweiten Seiten:

<http://localhost/MI-4Sem/bspForm04b.php?zahl1=12.5&zahl2=33.5&rbop=add>

5.7.2.2 Quellcode der Server-Seite

```
<body>
  <h3>Addition / Substraktion</h3>
<?php
  // http://localhost/MI-
  4Sem/bspForm04b.php?zahl1=12.5&zahl2=12.44&rbop=add
  $ok = true;
  $zahl1 = 0;
  $zahl2 = 0;
  $op = "-";
  $error = '';

  // Check zahl1
  if ( isset($_GET['zahl1']) ){    // gib es einen Brief mit der Aufschrift
zahl1
    $str_zahl1 = $_GET['zahl1'];
    if (is_numeric($str_zahl1)) {
      $zahl1 = floatval ( $str_zahl1);
    }
    else {
      $ok = false;
      $error.="Error: the variable 'zahl1' is not any number<br />";
    }
  }
  else {
    $ok = false;
    $error.="Error: The param 'zahl1' are not passed<br />";
  }

  // Check zahl2
  if ( isset($_GET['zahl2']) ){    // Ist gesetzt?
    $str_zahl2 = $_GET['zahl2'];
    if (is_numeric($str_zahl2)) {
      $zahl2 = floatval ( $str_zahl2);
    }
    else {
      $ok = false;
      $error.="Error: the variable 'zahl2' is not any number<br />";
    }
  }
  else {
    $ok = false;
    $error.="Error: The param 'zahl2' are not passed<br />";
  }

  // Check Addition oder Subtraktion
  if ( isset($_GET['rbop']) ){    // Ist gesetzt?
    $op = $_GET['rbop'];
    // Abfrage ob add oder sub
    if ($op=='add' || $op=='sub'){
    }
    else {
      $ok = false;
      $error.="Error: The param 'rbop' must add or sub. Value: $op<br />";
    }
  }
```

```

    }
}
else {
    $ok = false;
    $error.="Error: The param 'rbop' are not passed<br />";
}

if ($ok) {
    echo "1. Zahl $zahl1<br />";
    echo "2. Zahl $zahl2<br />";
    $erg = -1;
    if ($op=='add') {
        $erg = $zahl1+$zahl2;
        echo "Ergebnis: $erg<br />";
    }
    else if ($op=='sub'){
        $erg = $zahl1-$zahl2;
        echo "Ergebnis: $erg<br />";
    }
    else {
        echo "Vergessen die Operation zu implementieren: $op<br />";
    }
}
else {
    echo '<h2>Fehler</h2>';
    echo $error;
}

?>

</body>

```


5.7.3 Eingabe mit einer Liste (Auswahl mehrerer Elemente)

Auf der ersten Seite wird eine Zahl eingegeben und in einer Liste können dann die mathematischen Operationen ausgewählt werden. Durch das Attribut „required“ müssen korrekte Werte in den beiden Eingabeelementen vorhanden sein.

The image shows two browser windows side-by-side. The left window, titled 'Beispiel 1', displays a web form titled 'Berechnen von Funktionswerten'. It has an input field labeled 'Zahl' with the value '12' and a dropdown menu labeled 'Auswahl der math. Operation' with options: Wurzel, Quadrat, Kubik, Kehrwert, and Fakultät. Below the form are 'Send' and 'Delete' buttons. The right window, titled 'PHP-Beispiel 1', shows the results of the calculation: 'Zahl: 12', 'Math. Operation: Wurzel Ergebnis: 3.4641016151378', 'Math. Operation: Kubikzahl Ergebnis: 1728', 'Math. Operation: Kehrwert Ergebnis: 0.0833333333333333', and 'Math. Operation: Fakultät Ergebnis: 479001600'.

5.7.3.1 Quellcode der Formular-Seite

```
<body>
  <h3>Berechnen von Funktionswerten</h3>
  <form method="get" action="bspForm07b.php" >
    <fieldset>
      <legend>Eingaben</legend>
      Zahl <input type="number" name="zahl" min="0" max="100"
        placeholder="Zahl für die Berechnung" size="30"
        value="12" autofocus="autofocus" required="required"/>
      <br />
      Auswahl der math. Operation<br />
      <select id="mathop" name="mathop[]"
        autofocus="autofocus" required="required"
        size="5" multiple="multiple" >
        <option value="sqrt"> Wurzel</option>
        <option value="x2"> Quadrat </option>
        <option value="x3"> Kubik </option>
        <option value="1x"> Kehrwert</option>
        <option value="fak"> Fakultät </option>
      </select>
    </fieldset>
    <br />
    <input type="submit" value="Send" />
    <input type="reset" value="Delete"/><br />
  </form>
</body>
```

Beachten:

Das select-Element hat die Arrayklammer: `<select id="mathop" name="mathop[]"`

Link zur zweiten Seiten:

[http://localhost/MI-4Sem/bspForm07b.php?zahl=12&mathop\[\]=sqrt&mathop\[\]=x3&mathop\[\]=1x&mathop\[\]=fak](http://localhost/MI-4Sem/bspForm07b.php?zahl=12&mathop[]=sqrt&mathop[]=x3&mathop[]=1x&mathop[]=fak)

5.7.3.2 Quellcode der Server-Seite

```
<body>
  <h3>Berechnen von Funktionswerten</h3>
<?php
  // bspForm07b.php?zahl=12&mathop[]=x2&mathop[]=1x
  $ok = true;
  $zahl = 0;
  $mathops = array();
  $error = '';

  // Check des Parameters zahl
  if ( isset($_GET['zahl']) ){ // Ist gesetzt?
    $str_zahl = $_GET['zahl'];
    if (is_numeric($str_zahl)) {
      $zahl = intval ( $str_zahl, 10 );
    }
    else {
      $ok = false;
      $error.="Error in number conversion: the variable 'zahl' is not any
number<br />";
    }
  }
  else {
    $ok = false;
    $error.="Error: The param 'zahl' are not passed<br />";
  }

  // Check der math. Operation
  // the name of the select ui-element must have a '[]'
  if ( isset($_GET['mathop']) ){ // Is the list gesetzt?
    foreach ($_GET['mathop'] as $op) {
      $mathops[] = $op;
    }
    $anz = count($mathops);
    if ($anz==0) {
      $ok =false;
      $error.="Error: Der Parameter 'mathop' wurde nicht nicht
ausgewaehlt<br />";
    }
  }
  else {
    $ok =false;
    $error.="Error: Der Parameter 'mathop' wurde nicht übergeben<br />";
  }

  if ($ok) {
    echo "Zahl: $zahl<br />";
    foreach ($mathops as $op) {
      $erg='NaN';
```

```

$ausgabeOp='';

switch ($op) {
    case 'sqrt':
        $ausgabeOp='Wurzel';
        $erg=sqrt($zahl);
        break;
    case 'x2':
        $ausgabeOp='Quadratzahl';
        $erg=$zahl*$zahl;
        break;
    case 'x3':
        $ausgabeOp='Kubikzahl';
        $erg=$zahl*$zahl*$zahl;
        break;
    case '1x':
        $ausgabeOp='Kehrwert';
        $erg=1.0/$zahl;
        break;
    case 'fak':
        $ausgabeOp='Fakultät';
        $erg=1;
        for ($i=2; $i<=$zahl; $i++) {
            $erg*=$i;
        }
        break;
    default:
        echo "Nicht implementierte Operation $op<br />";
        break;
} // switch
echo "Math. Operation: $ausgabeOp          Ergebnis: $erg<br />";
}

}
else {
    echo '<h2>Fehler</h2>';
    echo $error;
}

?>
</body>

```

Besonderheit:

- Da es mehrere mathematische Operationen geben kann, werden die übergebenen Parameter in das Array „\$mathops“ eingetragen.

5.7.4 Eingabe mit einer Liste von Checkboxes(Auswahl mehrerer Elemente)

Auf der ersten Seite wird eine Zahl eingegeben und in einer „Checkbox-Liste“ können dann die mathematischen Operationen ausgewählt werden. Durch das Attribut „required“ müssen korrekte Werte in den beiden Eingabeelementen vorhanden sein. Bei den Checkbox ist das aber ein Problem, da mindestens eine aktiviert sein muss. **Mit dem Attribut „required“ müssten alle CheckBoxen aktiviert werden!** Abhilfe ist eine Javascript-Methode, die VOR dem Aufrufen des Servers aufgerufen wird. Nur wenn die Methode „true“ zurückgibt, wird die Serverseite aufgerufen. Im Validierungsscript muss dann nach den CheckBox mit „mathop[]“ gesucht werden.

The image shows two browser windows side-by-side. The left window, titled 'Beispiel 1', displays a form titled 'Berechnen von Funktionswerten'. It has an input field labeled 'Eingaben' with the value '12' and a section 'Auswahl der math. Opearation' (note the typo) containing five checkboxes: 'Wurzel' (checked), 'Quadrat' (unchecked), 'Kubik' (checked), 'Kehrwert' (checked), and 'Fakultät' (checked). At the bottom are 'Send' and 'Delete' buttons. The right window, titled 'PHP-Beispiel 1', shows the results of the calculation: 'Zahl: 12', 'Math. Operation: Wurzel Ergebnis: 3.4641016151378', 'Math. Operation: Kubikzahl Ergebnis: 1728', 'Math. Operation: Kehrwert Ergebnis: 0.0833333333333333', and 'Math. Operation: Fakultät Ergebnis: 479001600'.

5.7.4.1 Quellcode der Formular-Seite

```
<script type="text/javascript">
// 
"use strict";

function validation( ) {
    "use strict";
    let form = document.forms[0];
    // tests
    let anz=0;
    let checkboxes=document.getElementsByName('mathop[]');
    for (let i=0; i&lt;checkboxes.length; i++) {
        if (checkboxes[i].checked) {
            anz++;
        }
        if (anz==0) {
            alert('Bitte mindestens eine Checkbox anklicken');
            return false;
        }
        else {
            return true;
        }
    }
} // validation

// ]]&gt;</pre></div><div data-bbox="905 937 940 955" data-label="Page-Footer"><p>84</p></div>
```

```

</script>

<body>
  <h3>Berechnen von Funktionswerten</h3>
  <form method="get" onsubmit="return validation()"
action="bspForm08b.php" >
    <fieldset>
      <legend>Eingaben</legend>
      Zahl <input type="number" name="zahl" min="0" max="100"
        placeholder="Zahl für die Berechnung" size="30"
        value="12" autofocus="autofocus" required="required"/>
    </fieldset>
    <br />

    <fieldset>
      <legend>Auswahl der math. Opearation</legend>
      <input type="checkbox" name="mathop[]" value="sqrt" /> Wurzel
      <br />

      <input type="checkbox" name="mathop[]" value="x2" /> Quadrat
      <br />

      <input type="checkbox" name="mathop[]" value="x3" /> Kubik
      <br />

      <input type="checkbox" name="mathop[]" value="1x" /> Kehrwert
      <br />

      <input type="checkbox" name="mathop[]" value="fak" /> Fakultät
      <br />
    </fieldset>
    <br />
    <input type="submit" value="Send" />
    <input type="reset" value="Delete"/><br />

  </form>

</body>
</html>

```

Beachten:

Die einzelnen CheckBoxen haben den Namen mit einer Arrayklammer: name="mathop[]"

Link zur zweiten Seiten:

[http://localhost/MI-4Sem/bspForm08b.php?zahl=12&mathop\[\]=sqrt&mathop\[\]=x3&mathop\[\]=1x&mathop\[\]=fak](http://localhost/MI-4Sem/bspForm08b.php?zahl=12&mathop[]=sqrt&mathop[]=x3&mathop[]=1x&mathop[]=fak)

Die Serverseite entspricht exakt der Variante des vorigen Beispiels.

5.7.4.2 Quellcode der Server-Seite

```
<body>
  <h3>Berechnen von Funktionswerten</h3>
<?php
  // bspForm07b.php?zahl=12&mathop[]=x2&mathop[]=1x
  $ok = true;
  $zahl = 0;
  $mathops = array();
  $error = '';

  // Check des Parameters zahl
  if ( isset($_GET['zahl']) ) { // Ist gesetzt?
    $str_zahl = $_GET['zahl'];
    if (is_numeric($str_zahl)) {
      $zahl = intval ( $str_zahl, 10 );
    }
    else {
      $ok = false;
      $error.="Error in number conversion: the variable 'zahl' is not any
number<br />";
    }
  }
  else {
    $ok = false;
    $error.="Error: The param 'zahl' are not passed<br />";
  }

  // Check der math. Operation
  // the name of the select ui-element must have a '[]'
  if ( isset($_GET['mathop']) ) { // Is the list gesetzt?
    foreach ( $_GET['mathop'] as $op ) {
      $mathops[] = $op;
    }
    $anz = count($mathops);
    if ($anz==0) {
      $ok =false;
      $error.="Error: Der Parameter 'mathop' wurde nicht nicht
ausgewaehlt<br />";
    }
  }
  else {
    $ok =false;
    $error.="Error: Der Parameter 'mathop' wurde nicht übergeben<br />";
  }

  if ($ok) {
    echo "Zahl: $zahl<br />";
    foreach ($mathops as $op) {
```

```

$erg='NaN';
$ausgabeOp='';

switch ($op) {
    case 'sqrt':
        $ausgabeOp='Wurzel';
        $erg=sqrt($zahl);
        break;
    case 'x2':
        $ausgabeOp='Quadratzahl';
        $erg=$zahl*$zahl;
        break;
    case 'x3':
        $ausgabeOp='Kubikzahl';
        $erg=$zahl*$zahl*$zahl;
        break;
    case '1x':
        $ausgabeOp='Kehrwert';
        $erg=1.0/$zahl;
        break;
    case 'fak':
        $ausgabeOp='Fakultät';
        $erg=1;
        for ($i=2; $i<=$zahl; $i++) {
            $erg*=$i;
        }
        break;
    default:
        echo "Nicht implementierte Operation $op<br />";
        break;
} // switch
echo "Math. Operation: $ausgabeOp          Ergebnis: $erg<br />";
}

}
else {
    echo '<h2>Fehler</h2>';
    echo $error;
}

?>
</body>

```

Besonderheit:

- Da es mehrere mathematische Operationen geben kann, werden die übergebenen Parameter in das Array „\$mathops“ eingetragen.

6 Pattern

Kürzelbeschreibung:

.	Beliebiges Zeichen
\.	Punkt
[A-Z]	ein Zeichen aus A bis Z
[0-9]	ein Zeichen aus 0 bis 9
[a-z]	ein Zeichen aus a bis z
[^A-Z]	Negation, kein Zeichen aus A bis Z

Vordefinierte Zeichenklassen

\d	Beliebiges Ziffernzeichen; identisch mit [0-9]
\D	Alle Zeichen außer Ziffern; identisch mit [^0-9]
\s	Leerraumzeichen (White Space); identisch mit [\t\n\r\x0B\f]
\S	Alle Zeichen außer Leerraum; identisch mit [^\s]
\w	Jedes alphanummerische Zeichen; identisch mit [a-zA-Z_0-9]
\W	Alle Zeichen außer Jedes alphanummerische; identisch mit [^\w] oder [^a-zA-Z0-9]
(?<=0)	Blockfunktion, vorne ein Null positive lookbehind
(?=0)	Blockfunktion, hinten ein Null positive lookahead
(?!0)	Blockfunktion, keine 0

Grenzbezüge:

^	Start einer Zeile oder Beginn des Textes
\$	Zeilenende oder Ende des Textes
\b	Wortgrenze
\B	Alles außer eine Wortgrenze
\A	Beginn des Textes
\G	The end of the previous match
\Z	Ende des Textes oder des Textabschlusses
\z	Ende des Textes

Mengenbeschreibungen:

{m,n}	Mindestens m, maximal n-Mal
{m}	Genau m-Elemente
{m,}	Mindestens m, maximal ist beliebig
{0,n}	Mindestens 0, maximal n-Mal
+	1,n (veraltet)
*	0,1,n (veraltet)

6.1 Beispiel Nachname

Ein Name:

```
pattern = "^[A-ZÜÄÖ][a-züäö]{1,29}$";
```

Bis zu vier Namen

```
pattern = "^[A-ZÜÄÖ][a-züäö]{1,29}([ ] [A-ZÜÄÖ][a-züäö]{1,29}){0,3}$";
```


6.2 Beispiel Straße

```
pattern = "^[A-ZÜÄÖ][a-züäöß]{1,29}([ ][A-ZÜÄÖ][a-züäöß]{1,29}){0,2}[ ][1-9][0-9]{0,3}([a-z]){0,1}$";
```

6.3 Beispiel Geburtsdatum

```
<input id="geburtsdatum" type="text"
pattern="^(
  31|
  30|
  0[1-9]|
  [12][0-9]|
  [1-9])\.(0[1-9]|
  1[012][1-9])\,((18|19|20)\d{2})\d{2}
)$">
```

6.4 Beispiel E-Mail

```
<input type="email" name="email"
pattern= "
[a-z0-9._%+-]{1,}
@
[a-z0-9.-]{1,}
\
[a-z]{2,3}
$ "
>
```

6.5 Beispiel Drei Großbuchstaben, dann drei Ziffern

```
[A-Z]{3}[0-9]{4}"
```

6.5.1 Beispiel IPv4

```
pattern="((^|\.)
((25[0-5])|
(2[0-4]\d)|
(1\d\d)|
([1-9]? \d)))
{4}
$"
```

6.6 Beispiel Paßwort:

```
pattern="(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![\.\n]) (?=.*[A-Z])(?=.*[a-z]).*$"
```

\w [a-zA-Z0-9_]

\W [^a-zA-Z0-9_] \W ist die Negation von \w

(?<=0) Blockfunktion, vorne ein Null positive lookbehind

(?=0) Blockfunktion, hinten ein Null positive lookahead

(?!0) Blockfunktion, keine 0

```
pattern="
```

(?=^.{8,}\$) Mindestens 8 Zeichen

(

(?=.*\d) Nicht nur aus Ziffern

(?=.*\W+) Es muss kein Wort da sein, also Sonderzeichen

)

(?![\.\n]) Keine Zeilenumbrüche oder Punkte

(?=.*[A-Z]) Mindestens ein großer Buchstabe

(?=.*[a-z]) Mindestens ein kleiner Buchstabe

.*\$"

Weitere Infomationen:

- https://de.wikipedia.org/wiki/Regulärer_Ausdruck
- <https://danielfett.de/2006/03/20/regulaere-ausdruecke-tutorial/>
- <https://support.google.com/a/answer/1371417?hl=de>

7 Abfragen der Parameter in PHP

Für die Abfrage der Parameter an den Server benötigt man folgende Daten und Funktionen

- \$_GET bzw. \$_POST
- isset
- is_numeric
- isint
- infloat
- intval
- floatval

7.1 Abfrage eines String Wertes

```
$nachname='';
$ok = true;
$error='';

if ( isset($_GET['nachname']) ) {    // Ist gesetzt?
    $nachname = $_GET['nachname'];
    // Prüfung mit Regex !
    if (strlen($nachname)==0) {
        $ok = false;
        $error.="Error: The param 'nachname' had the length of zero<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'nachname' are not passed<br />";
}

if ($ok) {
    // Action
}
else {
    echo "<h2>Fehler</h2>";
    echo $error;
}
```

Mit „isset“ wird geprüft, ob der Parameter „nachname“ übergeben wurde. Dabei ist es aber möglich, dass kein Wert übergeben wurde. Dieses wird im obigen Beispiel mit der Funktion „strlen“ abgeprüft. Wenn Fehler auftreten, wird die Variable \$ok auf false gesetzt und der Fehlertext an die Variable \$error rangehängt.

