FH Salzburg

VO Web-Technologien

29.11.2023, Oliver Jung

Technik
Gesundheit
Medien

JavaScript Frameworks – Beispiele: Angular

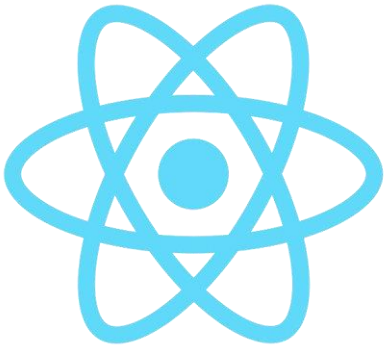
Angular – https://angular.io

- Nachfolger von AngularJS, Open-Source
- Lizenz: MIT, seit 2016
- basiert auf TypeScript (typisierte Variante von JavaScript)
- **Konzepte**:
  - MVC-Design-Entwurfsmuster
  - zwei-Wege-Datenbindung
  - klare und starre Strukturen

Geeignet für Projekt mit …

- vielen dynamischen Inhalten
- Unternehmensanwendungen

JavaScript Frameworks – Beispiele: React



React – https://reactjs.org

- Entwickelt von Facebook, jetzt Open-Source
- Lizenz: BSD, seit 2011
- **Konzepte**:
  - Virtual DOM und DOM-Diffing (nur benötige Elemente werden im DOM performant geändert)
  - deklarative, hierarchische Komponenten
  - Großteil wird mit JavaScript realisiert (keine separaten HTML-Dateien)

Geeignet für Projekte, die …

- Vielseitigkeit erfordern
- eine gute Suchmaschinen-Optimierung benötigen

JavaScript Frameworks – Beispiele: Vue.js

Vue.js – https://vuejs.org

- Ähnlichkeiten mit React und Angular, jedoch schlanker
- Lizenz: MIT, seit 2014
- Im Unternehmenskontext nicht so häufig verwendet
- **Konzepte**:
  - geringe Größe: nur 18KB groß (gzip)
  - einsteigerfreundlich

Geeignet für Projekte, die …

- auf Geschwindigkeit optimieren
- sehr schlanke Applikationen benötigen
- eine Integration mit bestehenden Applikationen voraussetzen

JavaScript Bibliotheken – Beispiele: jQuery