Nach der Plausibilitätskontrolle, die kann ja mehrere Parameter betreffen, wird die Variable \$ok abgeprüft und entweder eine Fehlermeldung ausgegeben oder die eigentliche Aktion ausgeführt.

7.2 Abfrage eines numerischen Wertes (int)

```
$zahl=0;
$ok = true;
$error='';
```

```

if ( isset($_GET['zahl']) ){          // Ist gesetzt?
    $str_zahl = $_GET['zahl'];
    if (is_int($str_zahl) ) {
        $zahl = intval($str_zahl, 10 );
        // optional wird eine Grenze abgeprüft
        if ($zahl<1 || $zahl>20) {
            $ok = false;
            $error.="Error: the variable 'zahl' muts be in 1 to 20<br />";
        }
    }
    else {
        $ok = false;
        $error.="Error: the variable 'zahl' is not any number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl' are not passed<br />";
}

if ($ok) {
    // Action
}
else {
    echo "<h2>Fehler</h2>";
    echo $error;
}

```

Mit „isset“ wird geprüft, ob der Parameter „zahl“ übergeben wurde. Dabei ist es aber möglich, dass kein oder ein nichtnumerischer Wert übergeben wurde. Dieses wird im obigen Beispiel mit der Funktion „is_int“ oder auch „is_numeric“ abgeprüft. Wenn Fehler auftreten, wird die Variable \$ok auf false gesetzt und der Fehlertext an die Variable \$error rangehängt. Mit „intval“ wird der „Zahlstring“ in eine echte Zahl umgewandelt. Danach kann, sollt man die logischen Grenzen der Zahl abprüfen.

Nach der Plausibilitätskontrolle, die kann ja mehrere Parameter betreffen, wird die Variable \$ok abgeprüft und entweder eine Fehlermeldung ausgegeben oder die eigentliche Aktion ausgeführt.

7.3 Abfrage eines numerischen Wertes (float)

```

$zahl=0;
$ok = true;
$error='';

if ( isset($_GET['zahl']) ){          // Ist gesetzt?
    $str_zahl = $_GET['zahl'];
    if (is_float($str_zahl) ) {
        $zahl = floatval($str_zahl, 10 );
        // optional wird eine Grenze abgeprüft
        if ($zahl<1.0 || $zahl>20.0) {
            $ok = false;
            $error.="Error: the variable 'zahl' muts be in 1 to 20<br />";
        }
    }
}

```

```

else {
    $ok = false;
    $error.="Error: the variable 'zahl' is not any number<br />";
}
}
else {
    $ok = false;
    $error.="Error: The param 'zahl' are not passed<br />";
}

if ($ok) {
    // Action
}
else {
    echo "<h2>Fehler</h2>";
    echo $error;
}

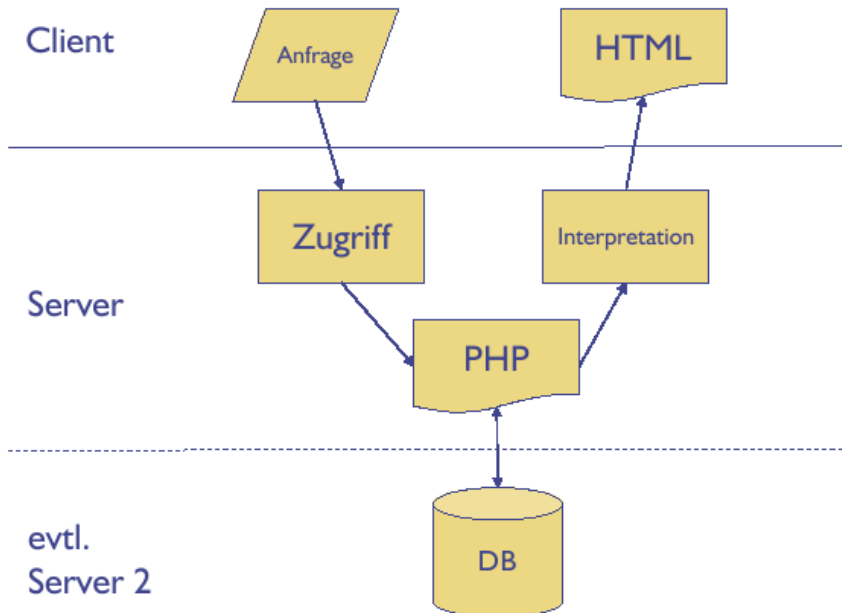
```

Mit „isset“ wird geprüft, ob der Parameter „zahl“ übergeben wurde. Dabei ist es aber möglich, dass kein oder ein nichtnumerischer Wert übergeben wurde. Dieses wird im obigen Beispiel mit der Funktion „**is_float**“ oder auch „is_numeric“ abgeprüft. Wenn Fehler auftreten, wird die Variable \$ok auf false gesetzt und der Fehlertext an die Variable \$error rangehängt. Mit „intval“ wird der „Zahlstring“ in eine echte Zahl umgewandelt. Danach kann, sollt man die logischen Grenzen der Zahl abprüfen.

Nach der Plausibilitätskontrolle, die kann ja mehrere Parameter betreffen, wird die Variable \$ok abgeprüft und entweder eine Fehlermeldung ausgegeben oder die eigentliche Aktion ausgeführt.

8 Datenbank MySQL

8.1 Datenbank Abfrage



Die folgenden Code-Beispiele verwenden den objektorientierten Syntax der verbesserten MySQL-Erweiterung (mysqli).

Dieser erfordert:

- ab PHP 5
- ab MySQL 4.1
- Vorteile:
 - Statt Funktionen werden Objektmethoden verwendet (Objektorientierung)
 - Dadurch entsteht eine stärkere Unabhängigkeit von der verwendeten Datenbank
 - Höhere Geschwindigkeit
- Select-Befehl
 - Gibt eine „Matrix“ als Ergebnis zurück
 - Spaltenweise: Tupel
 - Pro Tupel die Attribute
- Insert-, Update-, Delete-Befehl
 - Gibt die Anzahl der betroffenen Tupel zurück

8.2 Initialisierung

Initialisierung des Datenbank-Objekts:

```
// Initialisierung des Objekts
$db = new mysqli(host, user, password, database);

// Prüfen, ob auch alles geklappt hat
if ( mysqli_connect_errno() ) {
```

```
    die('Initialisierung der Datenbank ist gescheitert');
}
```

Besser, ist es, eine Methode für die Verbindung zu deklarieren:

```
function getConnection() {
    $servername='localhost';
    $username='root';
    $passwd='';
    $database='amacon';

    $conn = new mysqli($servername, $username, $passwd, $database);
    if ($conn->connect_error) {
        die('Connection failed: ' . $conn->connect_error);
    }
    return $conn;
}
```

8.3 Abfrage mit query

Absenden eines SQL-Befehls

8.3.1 Quellcode

```
$arr = array();
$conn = getConnection();
if ($conn->connect_error) {
    return $arr; // ein leeres Array wird zurückgegeben
}

$sql = "SET NAMES 'utf8'"; // ÄÖÜäöß Problematik
$result = $conn->query($sql);

$sql = "SELECT PINDEX, KURZBEZ, BEZ FROM TABELLE ORDER BY BEZ;";
$result = $conn->query($sql); // in result ist eine Matrix
if ($result) {
    // Schleife über alle "Zeilen" in fetch_array ist die aktuelle Zeile
    // row ist eine Hashtable ALSO: die Hashkey muss gleich sein
    while ( $row = $result->fetch_array() ) {
        $artikel = new Artikel();
        $artikel->pindex = intval( $row['PINDEX'] );
        $artikel->kurzbez = trim( $row['KURZBEZ'] );
        $artikel->bez = trim( $row['BEZ'] );
        $arr[] = $artikel;
    } // while
}
else {
    die('Die SQL Anweisung war fehlerhaft\n' . $sql);
}

return $arr;
```

8.3.2 Connection

Im ersten Teil wird eine Verbindung zur Datenbank erstellt. Im Fehler sollte eine Fehlermeldung zurückgegeben werden (siehe unten das Beispiel mit JSON).

```
$conn = getConnection();
```

8.3.3 Sonderzeichen

Um deutsche Sonderzeichen zu benutzen, muss man die Datenbank konfigurieren:

```
$sql = "SET NAMES 'utf8'"; // ÄÖÜüäöß Problematik
$result = $conn->query($sql);
```

8.3.4 Abfrage-Schleife

Im letzten Abschnitt werden in einer while-Schleife die einzelnen Zeilen aus der Matrix geholt und in einem Array von Objekten, hier Klasse Artikel, gespeichert.

```
$sql = "SELECT PINDEX, KURZBEZ, BEZ FROM TABELLE ORDER BY BEZ;";
$result = $conn->query($sql);
if ($result) {
    while ( $row = $result->fetch_array() ) {
        // Erstellen einer Instanz der Wrapper-Klasse
        $artikel = new Artikel();

        // Speichern der einzelnen Daten
        $artikel->pindex = intval( $row['PINDEX'] );
        $artikel->kurzbez = trim( $row['KURZBEZ'] );
        $artikel->bez = trim( $row['BEZ'] );

        // Speichern in die Ergebnisliste
        $arr[] = $artikel;
    } // while
}
else {
    die('Die SQL Anweisung war fehlerhaft\n' . $sql);
}

return $arr;
```

- In der while-Bedingung wird eine row gespeichert:
 - \$row = \$result->fetch_array() abrufen.
- Gibt z. B. ein SELECT mehrere Zeilen zurück, so erhalten wir mit dem Befehl jeweils nur eine Zeile. Rufen wir den Befehl erneut auf, so gibt er uns die nächste Zeile aus usw.
- Ist keine Zeile mehr vorhanden, so gibt fetch_array() FALSE zurück. Daher wird meist folgender Code verwendet:
- Dabei wird \$row ein Array mit den Rückgabewerten zugewiesen. Die Rückgabewerte sind normalerweise über den Namen und über ihre Position zugänglich.
 - **Dabei müssen die Keys der Hashtable exakt denen entsprechen, die in der Select-Anweisung angegeben wurden.**
 - SELECT PINDEX, KURZBEZ, BEZ
 - Also Haskeys:

- PINDEX
- KURZBEZ
- BEZ

•

Beispiel:

```
$sql = 'SELECT feld1, feld2 FROM tabelle';
if ( $result = $db->query($sql) )
{
    // Die Abfrage war erfolgreich!
    while ( $row = $result->fetch_array() )    {
        // Ausgabe einer Zeile
        echo $row['feld1'];
        echo $row['feld2'];
        echo $row[0];    // Gibt genau das Gleiche aus
        echo $row[1];    // Gibt genau das Gleiche aus
    }
}
else {
    // Die Abfrage war nicht erfolgreich
}
```

8.4 Sequence

MySQL hat eine automatische Incrementierung der PrimaryKeys. Manchmal benötigt man aber für komplexe Eingaben den PrimaryKeys bei einem Insert. Mit Hilfe der Sequence kann man VORHER eine eindeutige Nummer ermitteln und diese für einen neuen Datensatz benutzen.

```
function getDBSequence() {
    $conn = getConnection();
    if ($conn->connect_error) {
        return 0;
    }

    $sql = 'UPDATE sequence SET id=LAST_INSERT_ID(id+1)';
    $result = $conn->query($sql);
    if ( $result ){
        $sql = 'SELECT LAST_INSERT_ID()';
        $result = $conn->query($sql);
        if ( $result ){
            while ( $row = $result->fetch_array() ) {
                $sequence = intval($row[0]);
            } // while
        } // if
    }
    else {
        echo '2. sql ist false<br />';
    }
}
else {
    echo '1. sql ist false<br />';
    die('SQL-Anweisung gescheitert');
}
return $sequence;
}
```

8.5 Insert into

Beim Insert benötigt man den kompletten Datensatz. Man kann den PrimaryKey mit Hilfe der Sequence vorher oder in der Insaert-Methode holen und eintragen.

```
$conn = getConnection();
if ($conn->connect_error) {
    return false;
}

$customer->pindex = getDBSequence();

$sql = 'INSERT INTO customer(PINDEX, BEZ)' .
' VALUES( ' .
' ' . $customer->pindex . ', ' .
' ' . quoted($customer->bez) .
' )';

if ($conn->query($sql)) {
    $result = true;
}
else {
    $result = false;
}
return $result;
```

Im unteren Beispiel wurde ein kompletter JSON-Returnwert eingebaut.

8.6 Update

Beim Update benötigt man den kompletten Datensatz.

```
$conn = getConnection();
if ($conn->connect_error) {
    return false;
}

$sql = 'UPDATE customer ' .
' SET ' .
' CNR=' . $customer->cnr . ', ' .
' BEZ=' . quoted($customer->bez) .
' WHERE PINDEX=$customer->pindex ';

$resultSQL = $conn->query($sql);
$conn->close();

if ($resultSQL) {
    $result = true;
}
else {
    $result = false;
}
return $result;
```

Im unteren Beispiel wurde ein kompletter JSON-Returnwert eingebaut.

8.7 Delete

Beim Löschen eines Datensatzes benötigt man nur den PrimaryKey.

```
$conn = getConnection();
if ($conn->connect_error) {
    return false;
}

$sql = 'DELETE FROM customer ' . ' WHERE `PINDEX`=' . $pindex;
if ($conn->query($sql)) {
    $result = true;
}
else {
    $result = false;
}
$conn->close();
return $result;
```

Im unteren Beispiel wurde ein kompletter JSON-Returnwert eingebaut.

8.8 Kompletter Rahmen für eigene Datenbanken

8.8.1 SQL-Datenbank

```
create table `customer` (
    `PINDEX` integer not null,
    `bez` varchar(50) default '',
    constraint `PK_Customer` primary key(`pindex`)
);

create table sequence(
    id INT not null
);
insert into sequence values(30);

insert into customer(pindex, bez)
Values(11, 'Meier');

insert into customer(pindex, bez)
Values(12, 'Schulze' );

insert into customer(pindex, bez)
Values(13, 'Brandter');
```

8.8.2 Verwendete Klasse:

```
class Customer {
    public $pindex=0;
    public $bez='';
}
```

8.8.3 Rahmen

```
<?php

class Result1{
    public $isError=false;
    public $error='';
    public $liste = array();
}

class Result2{
    public $isError=false;
    public $error='';
    public $customer;
}

class Result3{
    public $isError=false;
    public $error='';
}

function quoted($item) {
    return '"' . $item . '\';
}

function getDBSequence() {
    $conn = getConnection();
    if ($conn->connect_error) {
        return 0;
    }
    $sql = 'UPDATE sequence SET id=LAST_INSERT_ID(id+1);';
    $result = $conn->query($sql);
    if ( $result ){
        $sql='SELECT * FROM `sequence` '; // SELECT LAST_INSERT_ID();
        $result = $conn->query($sql);
        if ( $result ){
            while ( $row = $result->fetch_array() ) {
                $sequence = intval($row[0]) ;
            } // while
        } // if
    }
    else {
        // die('SQL-Anweisung gescheitert');
        $sequence = 0;
    }
    return $sequence;
}
```

```

function getConnection() {
    $servername='localhost';
    $username='root';
    $passwd='';
    $database='amacon';
    $conn = new mysqli($servername, $username, $passwd, $database);
    if ($conn->connect_error) {
        die('Connection failed: ' . $conn->connect_error);
    }
    return $conn;
}

// Result isError, error, liste
function loadCustomersFromDB() {
    $result = new Result1();
    $conn = getConnection();
    if ($conn->connect_error) {
        $result->isError=true;
        $result->error = $conn->connect_error;
        return $result;
    }

    $sql = 'SELECT PINDEX, BEZ FROM customer';
    $resultSQL = $conn->query($sql);
    if (!$resultSQL) {
        $result->isError=true;
        $result->error = 'Fehlerhafte SQL-Anweisung<br />' . $conn->error .
'<br />' . $sql;
    }
    else {
        while ($row=$resultSQL->fetch_array()) {
            $customer = new Customer();
            $customer->pindex = intval($row['PINDEX']);
            $customer->bez = $row['BEZ'];
            $result->liste[] = $customer;
        } // while
        $resultSQL->close();
    }
    $conn->close();
    return $result;
} // loadCustomersFromDB

function loadCustomerFromDB($pindex) {
    $result = new Result2();

    $conn = getConnection();
    if ($conn->connect_error) {
        $result->isError=true;
        $result->error = $conn->connect_error;
        return $result;
    }

    $sql = 'SELECT PINDEX, BEZ FROM customer WHERE pindex=' . $pindex;
    $resultSQL = $conn->query($sql);

```

```

    if (!$resultSQL) {
        $result->isError=true;
        $result->error = 'Fehlerhafte SQL-Anweisung<br />' . $conn->error .
'<br />' . $sql;
    }
    else {
        $result->customer = new Customer();
        while($row=$resultSQL->fetch_array()) {
            $result->customer->pindex=intval($row['PINDEX']);
            $result->customer->bez=$row['BEZ'];
        } // while
        $resultSQL->close();
    } // else
    $conn->close();
    return $result;
} // loadCustomersFromDB

function updateCustomerDB($customer) {
    $result = new Result3();
    if (!$customer instanceof Customer) {
        $result->isError=true;
        $result->error = 'Fehlerhafter Parameter, Parameter ist kein
Customer';
        return $result;
    }

    $conn = getConnection();
    if ($conn->connect_error) {
        $result->isError=true;
        $result->error = $conn->connect_error;
        return $result;
    }

    $sql = 'UPDATE customer ' .
    ' SET ' .
    ' BEZ=' . quoted($customer->bez) .
    " WHERE PINDEX=$customer->pindex ";
    $resultSQL = $conn->query($sql);
    $conn->close();

    if ($resultSQL) {
        $result->isError=false;
    }
    else {
        $result->isError=true;
        $result->error = 'Fehlerhafte SQL-Anweisung<br />' . $conn->error .
'<br />' . $sql;
    }
    return $result;
} // updateCustomerDB

function deleteCustomerDB($pindex) {
    $result = new Result3();

    $conn = getConnection();
    if ($conn->connect_error) {
        $result->isError=true;

```

```

        $result->error = $conn->connect_error;
        return $result;
    }

    $sql = 'DELETE FROM customer ' . ' WHERE `PINDEX`=' . $pindex;
    //echo $sql . '<br />';
    if ($conn->query($sql)) {
        $result->isError=false;
    }
    else {
        $result->isError=true;
        $result->error = 'Fehlerhafte SQL-Anweisung<br />' . $conn->error .
'<br />' . $sql;
    }
    $conn->close();
    return $result;
} // deleteCustomerDB

function insertCustomerDB($customer) {
    $result = new Result3();
    if (!$customer instanceof Customer) {
        $result->isError=true;
        $result->error = 'Fehlerhafter Parameter, Parameter ist kein
Customer<br />Der Customer wurde NICHT eingetragen.';
        return $result;
    }

    $customer->pindex = getDBSequence();
    $sql = 'INSERT INTO customer(PINDEX, BEZ)' .
        ' VALUES( ' .
        ' ' . $customer->pindex . ', ' .
        ' ' . quoted($customer->bez)
        ' )';
    //echo $sql . '<br />';
    if ($conn->query($sql)) {
        $result->isError=false;
    }
    else {
        $result->isError=true;
        $result->error = 'Fehlerhafte SQL-Anweisung<br />' . $conn->error
. '<br />' . $sql;
    }

    return $result;
} // insertCustomerDB

?>

```

9 JavaScript

9.1 Allgemeine Eigenschaften

- Javascript ist eine leicht erlernbare und anwendbare Objekt-Sprache, die dazu dient, dynamische Elemente in HTML-Dateien einzuführen.
- Javascript ist kein direkter Bestandteil von HTML
- Es ist eine eigene Programmiersprache
- Es ist ähnlich wie Java (mit kleinen Unterschieden)
 - Unterschiede in
 - der Deklaration
 - Events
 - OOP
- Vom W3C wurde das Document Object Model (DOM) verabschiedet (Version 1,0 bis 2,0)
- DOM ist ein Interface, mit dem man auf die Elemente eines HTML-Dokuments zugreifen kann (bel. Sprache)
 - getElementById()
 - innerHTML

9.2 Spracheigenschaften

- Es ist eine Skriptsprache.
- Es ist objektorientiert
 -
- Integriert in HTML-Dateien
- Extern auch in Java-Dateien gespeichert
- Wird benötigt für HTML-Formulare
- Wird beim Client ausgeführt
- Keine Variablendeklaration, trotzdem sinnvoll (Strict-Modus)
 - `let myVariable;`
- Kein Compiler
- Strict-Modus
 - `"use strict"`
- Debugger verfügbar (Eclipse, Plug In)
- <https://addons.mozilla.org/de/firefox/addon/firebug/>
- Viele Sprach-Erweiterungen in HTML5
- Viele Frameworks
 - jQuery

9.3 Abschnitt definieren

Das Tag `<script>` definiert alle Textzeilen als Script

Beispiel:

```
<script type="text/javascript">
    "use strict"
    // Aktion
</script>
```


9.4 Datentypen

Deklariert werden Variablen mit der Anweisung „let“

Datentypen:

- int
- double
- String
- boolean
- Klassen

9.4.1 Methoden

```
let d = parseFloat( "123.456" );  
d = parseFloat( "s123.456" ); // Error und Rückgabewert ist NaN  
let k = parseInt( "123" );    // liefert 123  
n = parseInt("100px");       // liefert 100  
let sStr = k.toString();
```

9.5 Arrays

9.5.1 Anlegen eines Arrays

```
let feld = []; // leeres Array  
let feld = [1,2,3,4,5,6];
```

9.5.2 Anzahl eines Arrays

```
let anzahl = feld.length;
```

9.5.3 Eintrag hinzufügen

```
feld.push( 33 );  
feld.push( new MyClass() );
```

9.5.4 Eintrag löschen

```
feld.slice( i, 1 );
```

9.5.5 slice

Schneidet aus einem Array einen Bereich raus.

slice(start,end)

```
let feld = [1,2,3,4,5];  
res = feld.slice(1,3);  
alert(res); // 2,3
```

9.5.6 valueOf

Gibt alle Elemente des Array als String aus.

```
let feld = [1,2,3,4,5];  
res = feld.valueOf();  
alert(res); // 1,2,3,4,5
```

9.5.7 Mehrdimensionales Array

```
Let MAX = 100;  
let feld = new Array(MAX)  
for (let i=0; i<MAX; i++) {  
    feld[i] = new Array(MAX)  
}
```

9.6 Operatoren

Die Operatoren in PHP sind sehr ähnlich zu C, Perl oder Java

9.6.1 Mathematische Ausdrücke:

+	-	*	/	%	&		~		
+=	-=	*=	/=	%=	&=	=			
<	<=	==	===	=>	>	!=			
v++	++v	v--	--v						

9.6.2 Logische Verknüpfungen:

&&	And
	Or

9.7 Abfragen

9.7.1 If, then, else

```
if (Bedingung) {  
    Anweisung(en)  
}  
else {  
    Anweisung(en)
```

```
}

if (Bedingung)
    Anweisung1;
else
    Anweisung2;
```

Beispiel:

```
let zahl=5;
if(zahl>4) {
    alert ("Größer als 4");
}
else {
    alert("Kleiner gleich 4");
}
```

9.7.2 switch

```
switch(Variable){
    case Wert:
        Anweisung(en);
        break;

    ...
    case Wert:
        Anweisungen;
        break;
    default:
        Anweisungen
}
```

Beispiel:

```
let fkt = 1;
let x=2;
switch($fkt){
    case 1:
        y=2*x+1;
        break;
    case 2:
        y=2*x*x-2*x+1;
        break;
    case 1:
        y=-3*x*x-2*x+4;
        break;
    default:
        y=0;
}
```

9.8 Schleifen

9.8.1 For-Schleife

```
for( Initialisierung(en); Bedingung; Anweisung(en))
{
    Anweisung(en)
}
```

Beispiel:

```
for (let i=1; i<100; i++) {
    let k = i * i;
    alert("$i zum Quadrat ist " + k );
}
```

9.8.2 While-Schleife

Struktur:

```
while (Bedingung) {
    Anweisung(en)
}
```

Beispiel:

```
let i=1;
let k=1;
while(k<=100) {
    alert(i+" zum Quadrat ist " + k);
    $++;
    $ = i * i;
}
```

Beachten Sie auch die „break“ und „continue“-Anweisungen.

9.8.3 Do-While-Schleife

```
do {
    Anweisung(en)
}
while (Bedingung)
```

Beispiel:

```
let i=k=1;
do {
    k=i*i;
    alert(i+" zum Quadrat ist "+k);
    i++;
} while (k<100);
```

Unterschied zur while-Schleife:

Sie wird mindestens einmal durchlaufen.

Beachten Sie auch die „break“ und „continue“-Anweisungen.

9.8.4 Foreach-Schleife

In Javascript gibt es zwei verschiedene Foreach-Schleifen:

Die 1. foreach-Schleife dient nur als Kurzform für die Indizes für i.

```
let feld = [2,3,5,7,11,13];
let s=0
for (let item of feld ) {
    s+=item;
}
alert('Summe: ' +s);
```

Die 2. foreach-Schleife dient nur als Kurzform für die Indizes für i.

```
let feld = [2,3,5,7,11,13];
let s=0
for (let i in feld) {
    s+=feld[i];
}
alert('Summe: ' +s);
```

9.9 Funktionen

- Funktionen können rekursiv aufgerufen werden
- Die Argumente werden normalerweise "per value" übergeben

Beispiele:

```
// Rekursion ist erlaubt
function fak(n) {
    if (n>1)
        return n*fak(n-1);
    else
        return 1;
}
```

```
function addieren ($summand1, $summand2) {
    return $summand1 + $summand2;
}

let a=2;
let b=3;
let c=addieren($a,$b);
alert ( "Summe von " + a " und " + b + " ist " + c );
```

9.10 String-Methoden

charAt	Gibt das i-te Zeichen eines String aus. let s='abc'; let ch = s.charAt(1); // b
charCodeAt	Gibt den Code das i-te Zeichen eines String aus. let s='abc'; let ch = s.charCodeAt(1); // 98
concat	Fügt eine String am Ende hinzu. let s='abc'; s = s.concat('def');
length	Ermittelt die Länge des Strings let s='abc'; let n = s.length;
String.fromCharCode	Ausgabe eine Zeichen mittels des ASCII-Codes
indexOf	Sucht einen Teilstring in einem String. Results: Gefunden 0 bis n-1 Nicht gefunden: -1 let s='abc'; let index = s.indexOf('b');
lastIndexOf	Sucht eine Teilstring in einem String. Results: Gefunden 0 bis n-1 Nicht gefunden: -1 let s='abc22bc'; let index = s.lastIndexOf('bc'); alert(index); // Ergebnis 5
localeCompare	Vergleicht zwei String. Die Rückgabewerte sind die üblichen Werte <0,0,>0 let s='abc'; let res = s.localeCompare("abc"); Ergebnis: 0 let s='Abc'; let res = s.localeCompare("abc"); Ergebnis: 1
match	Sucht in einem String Teilstrings und gibt alle aus. let s="The rain in SPAIN stays mainly in the plain"; let res = s.match(/ain/g); alert(res); // Ergebnis rain, rain, rain
replace	Ersetzt das erste Vorkommen. let s="The rain in SPAIN stays mainly in the plain"; let res = s.replace("rain", "xyz"); alert(res); // The xyz in SPAIN stays mainly in the plain
replace (alle Elemente)	Ersetzt alle Vorkommen. let s="The rain in SPAIN stays mainly in the plain"; let res = s.replace(/ain/g, "xyz"); alert(res); // The rxyz in SPAIN stays mxyzly in the plxyz
search	Sucht einen Teilstring in einem String. Etwas langsamer als indexOf. Results: Gefunden 0 bis n-1 Nicht gefunden: -1 let s='abc'; let index = s.search('b'); // Ergebnis 1

slice	<p>Schneidet aus dem String einen Bereich raus. Wenn der Startindex kleiner 0 ist, starten die Berechnung vom Ende (0->n-1).</p> <pre>substr(start,end) let s='abcdef'; let res = s.slice(1,3); // start, ende alert(res); // bc</pre> <p>let s='abcdefghijk'; let res = s.slice(1,-3); alert(res); // bcdefgh</p> <p>let s='abcdefghijk'; let res = s.slice(-1,-3); alert(res); // bcdefgh</p>
Substr	<p>Schneidet aus dem String einen Bereich raus.</p> <pre>substr(start,end) chars ≤ end let s='abcdefghijk'; let res = s.substr(1,3); // start, ende alert(res); // bcd</pre>
substring	<p>Schneidet aus dem String einen Bereich raus.</p> <pre>substring(start,end) chars < end let s='abcdefghijk'; let res = s.substr(1,3); // start, ende alert(res); // bc</pre>
toLocaleLowerCase	Konvertieren in Kleinbuchstaben
toLocaleUpperCase	Konvertieren in Großbuchstaben
toLowerCase	Konvertieren in Kleinbuchstaben
toUpperCase	Konvertieren in Großbuchstaben
trim	Entfernt Trennzeichen am Anfang und am Ende des String
valueOf	The valueOf() method returns the primitive value of a String object.

9.11 Klassen

9.11.1 Klasse deklarieren

```
class Point {
  constructor(x,y) {
    this.x = x;
    this.y = y;
  }
} // Point
```

Unterschied zu Java:

- Um auf eine Klasse zugreifen zu können, muss sie zuvor definiert worden sein, sonst führt dies zu einem ReferenceError:

```
// anonyme Klasse
let point = class {
```

```

    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
};

// benannte Klasse
let point = class Point{
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
};

```

9.11.2 Vererbung

Vererbung mittels extends

```

class PointXY{
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }

    print() {
        console.log(this.x+", "+ this.y);
    }
} // class PoinXY

class PointZ extends PointXY{

    constructor(x, y,z) {
        super(x, y);
        this.z = z;
    }

    print() {
        super.print();
        console.log(this.z);
    }
} // class PointXYZ

let pxy = new PointXY(2,3);
let pxyz = new PointZ(12,13,14);
console.log( 22 );
pxy.print();
pxyz.print();

```


10 Ajax und JSON

Mit der Technik Ajax, Asynchron JavaScript ans XML, kann man dynamisch Teile einer HTML-Seite neu laden. Damit muss die komplette Seite nicht neu geladen werden, sondern nur einige Teile. Damit man nicht auf die Antwort warten muss, funktioniert der Ablauf asynchron.

10.1 Struktur von Ajax

Die Schritte bei Ajax:

- Es gibt einen Auslöser. Das ist meistens ein Schalter.
- Dieser Event ruft eine Javascript-Methode auf. Inm der unteren Abbildung „startAjax“ genannt.
- In „startAjax“ wird die Variable vom Typ XMLHttpRequest mit Parametern initialisiert und der Aufruf zum Server wird gestartet.
- Da Ajax asynchron abläuft, benötigt man eine Funktion, die nach der Rückkehr den Inhalt vom Server verarbeitet. In der unteren Abbildung heißt diese Funktion „receiveFromServer“. Diese Methode wird nach jeder Statusänderung aufgerufen. Man muss ja auch mitbekommen, ob der Server noch erreichbar ist. Insgesamt gibt es vier Werte für die Variable „readyState“:
- **readyState:**
 - 0: noch keine Initialisierung.
 - 1: Die Verbindung zum Server ist vorhanden.
 - 2: Die Daten vom Server sind empfangen.
 - 3: Die Daten vom Server wurden verarbeitet.
 - 4: Die Daten des Servers sind im „responseText“ eingetragen.
- Erst beim readyState gleich **vier** kann man die Daten verarbeiten.
 - Die Daten werden aus der Variablen „responseText“ des XMLHttpRequest-Variablen geholt und in geeigneter Form umgewandelt. Das heißt zum Beispiel eine einfache Ausgabe oder die Verarbeitung des JSon-Strings.
 - Am Schluss werden die Ergebnisse in einem div-Bereich mittels „innerHTML“ eingetragen.

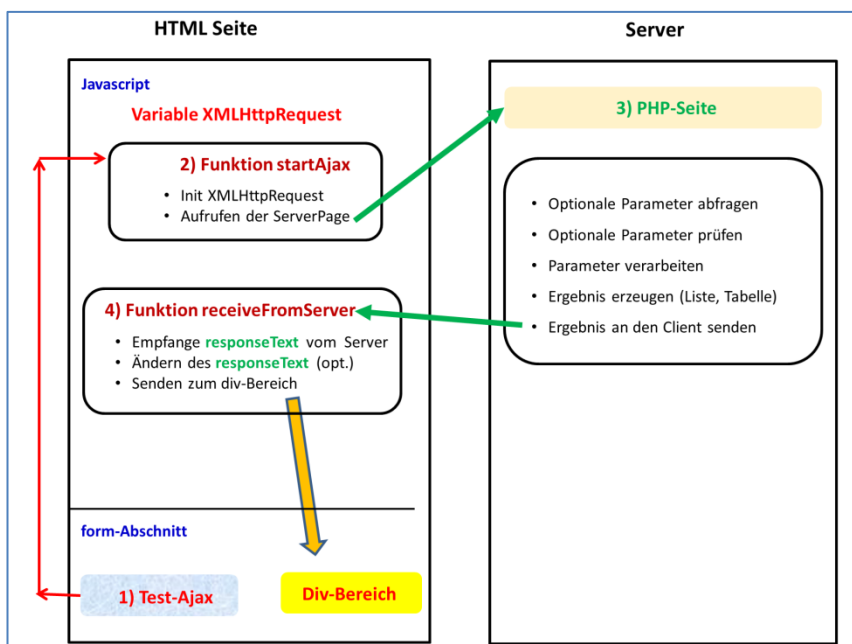


Abbildung 24 Struktur von Ajax

10.2 Ajax-Struktur als Quellcode

Der untere Quellcode zeigt ein einfaches Beispiel von Ajax, ohne JSon.

```
1 <script type="text/javascript">
2   // 
3   "use strict";
4
5   // globale Variable
6   let xmlhttp = new XMLHttpRequest(); // Ajax Manager
7
8
9   function startAjax( form ) {
10    "use strict";
11    // callBackMethode zuweisen
12    xmlhttp.onreadystatechange = receiveJson; // callBackMethode:
13    let param = 'ajax02b.php?zahl='+form.zahl.value;
14    alert(param);
15    xmlhttp.open('GET',param);
16    xmlhttp.send(); // aufbereiten der "Sendung", schicken
17  } // function startAjax
18
19
20   // callBackMethode
21   function receiveJson() {
22     "use strict";
23     // bei readyState ==4 bekommen wir Daten vom Server
24     if (xmlhttp.readyState==4) {
25       if (xmlhttp.status == 200) {
26         // in xmlhttp.responseText steht die Rückgabe vom Server
27         alert("Text:+"\n"+xmlhttp.responseText);
28         // Aktion
29       }
30       else {
31         // Fehlerhafter Aufruf: status bzw. statusText
32       }
33     }
34   }
35// ]]&gt;
36&lt;/script&gt;
37
38&lt;body&gt;
39&lt;form&gt;
40  Zahl
41  &lt;input type="number" name="zahl" min="0" max="100" value ="12" /&gt;
42  &lt;br /&gt;
43  &lt;br /&gt;
44  &lt;input type="button" value="Send" onclick="startAjax(this.form)"/&gt;
45&lt;/form&gt;
46&lt;/body&gt;</pre></div><div data-bbox="67 822 477 840" data-label="Section-Header"><h3>Abfragen beim Empfangen der Serverantwort:</h3></div><div data-bbox="98 841 374 923" data-label="List-Group"><ul><li>• status<ul><li>○ 200: "OK"</li><li>○ 403: "Forbidden"</li><li>○ 404: "Page not found"</li></ul></li><li>• statusText</li></ul></div><div data-bbox="895 936 940 955" data-label="Page-Footer"><p>114</p></div>
```

Erläuterung:

- 1: Beginn eines Script-Bereichs
- 2: CDATA verhindert, dass „if (i<3)“ als Tag interpretiert wird.
- 3: Im Strict-Modus müssen alle Variablen mit „let“ deklariert werden.
- 6: Die globale Variable „xmlhttp“ wird erzeugt. Sie wird in „startAjax“ und „receiveJson“ benötigt.
- 9: Funktion „startAjax“. Aufgerufen vom Button ganz unten
- 10: Im Strict-Modus müssen alle Variablen mit „let“ deklariert werden.
- 12: xmlhttp wird mit der Funktion receiveJson verbunden. Aufgerufen, wenn sich der readState ändert.
- 13: Der Parameter zum Server wird zusammengefasst.
- 14: Testausgabe des params.
- 15: Übergabe des Parameters.
- 16: Aktivieren des Aufrufs (readyState von 0 bis 4).
- 21: Funktion „receiveJson“. Wird aufgerufen von Objekt „xmlhttp“.
- 22: Im Strict-Modus müssen alle Variablen mit „let“ deklariert werden.
- 24: Sinnvoll ist die Abfrage auf „4“, da alle anderen Varianten unwichtig sind.
- 25: Abfrage des Rückgabewerts: 200 ok, sonst Fehlercode 4??
- 28: Aktion: Eintragen in einem Div-Bereich
- 31: Ausgabe des Fehlertextes im Fehlerfall.
- 42: „number“-Eingabeelement, Name=„zahl“.
- 45: Ein Button-Schalter der die funktion „startAjax“ aufruft.

Komplette Funktion:

```
function receiveJson() {
    "use strict";
    // bei n==4 bekommen wir Daten vom Server
    if (xmlhttp.readyState==4) {
        if (xmlhttp.status==200) {
            // in xmlhttp.responseText steht die Rückgabe vom Server
            let elementAjax = document.getElementById("ajax");
            elementAjax.innerHTML = xmlhttp.responseText;
        }
        else {
            let elementAjax = document.getElementById("ajax");
            elementAjax.innerHTML = 'Fehler:<br />Status: '+xmlhttp.status+
                '<br />StatusText: '+xmlhttp.statusText;
        }
    }
}
```

10.3 Status-Code bei Ajax

Link: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

100 Continue

The server has received the request headers and the client should proceed to send the request body (in the case of a request for which a body needs to be sent; for example, a [POST](#) request). Sending a large request body to a server after a request has been rejected for inappropriate headers would be inefficient. To have a server check the request's headers, a client must send `Expect: 100-continue` as a header in its initial request and receive a 100 Continue status code in response before sending the body. If the client receives an error code such as 403 (Forbidden) or 405 (Method Not Allowed) then it shouldn't send the request's body. The response 417 Expectation

`Failed` indicates that the request should be repeated without the `Expect` header as it indicates that the server doesn't support expectations (this is the case, for example, of HTTP/1.0 servers).

101 Switching Protocols

The requester has asked the server to switch protocols and the server has agreed to do so.

102 Processing ([WebDAV](#); [RFC 2518](#))

A WebDAV request may contain many sub-requests involving file operations, requiring a long time to complete the request. This code indicates that the server has received and is processing the request, but no response is available yet. This prevents the client from timing out and assuming the request was lost.

103 Early Hints ([RFC 8297](#))

Used to return some response headers before final HTTP message.

2xx Success

This class of status codes indicates the action requested by the client was received, understood, and accepted.

200 OK

Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action.

201 Created

The request has been fulfilled, resulting in the creation of a new resource.

202 Accepted

The request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, and may be disallowed when processing occurs.

203 Non-Authoritative Information (since HTTP/1.1)

The server is a transforming proxy (e.g. a [Web accelerator](#)) that received a 200 OK from its origin, but is returning a modified version of the origin's response.

204 No Content

The server successfully processed the request, and is not returning any content.

205 Reset Content

The server successfully processed the request, asks that the requester reset its document view, and is not returning any content.

206 Partial Content ([RFC 7233](#))

The server is delivering only part of the resource ([byte serving](#)) due to a range header sent by the client. The range header is used by HTTP clients to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.

207 Multi-Status ([WebDAV](#); [RFC 4918](#))

The message body that follows is by default an [XML](#) message and can contain a number of separate response codes, depending on how many sub-requests were made.

208 Already Reported ([WebDAV](#); [RFC 5842](#))

The members of a DAV binding have already been enumerated in a preceding part of the (multistatus) response, and are not being included again.

226 IM Used ([RFC 3229](#))

The server has fulfilled a request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

3xx Redirection

This class of status code indicates the client must take additional action to complete the request. Many of these status codes are used in [URL redirection](#).

A user agent may carry out the additional action with no user interaction only if the method used in the second request is GET or HEAD. A user agent may automatically redirect a request. A user agent should detect and intervene to prevent cyclical redirects.

300 Multiple Choices

Indicates multiple options for the resource from which the client may choose (via [agent-driven content negotiation](#)). For example, this code could be used to present multiple video format options, to list files with different [filename extensions](#), or to suggest [word-sense disambiguation](#).

[301 Moved Permanently](#)

This and all future requests should be directed to the given [URI](#).

[302 Found \(Previously "Moved temporarily"\)](#)

Tells the client to look at (browse to) another URL. 302 has been superseded by 303 and 307. This is an example of industry practice contradicting the standard. The HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"), but popular browsers implemented 302 with the functionality of a 303 See Other. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviours. However, some Web applications and frameworks use the 302 status code as if it were the 303.

[303 See Other](#) (since HTTP/1.1)

The response to the request can be found under another [URI](#) using the GET method. When received in response to a POST (or PUT/DELETE), the client should presume that the server has received the data and should issue a new GET request to the given URI.

304 Not Modified ([RFC 7232](#))

Indicates that the resource has not been modified since the version specified by the [request headers](#) If-Modified-Since or If-None-Match. In such case, there is no need to retransmit the resource since the client still has a previously-downloaded copy.

305 Use Proxy (since HTTP/1.1)

The requested resource is available only through a proxy, the address for which is provided in the response. For security reasons, many HTTP clients (such as [Mozilla Firefox](#) and [Internet Explorer](#)) do not obey this status code.

306 Switch Proxy

No longer used. Originally meant "Subsequent requests should use the specified proxy."

307 Temporary Redirect (since HTTP/1.1)

In this case, the request should be repeated with another URI; however, future requests should still use the original URI. In contrast to how 302 was historically implemented, the request method is not allowed to be changed when reissuing the original request. For example, a POST request should be repeated using another POST request.

308 Permanent Redirect ([RFC 7538](#))

The request and all future requests should be repeated using another URI. 307 and 308 parallel the behaviors of 302 and 301, but *do not allow the HTTP method to change*. So, for example, submitting a form to a permanently redirected resource may continue smoothly.

4xx Client errors

This class of status code is intended for situations in which the error seems to have been caused by the client. Except when responding to a HEAD request, the server *should* include an entity containing an

explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents *should* display any included entity to the user.

400 Bad Request

The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).

401 Unauthorized ([RFC 7235](#))

Similar to *403 Forbidden*, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See [Basic access authentication](#) and [Digest access authentication](#). 401 semantically means "unauthorised", the user does not have valid authentication credentials for the target resource.

Note: Some sites incorrectly issue HTTP 401 when an [IP address](#) is banned from the website (usually the website domain) and that specific address is refused permission to access a website.

402 Payment Required

Reserved for future use. The original intention was that this code might be used as part of some form of [digital cash](#) or [micropayment](#) scheme, as proposed, for example, by [GNU Taler](#), but that has not yet happened, and this code is not widely used. [Google Developers](#) API uses this status if a particular developer has exceeded the daily limit on requests. [Sipgate](#) uses this code if an account does not have sufficient funds to start a call. [Shopify](#) uses this code when the store has not paid their fees and is temporarily disabled. [Stripe](#) uses this code for failed payments where parameters were correct, for example blocked fraudulent payments.

[403 Forbidden](#)

The request contained valid data and was understood by the server, but the server is refusing action. This may be due to the user not having the necessary permissions for a resource or needing an account of some sort, or attempting a prohibited action (e.g. creating a duplicate record where only one is allowed). This code is also typically used if the request provided authentication by answering the WWW-Authenticate header field challenge, but the server did not accept that authentication. The request should not be repeated.

[404 Not Found](#)

The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

405 Method Not Allowed

A request method is not supported for the requested resource; for example, a GET request on a form that requires data to be presented via [POST](#), or a PUT request on a read-only resource.

406 Not Acceptable

The requested resource is capable of generating only content not acceptable according to the Accept headers sent in the request. See [Content negotiation](#).

407 Proxy Authentication Required ([RFC 7235](#))

The client must first authenticate itself with the [proxy](#).

408 Request Timeout

The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."

409 Conflict

Indicates that the request could not be processed because of conflict in the current state of the resource, such as an [edit conflict](#) between multiple simultaneous updates.

410 Gone

Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource in the future. Clients such as search engines should remove the resource from their indices. Most use cases do not require clients and search engines to purge the resource, and a "404 Not Found" may be used instead.

411 Length Required

The request did not specify the length of its content, which is required by the requested resource.

412 Precondition Failed ([RFC 7232](#))

The server does not meet one of the preconditions that the requester put on the request header fields.

413 Payload Too Large ([RFC 7231](#))

The request is larger than the server is willing or able to process. Previously called "Request Entity Too Large".

414 URI Too Long ([RFC 7231](#))

The [URI](#) provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request. Called "Request-URI Too Long" previously.

415 Unsupported Media Type ([RFC 7231](#))

The request entity has a [media type](#) which the server or resource does not support. For example, the client uploads an image as [image/svg+xml](#), but the server requires that images use a different format.

416 Range Not Satisfiable ([RFC 7233](#))

The client has asked for a portion of the file ([byte serving](#)), but the server cannot supply that portion. For example, if the client asked for a part of the file that lies beyond the end of the file. Called "Requested Range Not Satisfiable" previously.^[51]

417 Expectation Failed

The server cannot meet the requirements of the Expect request-header field.

[418 I'm a teapot](#) ([RFC 2324](#), [RFC 7168](#))

This code was defined in 1998 as one of the traditional [IETF April Fools' jokes](#), in [RFC 2324](#), [Hyper Text Coffee Pot Control Protocol](#), and is not expected to be implemented by actual HTTP servers. The RFC specifies this code should be returned by teapots requested to brew coffee. This HTTP status is used as an [Easter egg](#) in some websites, such as [Google.com's I'm a teapot](#) easter egg.

421 Misdirected Request ([RFC 7540](#))

The request was directed at a server that is not able to produce a response (for example because of connection reuse).

422 Unprocessable Entity (WebDAV; [RFC 4918](#))

The request was well-formed but was unable to be followed due to semantic errors.

423 Locked (WebDAV; [RFC 4918](#))

The resource that is being accessed is locked.

424 Failed Dependency (WebDAV; [RFC 4918](#))

The request failed because it depended on another request and that request failed (e.g., a PROPPATCH).

425 Too Early ([RFC 8470](#))

Indicates that the server is unwilling to risk processing a request that might be replayed.

426 Upgrade Required

The client should switch to a different protocol such as [TLS/1.0](#), given in the [Upgrade header](#) field.

428 Precondition Required ([RFC 6585](#))

The origin server requires the request to be conditional. Intended to prevent the 'lost update' problem, where a client GETs a resource's state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict.^[59]

429 Too Many Requests ([RFC 6585](#))

The user has sent too many requests in a given amount of time. Intended for use with [rate-limiting](#) schemes.^[59]

431 Request Header Fields Too Large ([RFC 6585](#))

The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.

[451 Unavailable For Legal Reasons](#) ([RFC 7725](#))

A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource. The code 451 was chosen as a reference to the novel [Fahrenheit 451](#) (see the Acknowledgements in the RFC).

5xx Server errors

The [server](#) failed to fulfill a request.

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Except when responding to a HEAD request, the server *should* include an entity containing an explanation of the error situation, and indicate whether it is a temporary or permanent condition. Likewise, user agents *should* display any included entity to the user. These response codes are applicable to any request method.

500 Internal Server Error

A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

501 Not Implemented

The server either does not recognize the request method, or it lacks the ability to fulfil the request. Usually this implies future availability (e.g., a new feature of a web-service API).

502 Bad Gateway

The server was acting as a [gateway](#) or proxy and received an invalid response from the upstream server.

503 Service Unavailable

The server cannot handle the request (because it is overloaded or down for maintenance). Generally, this is a temporary state.

504 Gateway Timeout

The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

505 HTTP Version Not Supported

The server does not support the HTTP protocol version used in the request.

506 Variant Also Negotiates ([RFC 2295](#))

Transparent [content negotiation](#) for the request results in a [circular reference](#).

507 Insufficient Storage (WebDAV; [RFC 4918](#))

The server is unable to store the representation needed to complete the request.

508 Loop Detected (WebDAV; [RFC 5842](#))

The server detected an infinite loop while processing the request (sent instead of [208 Already Reported](#)).

510 Not Extended ([RFC 2774](#))

Further extensions to the request are required for the server to fulfil it.

511 Network Authentication Required ([RFC 6585](#))

The client needs to authenticate to gain network access. Intended for use by intercepting proxies used to control access to the network (e.g., "[captive portals](#)" used to require agreement to Terms of Service before granting full Internet access via a [Wi-Fi hotspot](#)).

Unofficial codes

The following codes are not specified by any standard.

103 Checkpoint

Used in the resumable requests proposal to resume aborted PUT or POST requests.

218 This is fine ([Apache Web Server](#))

Used as a catch-all error condition for allowing response bodies to flow through Apache when ProxyErrorOverride is enabled. When ProxyErrorOverride is enabled in Apache, response bodies that contain a status code of 4xx or 5xx are automatically discarded by Apache in favor of a generic response or a custom response specified by the ErrorDocument directive.

419 Page Expired ([Laravel Framework](#))

Used by the Laravel Framework when a CSRF Token is missing or expired.

420 Method Failure ([Spring Framework](#))

A deprecated response used by the Spring Framework when a method has failed.

420 Enhance Your Calm ([Twitter](#))

Returned by version 1 of the Twitter Search and Trends API when the client is being rate limited; versions 1.1 and later use the [429 Too Many Requests](#) response code instead. The phrase "Enhance your calm" comes from the [1993 movie Demolition Man](#), and its association with this number is likely a reference to [cannabis](#).^{[\[citation needed\]](#)}

430 Request Header Fields Too Large ([Shopify](#))

Used by [Shopify](#), instead of the [429 Too Many Requests](#) response code, when too many URLs are requested within a certain time frame.

450 Blocked by Windows Parental Controls (Microsoft)

The Microsoft extension code indicated when Windows Parental Controls are turned on and are blocking access to the requested webpage.

498 Invalid Token (Esri)

Returned by [ArcGIS for Server](#). Code 498 indicates an expired or otherwise invalid token.

499 Token Required (Esri)

Returned by [ArcGIS for Server](#). Code 499 indicates that a token is required but was not submitted.

509 Bandwidth Limit Exceeded ([Apache Web Server/cPanel](#))

The server has exceeded the bandwidth specified by the server administrator; this is often used by shared hosting providers to limit the bandwidth of customers.

526 Invalid SSL Certificate

Used by [Cloudflare](#) and [Cloud Foundry](#)'s gorouter to indicate failure to validate the SSL/TLS certificate that the origin server presented.

529 Site is overloaded

Used by [Qualys](#) in the SSL Labs server testing API to signal that the site can't process the request.

530 Site is frozen

Used by the [Panttheon](#) web platform to indicate a site that has been frozen due to inactivity.

598 (Informal convention) Network read timeout error

Used by some HTTP proxies to signal a network read timeout behind the proxy to a client in front of the proxy.

Internet Information Services

Microsoft's [Internet Information Services](#) (IIS) web server expands the 4xx error space to signal errors with the client's request.

440 Login Time-out

The client's session has expired and must log in again.

449 Retry With

The server cannot honour the request because the user has not provided the required information.

451 Redirect

Used in [Exchange ActiveSync](#) when either a more efficient server is available or the server cannot access the users' mailbox. The client is expected to re-run the HTTP AutoDiscover operation to find a more appropriate server.

IIS sometimes uses additional decimal sub-codes for more specific information, however these sub-codes only appear in the response payload and in documentation, not in the place of an actual HTTP status code.

10.4 JSON

Der Datentransfer vom Server zum Client/Browser geschieht Definitionsgemäß mit XML. Da XML aber jeweils zwei Tags hat, wurde JSON, Javascript-Object-Notation. Diese Struktur hat folgende Vorteile:

- Wird vom Server unterstützt.
- Wird in JavaScript unterstützt.
- Es hat nur ein Tag.
- Es ist Maschinenlesbar.
- Es ist für Menschen lesbar

10.4.1 Aufbau von JSON

Anfangstag:	{
Endtag:	}
Arrayanfang:	[
Arrayende:]
Tag:	Name wird gequoted
Datentypen:	Zahlen, boolean
Datentypen:	String (gequoted)
Objekte	Attribute durch Komma getrennt

10.4.2 JSON-Beispiele:

`{"zahl":123}` Übergabe eines numerischen Wertes

`{"zahl":123.456}` Übergabe eines numerischen Wertes

`{"zahl1":12, "zahl2":34}` Übergabe von zwei numerischen Werten

Übergabe von zwei numerischen Werten, der Summe und eventuelle Fehlertexte

```
{
"result":true,    "errorstring":"-",
"zahl1":12,      "zahl2":34,
"summe":46
}
```

`{"nachname":"Meier","gehalt":2234.45}` Übergabe eines Mitarbeiters mit dem jeweiligen Gehalt

`[{"nachname":"Meier","gehalt":2234.45}, {"nachname":"Schulze","gehalt":2734.45}]`

Übergabe zweier Mitarbeiter mit dem jeweiligen Gehalt.

Diese Variante entspricht einem Array mit der Klasse Mitarbeiter in Java:

```
class Mitarbeiter {
    public String nachname="";
    public double gehalt = 0.0;
}
```

```
{
  "result":true,
  "errorstring":"-",
  "mitarbeiter" :[ { "nachname","Meier","gehalt":2234}, { "nachname","Schulze","gehalt":2734} ]
}
```

```
class Mitarbeiter {
    public String nachname="";
    public double gehalt = 0.0;
}

class Result {
    public boolean result=true;
    public String errorstring="";
    public Mitarbeiter[] mitarbeiter;
}
```

Um auch Fehlertext zum Browser zu bringen, kann man die obige Variante definieren. Im Prinzip hat man zwei Teile.

1. Rückgabe eventueller Fehler
2. Übergabe der Mitarbeiter mit dem jeweiligen Nachnamen und Gehalt.

10.4.3 PHP und JSon

In PHP verwendet man die Methode „json_decode“ um ein Array oder ein Objekt automatisch in einen JSon-String umzuwandeln.

Wichtig:

- Das funktioniert nur, wenn man die Attribute auf public setzt.
- Ansonsten benötigt man noch zwei Methoden pro Klasse.
 - Methode „processArray“
 - Methode „toArray“
 - Beides erhältlich in der DIE unter PHP/JSon

10.4.4 Javascript und JSon

In Javascript verwendet man die Methode „JSON.parse“ um einen JSon-Text in ein Array oder in ein umzuwandeln.

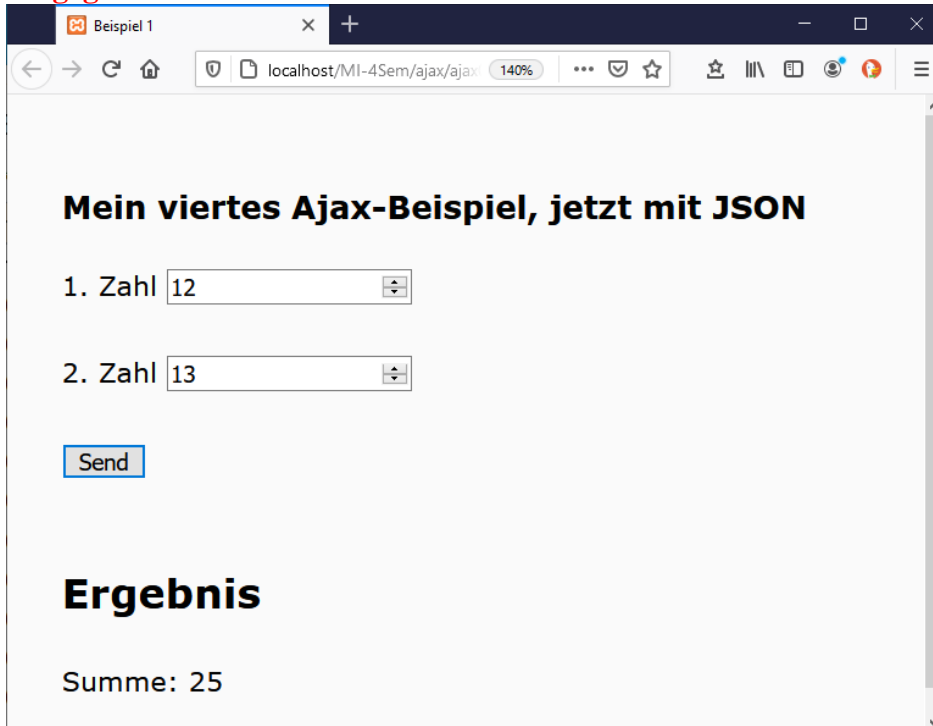
Wichtig:

- **Der Aufbau und die Namen der Attribute entsprechen genau den Klassen in PHJP.**

10.5 Beispiele

10.5.1 Summe zweier Zahlen

Das aktuelle Beispiel hat als Eingabe zwei Number-Zahlen, die durch den Server addiert werden sollen. Das Ergebnis soll als JSON-Text zurückgegeben werden. **Dabei soll auch eine optionale Fehlermeldung mitgegeben werde.**



Beispiel 1

localhost/MI-4Sem/ajax/ajax 140%

Mein viertes Ajax-Beispiel, jetzt mit JSON

1. Zahl

2. Zahl

Ergebnis

Summe: 25

Abbildung 25 Erstes JAX-JSON-Beispiel

10.5.1.1 Aufbau der ersten Seite

Zumbesseren Verständnis hier die PHP-Klasse:

```
class Result {
    public $isError=false;
    public $error='';
    public $ergebnis=0;
    public function __construct($ergebnis) {
        $this->ergebnis = $ergebnis;
    }
}
```

Man braucht:

- Erstellen einer HTML-Seite: Bsp01a.xhtml
 - body-Abschnitt
 - einen form-Abschnitt
 - Zwei Number-Elemente
 - Setzen mit 12 und 13 (also unterschiedlich)
 - Einen button-Schalter mit der Methode „startAjax“
 - Eintragen eines div-Abschnittes
 - <div id="ajax"></div>
 - Erstellen eines Javascript-Abschnittes mit Ajax (am besten den Menüpunkt benutzen)

- Eintragen der Parameter
- let param = 'Bsp01b.php?zahl1='+form.zahl1.value+'&zahl2='+form.zahl2.value;
- Funktion „receiveJson“
 - Ausgabe des Rückgabewertes
 - alert(xmlhttp1.responseText);
 - Abfragen der Variable „isError“
 - Ausgabe des Fehlertextes
 - Umwandeln des JSON-String in ein Objekt
 - let result = JSON.parse(xmlhttp1.responseText);
 - Holen des div-Abschnittes mit getElementbyID
 - let elementAjax = document.getElementById("ajax");
 - Setzen des Ergebnisses
 - let s = '<h2>Ergebnis</h2>Summe: '+ result.ergebnis;
 - elementAjax.innerHTML = s;
- Erstellen einer PHP-Seite: Bsp01b.php
 - Eintragen der Klasse „Result“ mit den Attributen:
 - isError
 - errortext
 - ergebnis
 - Eintragen der Grundvariablen
 - \$ok = true;
 - \$zahl1 = 0;
 - \$zahl2 = 0;
 - \$error = "";
 - Abfrage der beiden Parameter zahl1 und zahl2
 - isset / is_numeric
 - setzen der Fehlertexte
 - Fehler vorhanden
 - Erzeugen einer Instant der Klasse „Result“
 - Eintragen des Fehlertextes
 - Setzen des Attributs isError auf true
 - Ausgabe mit „json_encode“
 - Keine Fehler
 - Erzeugen einer Instant der Klasse „Result“
 - Ausgabe mit „json_encode“

10.5.1.2 Quellcode der ersten Seite

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title> Beispiel 1 </title>
    <meta name="author" content="Administrator"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" href="ajax04a.css"
title="style1"/>
  </head>

  <script type="text/javascript">
    // <![CDATA[
```

```

"use strict";

let xmlhttp1 = new XMLHttpRequest();
xmlhttp1.onreadystatechange = receiveJson; // Funktionspointer

function startAjax(form) {
    "use strict";
    let param = 'ajax04b.php?zahl1='+form.zahl1.value+'&zahl2=' +
        form.zahl2.value;
    alert(param);
    xmlhttp1.open("GET", param);
    xmlhttp1.send();
} // startAjax

function receiveJson() {
    "use strict";
    if (xmlhttp1.readyState == 4) {

        alert(xmlhttp1.responseText);
        let elementAjax = document.getElementById("ajax");
        let result = JSON.parse(xmlhttp1.responseText);
        if (result.isError) {
            let s='<h2>Fehler</h2>'+result.error;
            elementAjax.innerHTML = s;
        }
        else {
            let s='<h2>Ergebnis</h2>';
            s+='Summe: ' + number.ergebnis;
            elementAjax.innerHTML = s;
        }
    } // if
} //receiveJson

// ]]>
</script>

```

```

<body>
    <h3>Mein viertes Ajax-Beispiel, jetzt mit JSON</h3>
    <form>
        1. Zahl <input type="number" name="zahl1" min="0" max="100" value
="12" />
        <br />
        <br />
        2. Zahl <input type="number" name="zahl2" min="0" max="100" value
="13" />
        <br />
        <br />
        <input type="button" value="Send" onclick="startAjax(this.form)"/>
    </form>
    <br />
    <div id="ajax"></div>
</body>
</html>

```

10.5.1.3 Quellcode der zweiten Seite

```
<?php

class Result {
    public $isError=false;
    public $errortext='';
    public $ergebnis=0;
    public function __construct($ergebnis) {
        $this->ergebnis = $ergebnis;
    }
}

$ok = true;
$zahl1 = 0;
$zahl2 = 0;
$error = '';

if ( isset($_GET['zahl1']) ){    // Ist gesetzt?
    $str_zahl1 = $_GET['zahl1'];
    if (is_numeric($str_zahl1)) {
        $zahl1 = intval ( $str_zahl1, 10 );
    }
    else {
        $ok = false;
        $error.="Error: the variable 'zahl' is not any number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl1' are not passed<br />";
}

if ( isset($_GET['zahl2']) ){    // Ist gesetzt?
    $str_zahl2 = $_GET['zahl2'];
    if (is_numeric($str_zahl2)) {
        $zahl2 = intval ( $str_zahl2, 10 );
    }
    else {
        $ok = false;
        $error.="Error: the variable 'zahl2' is not any number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl2' are not passed<br />";
}

if ($ok) {
    $ergebnis = $zahl1+$zahl2;
    $result = new Result($ergebnis);
    $result->isError=false;
    echo json_encode($result);
}
```

```

else {
    $result = new Result(0);
    $result->isError=true;
    $result->errortext=$error;
    echo json_encode($result);
}
?>

```

10.5.2 Taschenrechner mit zwei Zahlen

Das zweite Beispiel hat als Eingabe zwei Number-Zahlen und vier RadioButtons, die die mathematische Operation bestimmen. Das Ergebnis soll als JSon-Text zurückgegeben werden. **Dabei soll auch eine optionale Fehlermeldung mitgegeben werde.**

Taschenrechner mit zwei Zahlen

Eingabe der Zahlen

1. Number

2. Number

math. Operation

☒ Addition

☐ Subtraktion

☐ Multiplikation

☐ Division

Ergebnis

- 1. Zahl: 12
- 2. Zahl: 13
- Ergebnis: 25

Abbildung 26 Zweites AJAX-JSon-Beispiel

10.5.2.1 Aufbau der ersten Seite

Zumbesseren Verständnis hier die PHP-Klasse:

```

class Result {
    public $error='';
    public $iserror=false;
    public $zahl1=0;
    public $zahl2=0;
    public $ergebnis=0;
    public function __construct($zahl1, $zahl2) {
        $this->zahl1 = $zahl1;
        $this->zahl2 = $zahl2;
    }
}

```


Man braucht:

- Erstellen einer HTML-Seite: Bsp01a.xhtml
 - body-Abschnitt
 - einen form-Abschnitt
 - Zwei Number-Elemente in einem Fieldset einfügen.
 - Setzen mit 12 und 13 (also unterschiedlich).
 - Vier RadioButtons in einem Fieldset setzen.
 - Einen button-Schalter mit der Methode „startAjax“
 - Eintragen eines div-Abschnittes
 - `<div id="ajax"></div>`
 - Erstellen eines Javascript-Abschnittes mit Ajax (am besten den Menüpunkt benutzen)
 - Eintragen der Parameter
 - `let param = 'Bsp02b.php?zahl1=' + form.zahl1.value + '&zahl2=' + form.zahl2.value + '&mathop='+form.rbmathop.value;`
 - Funktion „receiveJson“
 - Ausgabe des Rückgabewertes
 - `alert(xmlhttp1.responseText);`
 - Abfragen der Variable „isError“
 - Ausgabe des Fehlertextes
 - Umwandeln des JSon-String in ein Objekt
 - `let result = JSON.parse(xmlhttp1.responseText);`
 - Holen des div-Abschnittes mit `getElementbyID`
 - `let elementAjax = document.getElementById("ajax");`
 - Setzen des Ausgabetextes
 - `let s='<h2>Ergebnis</h2>';`
 - `s+='';`
 - `s+=''+ '1. Zahl: ' + result.zahl1 + '';`
 - `s+=''+ '2. Zahl: ' + result.zahl2 + '';`
 - `s+=''+ 'Ergebnis: ' + result.ergebnis + '';`
 - `s+='';`
 - `elementAjax.innerHTML = s;`
 - Erstellen einer PHP-Seite: Bsp02b.php
 - Eintragen der Klasse „Result“ mit den Attributen:
 - `isError`
 - `errortext`
 - `zahl1`
 - `zahl2`
 - `ergebnis`
 - Eintragen der Grundvariablen
 - `$ok = true;`
 - `$zahl1 = 0;`
 - `$zahl2 = 0;`
 - `$mathop = "";`
 - `$error = "";`
 - Abfrage der beiden Parameter `zahl1` und `zahl2` und des `mathops`
 - `isset / is_numeric`
 - setzen der Fehlertexte
 - Fehler vorhanden
 - Erzeugen einer Instant der Klasse „Result“
 - Eintragen des Fehlertextes
 - Setzen des Attributs `isError` auf `true`
 - Ausgabe mit „`json_encode`“
 - Keine Fehler

- Erzeugen einer Instanz der Klasse „Result“
- Ausgabe mit „json_encode“

10.5.2.2 Quellcode der ersten Seite

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title> Beispiel 1 </title>
    <meta name="author" content="Administrator"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" href="ajax07a.css"
title="style1"/>
  </head>

  <script type="text/javascript">
    // 
      "use strict";

      let xmlhttp1 = new XMLHttpRequest();
      xmlhttp1.onreadystatechange = receiveJson; // Funktionspointer

      function startAjax(form) {
        "use strict";
        let param = 'ajax07b.php?zahl1=' + form.zahl1.value + '&zahl2=' +
form.zahl2.value+'&amp;mathop='+form.rbmathop.value;
        alert(param);
        xmlhttp1.open("GET", param);
        xmlhttp1.send();
      } // startAjax

      function receiveJson() {
        "use strict";
        // alert(xmlhttp1.readyState);
        if (xmlhttp1.readyState == 4) {
          let elementAjax = document.getElementById("ajax");
          alert(xmlhttp1.responseText);
          let result = JSON.parse(xmlhttp1.responseText);
          if (result.iserror) {
            let s='&lt;h2&gt;Fehler&lt;/h2&gt;';
            s+=result.error;
            elementAjax.innerHTML = s;
          }
          else {
            let s='&lt;h2&gt;Ergebnis&lt;/h2&gt;';
            s+='&lt;ul&gt;';
            s+='&lt;li&gt;'+ '1. Zahl: ' + result.zahl1 + '&lt;/li&gt;';
            s+='&lt;li&gt;'+ '2. Zahl: ' + result.zahl2 + '&lt;/li&gt;';
            s+='&lt;li&gt;'+ 'Ergebnis: ' + result.ergebnis + '&lt;/li&gt;';
            s+='&lt;/ul&gt;';
            elementAjax.innerHTML = s;
          }
        }
      } // if</pre>
</div>
<div data-bbox="896 938 939 955" data-label="Page-Footer">
<p>130</p>
</div>
```

```

    } //receiveJson

    // ]]>
</script>

<body>
  <h3>Taschenrechner mit zwei Zahlen</h3>
  <form>
    <fieldset>
      <legend>Eingabe der Zahlen</legend>
      1. Number <input type="number" name="zahl1" min="0" max="100"
value ="12" />
      <br />
      2. Number <input type="number" name="zahl2" min="0" max="100"
value ="13" />
    </fieldset>
    <br />
    <fieldset>
      <legend>math. Operation</legend>
      <input type="radio" name="rbmathop" value="add" checked="checked" />
Addition<br />
      <input type="radio" name="rbmathop" value="sub" /> Subtraktion
      <br />
      <input type="radio" name="rbmathop" value="mult" /> Multiplikation
      <br />
      <input type="radio" name="rbmathop" value="div" /> Division
      <br />
    </fieldset>
    <br />
    <br />
    <input type="button" value="Send" onclick="startAjax(this.form)"/>

  </form>
  <div id="ajax"/>

</body>
</html>

```

10.5.2.3 Quellcode der zweiten Seite

```

<?php

class Result {
    public $error='';
    public $iserror=false;
    public $zahl1=0;
    public $zahl2=0;
    public $ergebnis=0;
    public function __construct($zahl1, $zahl2) {
        $this->zahl1 = $zahl1;
        $this->zahl2 = $zahl2;
    }
}

```

```

$ok = true;
$zahl1 = 0;
$zahl2 = 0;
$mathop = '';
$error = '';

if ( isset($_GET['zahl1']) ){ // Ist gesetzt?
    $str_zahl1 = $_GET['zahl1'];
    if (is_numeric($str_zahl1)) {
        $zahl1 = intval ( $str_zahl1, 10 );
    }
    else {
        $ok = false;
        $error.="Error in number conversion: the variable 'zahl1' is not any
number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl1' are not passed<br />";
}

if ( isset($_GET['zahl2']) ){ // Ist gesetzt?
    $str_zahl2 = $_GET['zahl2'];
    if (is_numeric($str_zahl2)) {
        $zahl2 = intval ( $str_zahl2, 10 );
    }
    else {
        $ok = false;
        $error.="Error in number conversion: the variable 'zahl2' is not any
number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl2' are not passed<br />";
}

if ( isset($_GET['mathop']) ){ // Ist gesetzt?
    $dummy = $_GET['mathop'];
    $feld = array("add", "sub", "mult", "div");
    $gefunden=false;
    foreach($feld as $op) {
        if ($dummy==$op) {
            $mathop = $op;
            $gefunden=true;
            break;
        }
    }
    if (!$gefunden) {
        $ok = false;
        $error.="Error: The param 'mathop' are not correct: $dummy<br />";
    }
}
else {

```

```

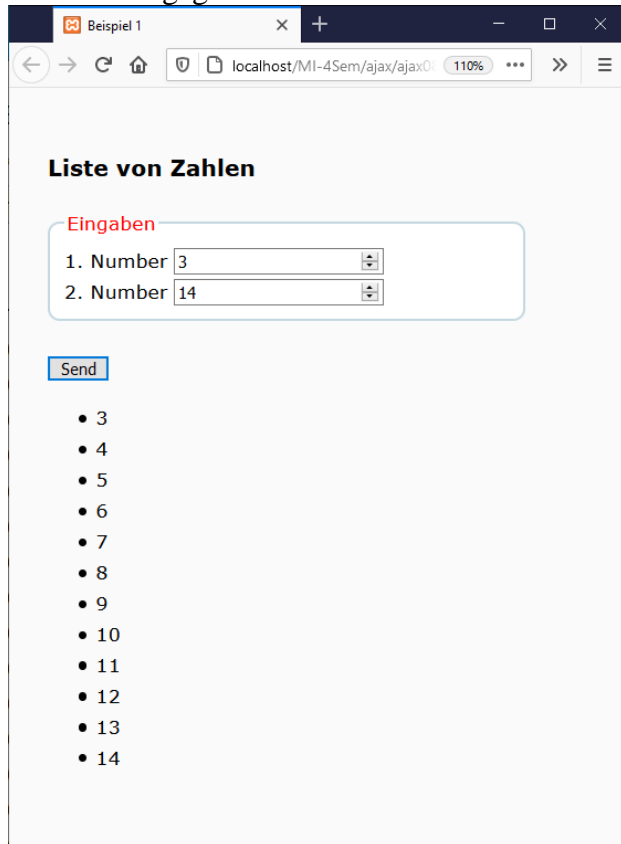
$ok = false;
$error.="Error: The param 'mathop' are not passed<br />";
}

if ($ok) {
    $result = new Result($zahl1,$zahl2);
    switch ($mathop) {
        case 'add':
            $result->ergebnis=$zahl1+$zahl2;
            break;
        case 'sub':
            $result->ergebnis=$zahl1-$zahl2;
            break;
        case 'mult':
            $result->ergebnis=$zahl1*$zahl2;
            break;
        case 'div':
            if ($zahl2==0) {
                $result->error = "Fehlerhafte Div, der Nenner ist Null";
                $result->iserror=true;
                break;
            }
            else {
                $result->ergebnis=$zahl1/$zahl2;
            }
            break;
        default:
            $result->error = "Fehlerhafte switch/case: $mathop";
            $result->iserror=true;
            break;
    } // switch
    echo json_encode($result);
}
else {
    $result = new Result(0,0);
    $result->error = $error;
    $result->iserror=true;
    echo json_encode($result);
}
?>

```

10.5.3 Liste mit Zahlen

Das dritte Beispiel hat als Eingabe zwei Number-Zahlen. Der Server soll daraus eine Liste von Zahlen generieren. Im Javascript soll daraus eine ungeordnetet Liste erstellt werden. Das Ergebnis soll als JSon-Text zurückgegeben werden. **Dabei soll auch eine optionale Fehlermeldung mitgegeben werde.**



The screenshot shows a web browser window with the title 'Beispiel 1'. The address bar shows 'localhost/MI-4Sem/ajax/ajax0'. The page content is titled 'Liste von Zahlen'. Below the title is a form with the label 'Eingaben'. Inside the form are two input fields: '1. Number' with the value '3' and '2. Number' with the value '14'. Below the form is a 'Send' button. The output of the form is a bulleted list of numbers from 3 to 14.

Abbildung 27 Drittes AJAX-JSon-Beispiel

10.5.3.1 Aufbau der ersten Seite

Zumbesseren Verständnis hier die PHP-Klassen:

```
class Number {
    public $zahl=0;
    public function __construct($zahl) {
        $this->zahl = $zahl;
    }
} // Number

class Result {
    public $error='';
    public $iserror=false;
    public $liste= array();
} // Result
```

Man braucht:

- Erstellen einer HTML-Seite: Bsp03a.xhtml
 - body-Abschnitt
 - einen form-Abschnitt
 - Zwei Number-Elemente in einem Fieldset einfügen.
 - Setzen mit 4 und 13 (also unterschiedlich).
 - Einen button-Schalter mit der Methode „startAjax“

- Eintragen eines div-Abschnittes
 - `<div id="ajax"></div>`
 - Erstellen eines Javascript-Abschnittes mit Ajax (am besten den Menüpunkt benutzen)
 - Eintragen der Parameter
 - `let param = 'Bsp03b.php?zahl1=' + form.zahl1.value + '&zahl2=' + form.zahl2.value;`
 - Funktion „receiveJson“
 - Ausgabe des Rückgabewertes
 - `alert(xmlhttp1.responseText);`
 - Abfragen der Variable „isError“
 - Ausgabe des Fehlertextes
 - Umwandeln des JSon-String in ein Objekt
 - `let result = JSON.parse(xmlhttp1.responseText);`
 - Holen des div-Abschnittes mit getElementbyID
 - `let elementAjax = document.getElementById("ajax");`
 - Setzen des Ausgabetextes
 - `let s='';`
 - `for (let i in result.liste) {`
 - `s+='' + result.liste[i].zahl + '\n' ;`
 - `}`
 - `s+='';`
 - `elementAjax.innerHTML = s;`
 - Erstellen einer PHP-Seite: Bsp03b.php
 - Eintragen der Klasse „Result“ und Number mit den Attributen:
 - isError
 - errortext
 - Liste
 - zahl
 - Eintragen der Grundvariablen
 - `$ok = true;`
 - `$zahl1 = 0;`
 - `$zahl2 = 0;`
 - `$error = "";`
 - Abfrage der beiden Parameter zahl1 und zahl2 und des mathops
 - `isset / is_numeric`
 - setzen der Fehlertexte
 - Fehler vorhanden
 - Erzeugen einer Instant der Klasse „Result“
 - Eintragen des Fehlertextes
 - Setzen des Attributs isError auf true
 - Ausgabe mit „json_encode“
 - Keine Fehler
 - Erzeugen einer Instant der Klasse „Result“
 - Erstellen einer Liste
 - In einer Schleife die einzelnen Zahlen erzeugen und in die Liste eintragen.
 - Ausgabe mit „json_encode“

10.5.3.2 Quellcode der ersten Seite

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title> Beispiel 1 </title>
    <meta name="author" content="Administrator"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" href="ajax08a.css"
title="style1"/>
  </head>

  <script type="text/javascript">
    // <![CDATA[
    "use strict";

    let xmlhttp1 = new XMLHttpRequest();
    xmlhttp1.onreadystatechange = receiveJson; // Funktionspointer

    function startAjax(form) {
      "use strict";
      let param = 'ajax08b.php?zahl1=' + form.zahl1.value + '&zahl2=' +
form.zahl2.value;
      //alert(param);
      xmlhttp1.open("GET", param);
      xmlhttp1.send();
    } // startAjax

    function receiveJson() {
      "use strict";
      // alert(xmlhttp1.readyState);
      if (xmlhttp1.readyState == 4) {
        let elementAjax = document.getElementById("ajax");
        alert(xmlhttp1.responseText);
        let result = JSON.parse(xmlhttp1.responseText);
        if (result.iserror) {
          let s='<h2>Fehler</h2>';
          s+=result.error;
          elementAjax.innerHTML = s;
        }
        else {
          let s='<ul>';
          for (let i in result.liste) {
            s+='<li>' + result.liste[i].zahl + '</li>\n' ;
          }
          s+='</ul>';
          elementAjax.innerHTML = s;
        }
      } // if
    } //receiveJson

    // ]]>
  </script>
```



```

<body>
  <h3>Liste von Zahlen</h3>
  <form>
    <fieldset>
      <legend>Eingaben</legend>
      1. Number <input type="number" name="zahl1" min="0" max="100"
value ="3" />
      <br />
      2. Number <input type="number" name="zahl2" min="0" max="100"
value ="14" />
    </fieldset>
    <br />
    <input type="button" value="Send" onclick="startAjax(this.form)"/>
  </form>
  <div id="ajax"/>

</body>
</html>

```

10.5.3.3 Quellcode der zweiten Seite

```

<?php

class Number {
  public $zahl=0;
  public function __construct($zahl) {
    $this->zahl = $zahl;
  }
} // Number

class Result {
  public $error='';
  public $iserror=false;
  public $liste= array();
} // Result

$ok = true;
$zahl1 = 0;
$zahl2 = 0;
$error = '';

if ( isset($_GET['zahl1']) ){ // Ist gesetzt?
  $str_zahl1 = $_GET['zahl1'];
  if (is_numeric($str_zahl1)) {
    $zahl1 = intval ( $str_zahl1, 10 );
  }
  else {
    $ok = false;
    $error.="Error in number conversion: the variable 'zahl1' is not any
number<br />";
  }
}

```

```

}
else {
    $ok = false;
    $error.="Error: The param 'zahl1' are not passed<br />";
}

if ( isset($_GET['zahl2']) ){    // Ist gesetzt?
    $str_zahl2 = $_GET['zahl2'];
    if (is_numeric($str_zahl2)) {
        $zahl2 = intval ( $str_zahl2, 10 );
    }
    else {
        $ok = false;
        $error.="Error in number conversion: the variable 'zahl2' is not any
number<br />";
    }
}
else {
    $ok = false;
    $error.="Error: The param 'zahl2' are not passed<br />";
}

if ($ok) {
    // ist zahl1>zahl2, dann tauschen
    if ($zahl1>$zahl2) {
        $zahl = $zahl1;
        $zahl1 = $zahl2;
        $zahl2 = $zahl;
    }
    $result = new Result();

    for ($i=$zahl1; $i<=$zahl2; $i++) {
        $number = new Number($i);
        $result->liste[] = $number;
    }
    echo json_encode($result );
}
else {
    $result = new Result();
    $result->error = $error;
    $result->iserror=true;
    echo json_encode($result);
}
?>

```

10.6 Probleme von Ajax

Historie des Browsers / Bookmarks

Abhilfe:

- Speichern der Parameter des Aufrufs

11 XAMPP-Installation

11.1 Windows

Link: <https://xampp.site/downloads/>

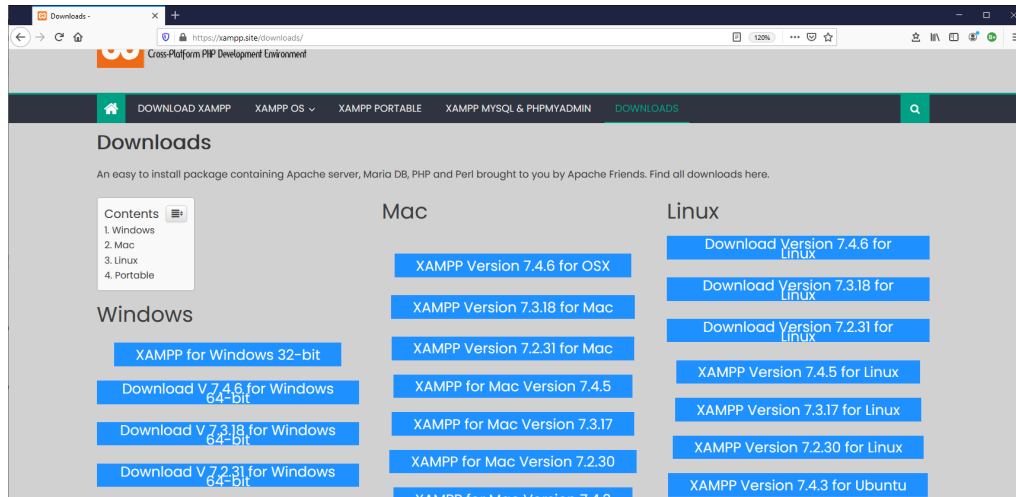


Abbildung 28 XAMPP-Downloadseite

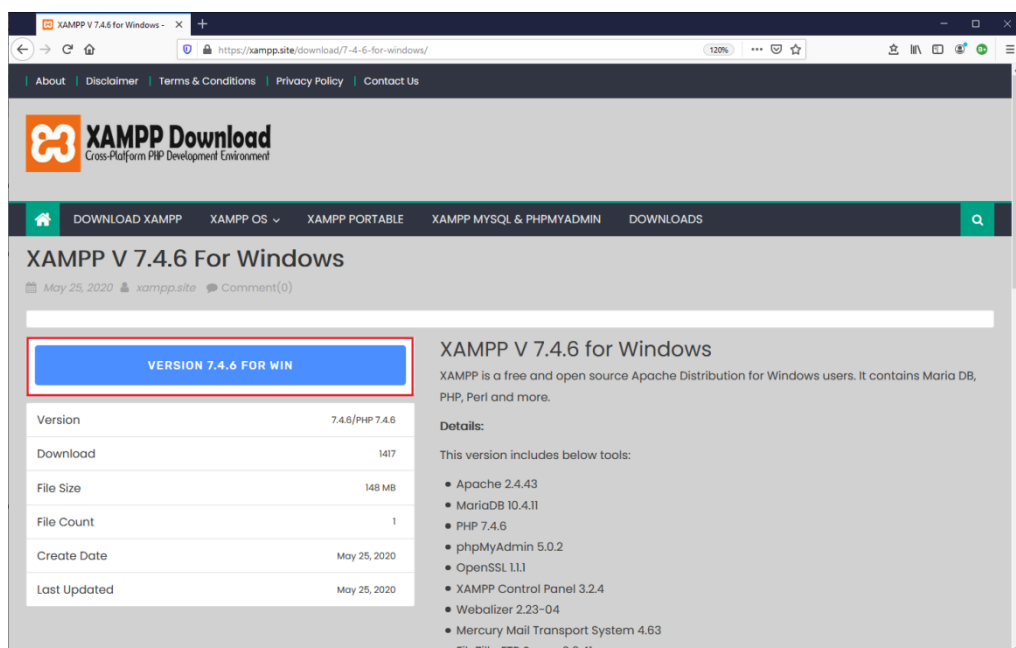


Abbildung 29 Starten des Downloads

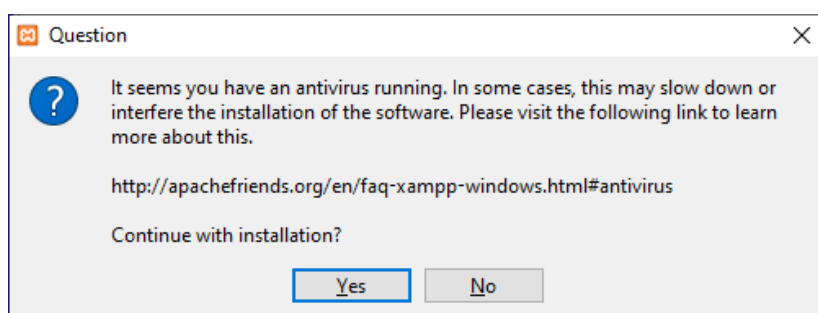


Abbildung 30 Frage nach dem AntiVirusprogramm

Hinweis:

- Um Rechte-Probleme zu vermeiden, sollte man als Installationspfad **C:** oder **D:** wählen.

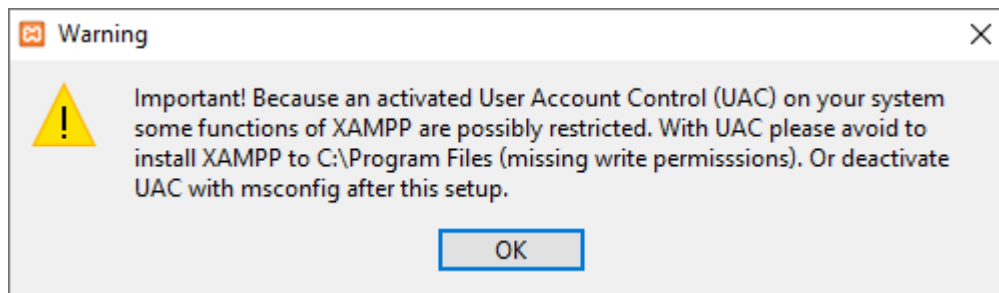


Abbildung 31 Rechteproblematik

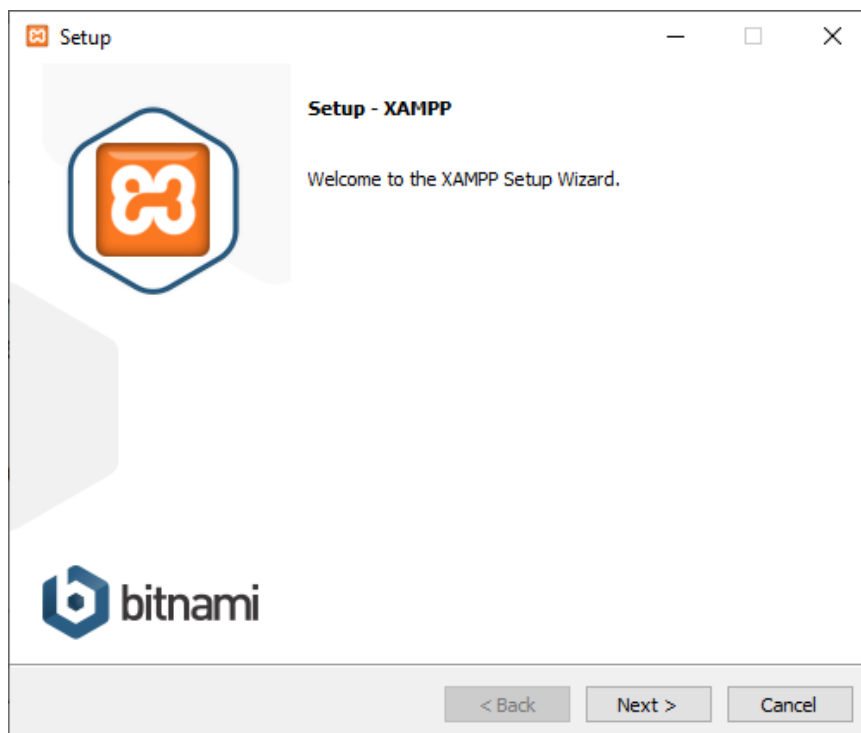


Abbildung 32 Echtes Startfenster

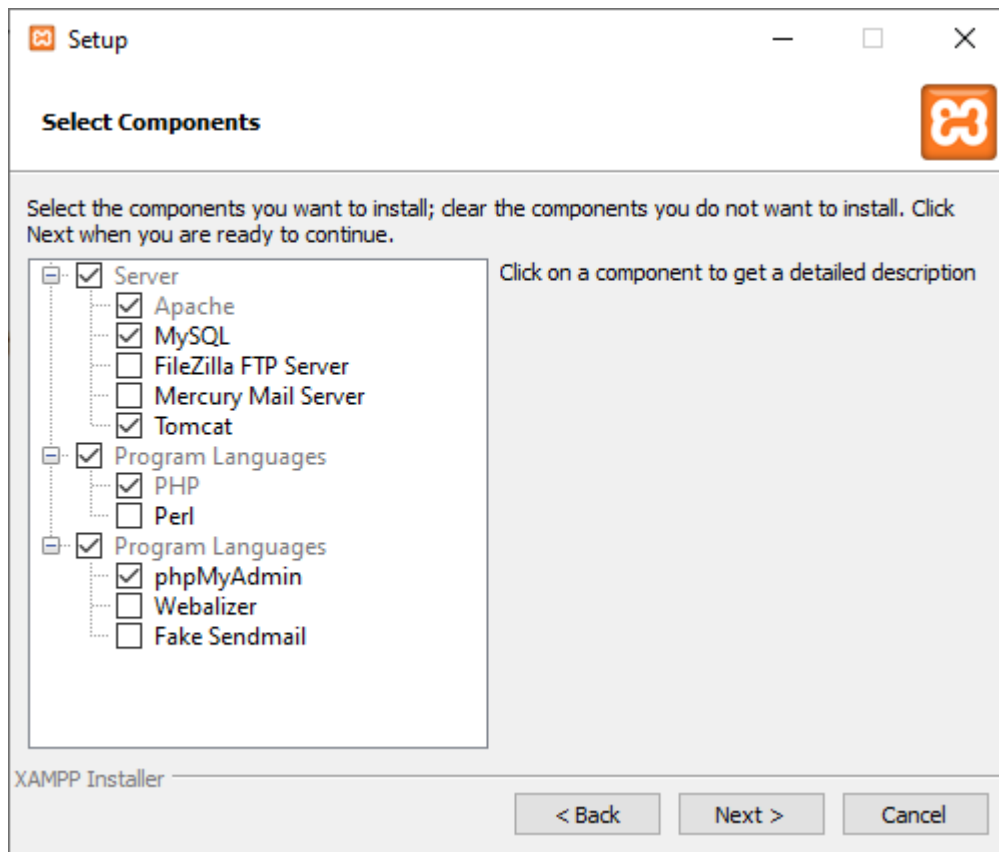


Abbildung 33 Auswahl der Komponenten

Hinweis:

TomCat benötigt man nur, wenn man mit JSP programmieren will.

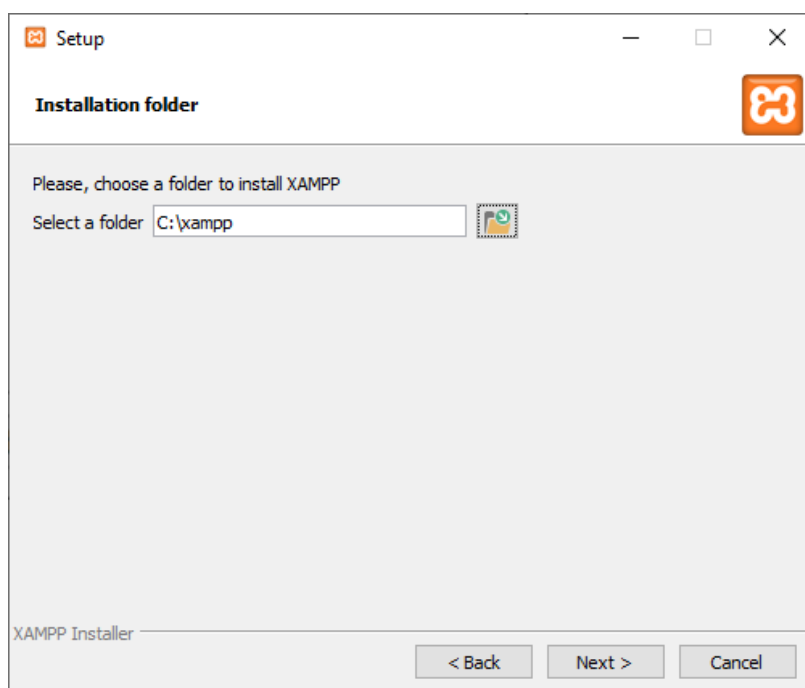


Abbildung 34 Installationspfad

Hinweis:

- Um Rechte-Probleme zu vermeiden, sollte man als Installationspfad **C:** oder **D:** wählen.

11.2 Starten vom XAMPP

XAMPP wird mit dem folgenden Befehlen gestartet:

- c:\xampp-control.exe
- d:\xampp-control.exe

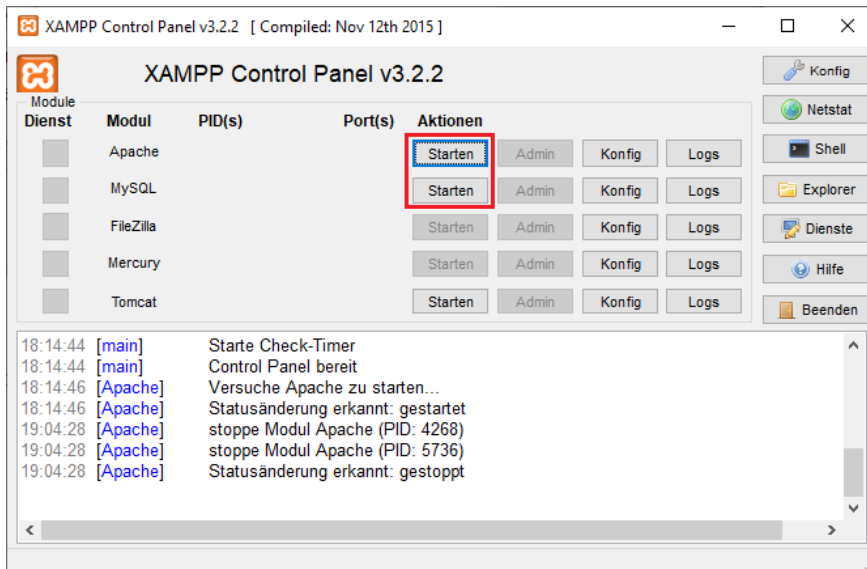


Abbildung 35 Auswahl im XAMPP-Dialog

Für PHP/Apache benötigt man nur den Schalter „Apache/Starten“.
Für Datenbanken benötigt man nur den Schalter „MySQL/Starten“.

Nach einem erfolgreichen Starten sieht der Dialog dann so aus:

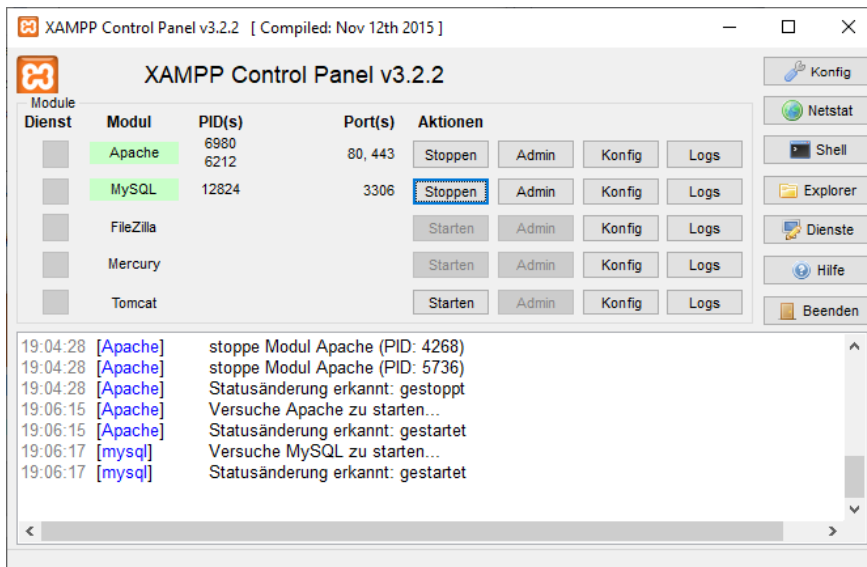


Abbildung 36 Erfolgreiche Auswahl im XAMPP-Dialog

Bei Problemen mit anderen Programmen muss man den Port ändern.
Siehe nächstes Kapitel.

Wichtig:

- Dateien werden im Verzeichnis „c:\xampp\htdocs“ gespeichert

11.3 Port-Probleme

Wenn es zu Port-Problemen kommt, kann man den Port ändern.

Wichtig:

- Dieser Port muss in den Eigenschaften der IDE eingetragen werden.

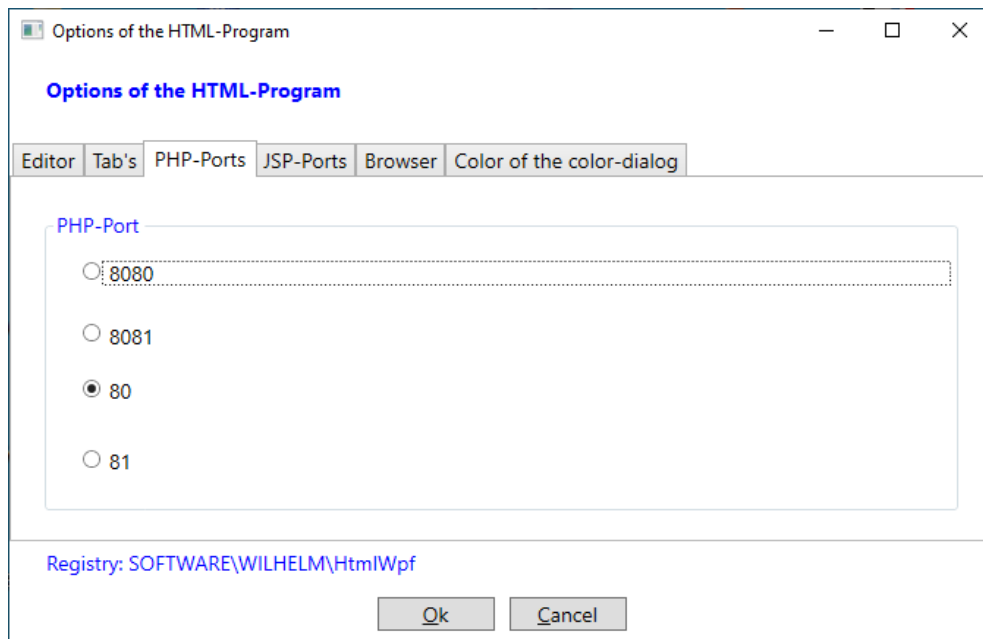


Abbildung 37 Eintragen des Ports für PHP

Meine IDE's wechseln dann automatisch zum neuen Port:

- localhost:8080/MI-4Sem/ajax/ajax01a.xhtml

Abhilfe

- Starten „XAMPP“
- Starten „Apache“
- Schalter „Config“ in der Reihe Apache
- Datei: „Apache: httpd.conf“
- Ändern der Ports von 80 auf 8080
- Der Aufruf muss dann geändert werden:
 - Aufruf: `http://localhost:8080/myfirst.php`

11.4 Apple

Link: <https://xampp.site/downloads/>

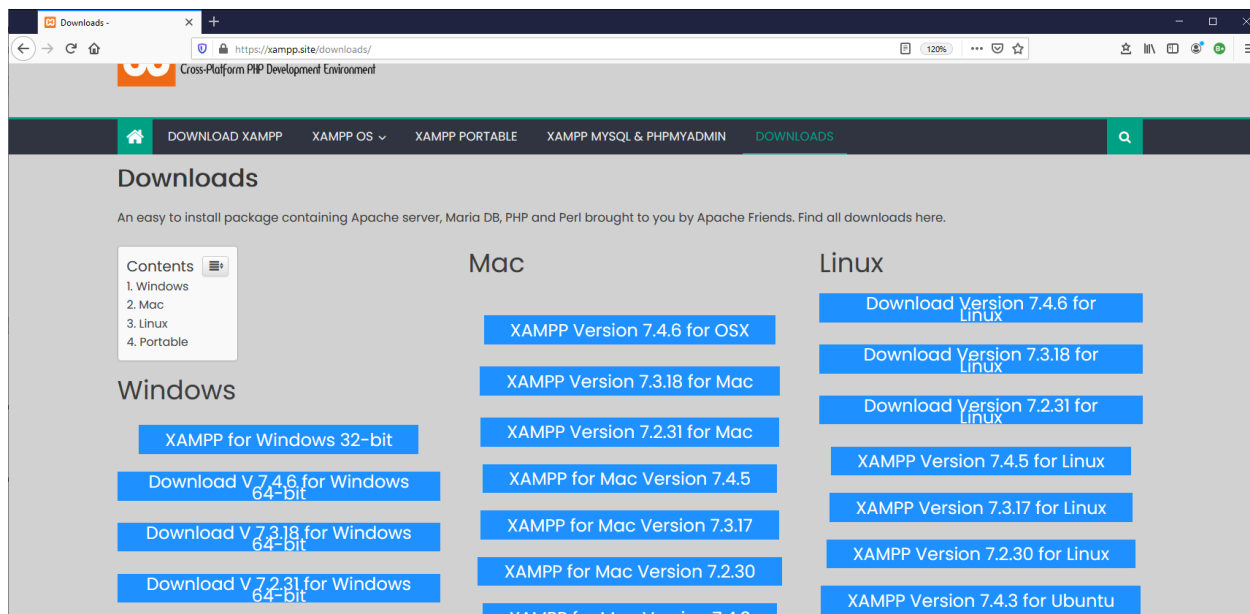


Abbildung 38 XAMPP-Downloadseite

Download die dmg-Datei

Starten der dmg-Datei

Nach dem Download des DMG-Images öffne ich das DMG-Image, falls dies nicht automatisch geschehen ist, und ziehe den XAMPP-Ordner in den Applications-Ordner (Programme), siehe Screenshot.

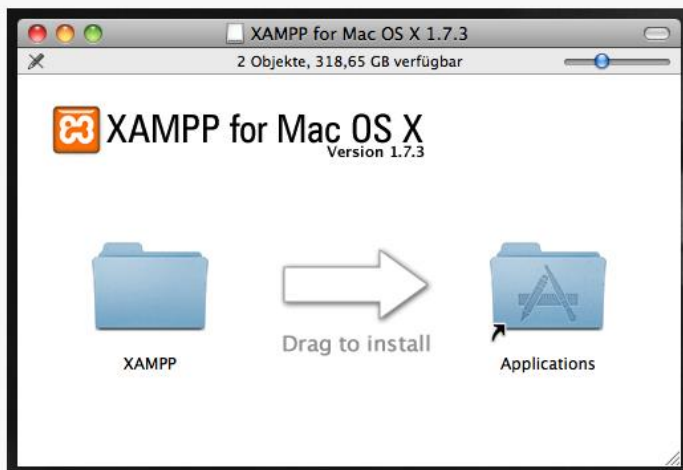


Abbildung 39 Mit Drag&Drop XAMPP nach Application ziehen

Nach der Installation in „Programm“ xampp.app starten

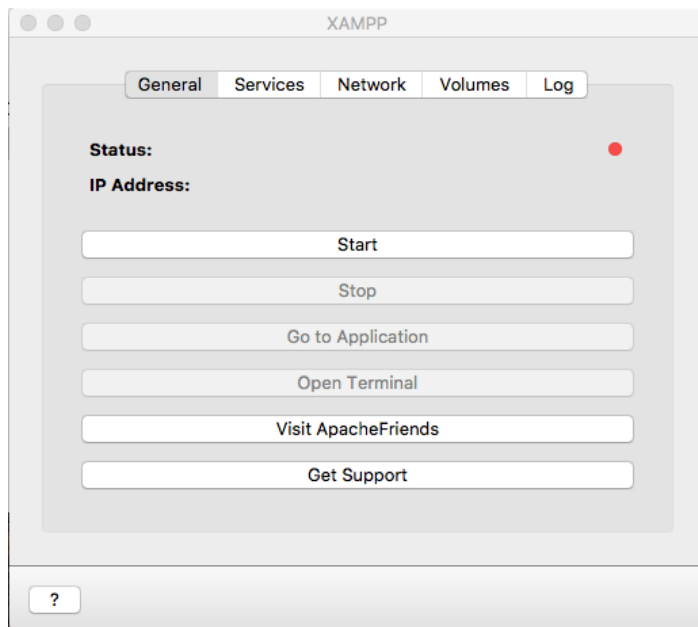


Abbildung 40 Startdialog von XAMPP

Nun den Schalter „Start“ betätigen

Nach dem Starten zeigt XAMPP die IP-Adresse an

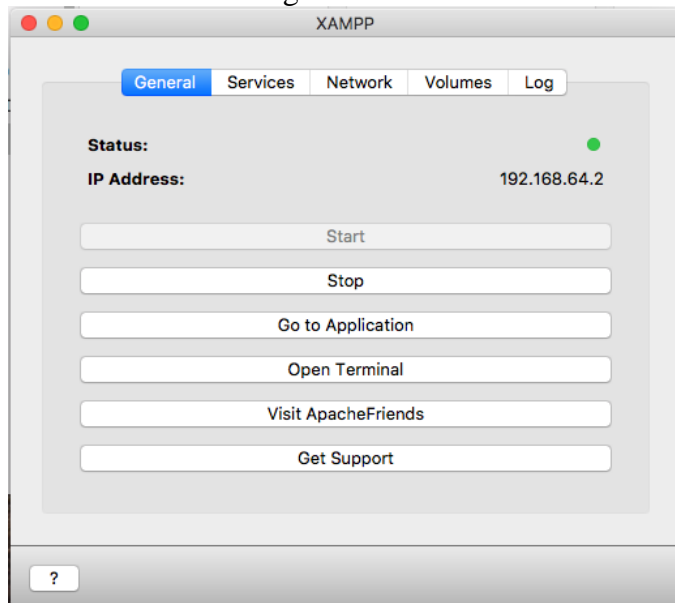
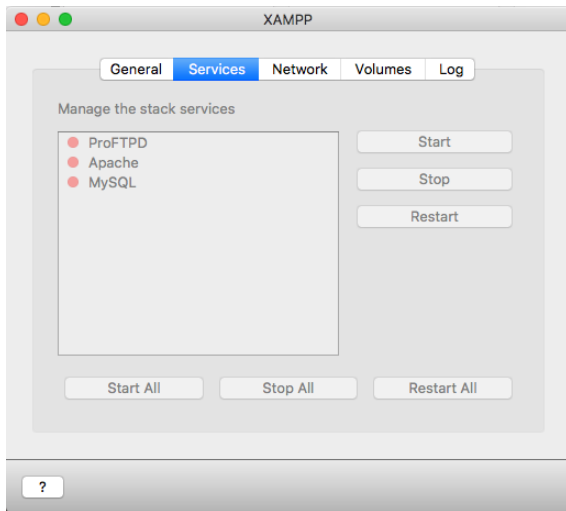


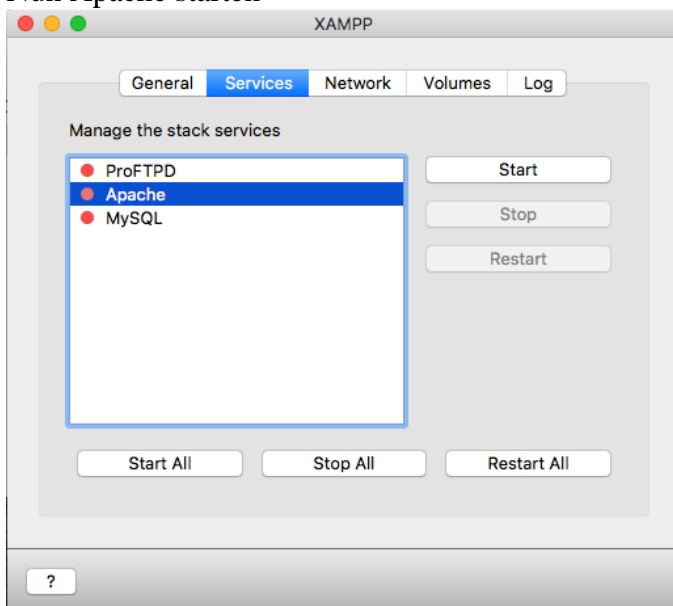
Abbildung 41 IP-Adresse: 192.168.64.2

Im zweiten Register mindestens „Apache“ starten:

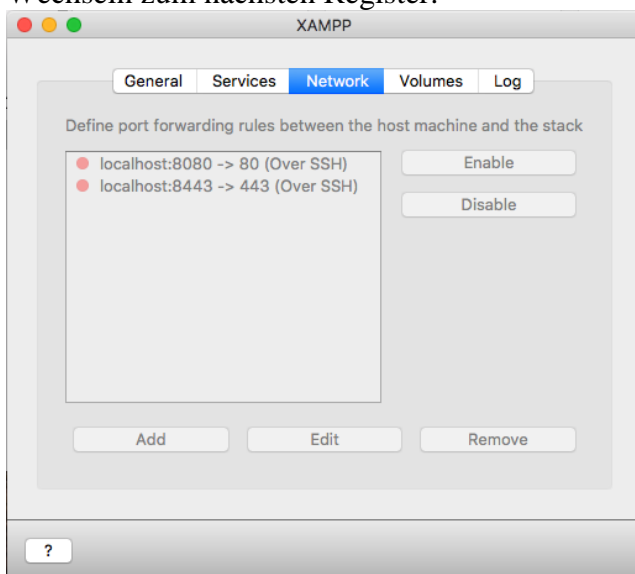
- Apache anklicken, dann den Schalter „Start“ anklicken
- Er sollte dann grün leuchten



Nun Apache starten



Wechseln zum nächsten Register:



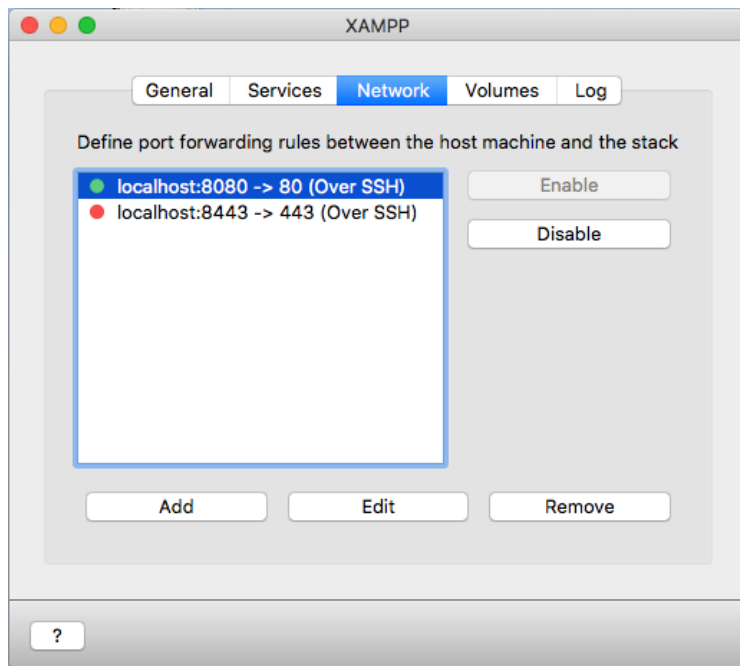


Abbildung 42 Port 8080 freischalten

Nun die Verknüpfung mit dem Ordner „htdocs“ erstellen

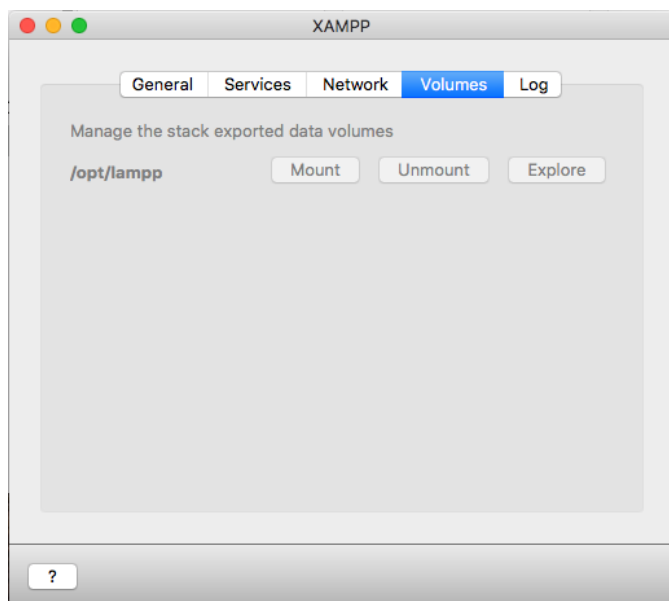


Abbildung 43 Wechseln zum Register Volumes

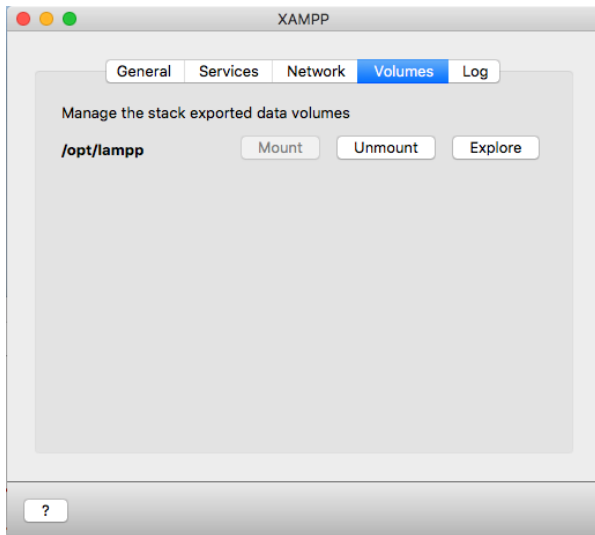
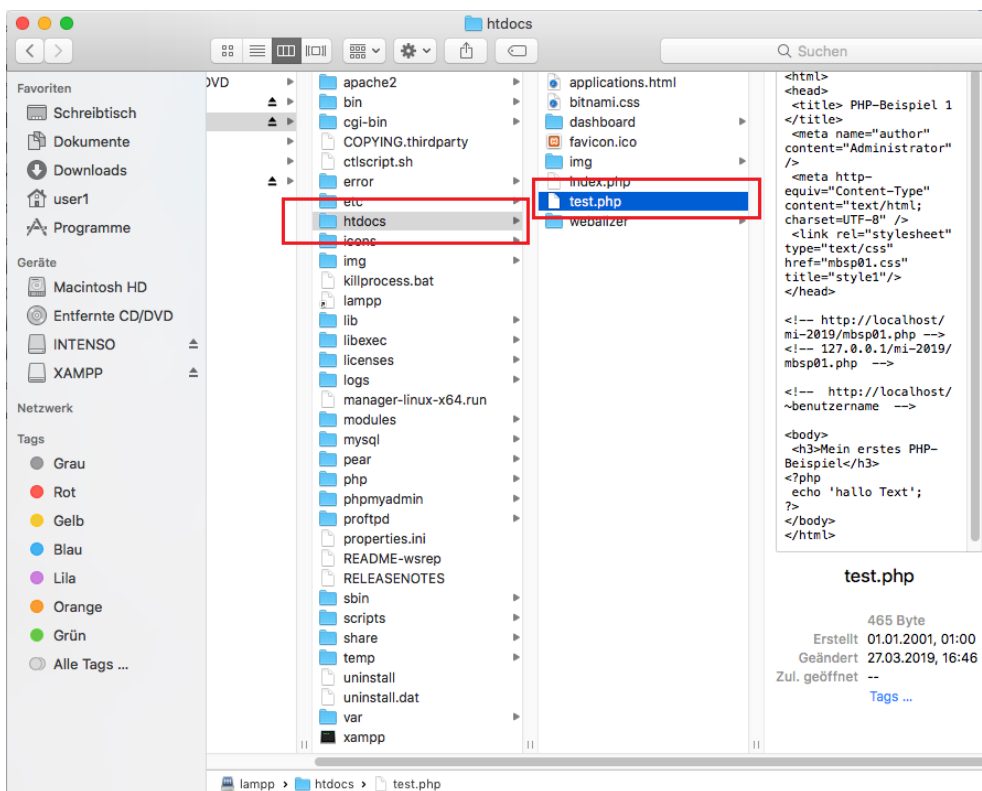
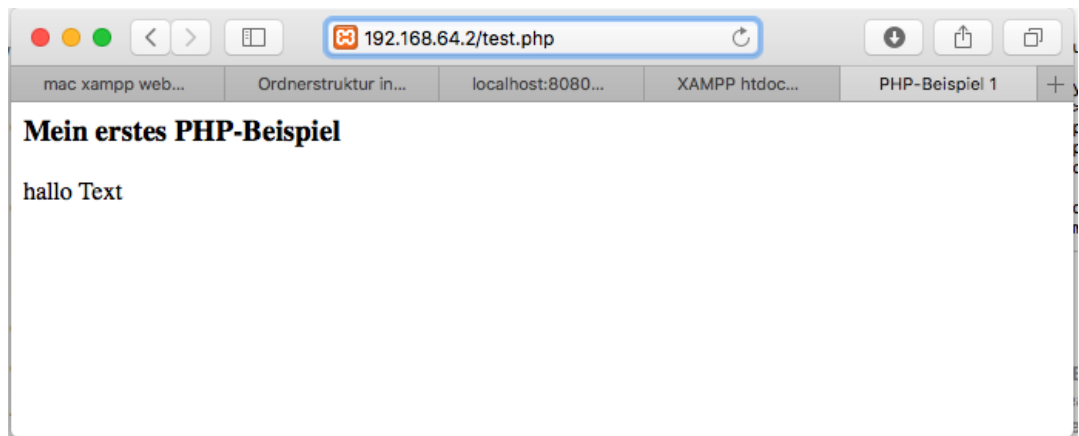


Abbildung 44 Anklicken des Schalters "Mount"

- Schalter „Explore“ aufrufen
- Es erscheint der Finder mit dem xampp-Verzeichnissen
- Htdocs auswählen und eine Datei im Ordner speichern



Aufruf der Datei mit der IP-Adresse, einem optionalen Ordner und dem Dateinamen.



12 HTML-Editoren

12.1 HTML (Windows)

12.2 HTMLRtf (Windows)

12.2.1 iHTML (iOS, Linux)

12.3 iHTMLRtf (iOS, Linux)

12.4 iHTMLTab (iOS, Linux)

13 Stichwortverzeichnis

A

Abfragen der Parameter in PHP.....	91
addslashes.....	37
addslashes	37
Ajax	113
Beispiele	124
Json.....	122
Struktur	113
Array	105
count	29
Funktionen	30
slice.....	105
Arrays.....	27
Assoziierte Arrays	33

B

bin2hex	37
button	71

C

charAt	110
checkbox.....	67
chop.....	37
chr37	
chunk_split	37
convert_cyr_string.....	37
convert_uudecode.....	37
convert_uuencode.....	37
count.....	29
count_chars	37
crc32	38
crypt.....	38
CSS	11
div 13	
Farbdefinitionen	11
Feste Farben	12
Margin	16
Messeinheiten	11
Padding.....	16
Pseudo-Elemente	16
rahmen	15
Sonderzeichen	12
span	14
current	24

D

Datenbank	94
Datentypen	21
Delete	99
div 13	

E

each	24
echo	26, 38

Editor	150
HTML	150
HTMLRtf	150
iHTML	150
iHTMLRtf.....	150
iHTMLRtfTab.....	150
iHTMLTab	150
end.....	24
exception	35
explode	38

F

Farbdefinitionen	11
Feste Farben	12
file72	
Formular Elemente	65
Formulare	62
fprintf.....	38
Funktionen.....	35, 109

G

Get	64
get_html_translation_table.....	38
getConnection	95

H

Hashtables	33
hex2bin	38
hidden.....	69
HTML	7
html_entity_decode	38
htmlentities	38
htmlspecialchars.....	39

I

image	70
implode.....	39
Insert.....	98

J

Javascript	
Abfragen	106
Array.....	105
Do While.....	108
Else	106
For	108
Foreach.....	109
Funktionen	109
If 106	
Klasse.....	111
Logische Verknüpfungen	106
Operatoren.....	106
Schleifen	108
String Methoden	110
Switch	107
While	108
JavaScript	104
join.....	39
Json.....	122
JSON.....	113

JSON.parse	123
json_decode	123

K

Klasse	111
Klassen	56

L

label	66, 69
Ländercode	52
lcfirst	39
levenshtein	39
localeconv	39, 49
ltrim	39

M

Margin	16
md5	39
Messeinheiten	11
metaphone	39
money_format	39
MySQL	94
Delete	99
getConnection	95
Insert	98
query	95
Sequence	97
SQL-Rahmen	99
Update	98

N

n2br	39
next	24
nl_langinfo	39, 50
number	69
number_format	40

O

onsubmit	85
OOP	56
Abstrakte Klassen	57
Beispiel mit abgeleiteter Klasse	58
Beispiel mit Konstruktor	58
Eigenschaften	56
Einfaches Beispiel	57
Eintragen in eine Liste	60
Interface	57
ord	40

P

Padding	16
parse_str	40
password	67
Pattern	88
EMail	89
Geburtsdatum	89
Großbuchstaben/Ziffern	89
IPv4	89

Nachname	88
Passwort	90
Straße	89
PHP	18, 21
Abfragen	22
Arrays	27
Assoziierte Arrays	33
current	24
Do While	23
each	24
echo	26
Else	22
end	24
exception	35
For	23
Foreach	24
Funktionen	35
If	22
Logische Verknüpfungen	22
next	24
Operatoren	21
prev	24
reset	24
Schleifen	23
Sessions	72
String Methoden	37
Switch	22
try-catch	35
Variablen	21
While	23
Post	64
prev	24
printf	40
Pseudo-Elemente	16

Q

query	95
quoted_printable_decode	40
quoted_printable_encode	40
quotemeta	40

R

radiobutton	67
rahmen	15
reset	24, 71
rtrim	40

S

Schleifen	
current	24
Do While	23, 108
each	24
end	24
For	23, 108
Foreach	24, 109
next	24
prev	24
reset	24
While	23, 108
select	68
Sequence	97
Sessions	72
setlocale	40

sha1	41
similar_text	41
slice	105
Sonderzeichen	12
soundex	41
span	14
sprintf	41
SQL-Rahmen	99
sscanf	41
str_getcsv	41
str_ireplace	42
str_pad	42
str_repeat	42
str_replace	42
str_rot13	42
str_shuffle	42
str_split	43
str_word_count	43
strbrk	45
strcasecmp	43
strchr	43
strcmp	43
strcoll	43
strcspn	43
String Methoden	37, 110
strip_tags	44
stripcslashes	44
stripos	44
stripslashes	44
stristr	44
strlen	44
strnatcasecmp	44
strnatcmp	44
strncmp	45
strpos	45
strrchr	45
strrev	46
strripos	46
strrpos	46
strspn	46
strstr	45, 46
strtok	46
strtolower	46
strtoupper	46
strtr	46
submit	70
substr	47
substr_compare	47
substr_count	47
substr_replace	47

T

text	66
trim	47
try-catch	35

U

ucfirst	47
ucwords	48
Update	98

V

validation	85
------------------	----

vprintf	48
vsprintf.....	48

W

wordwrap	48
----------------	----

X

XAMPP	
Apple	144
Windows.....	139

Z

Zeilenumbrüche.....	7, 8, 10
---------------------	----------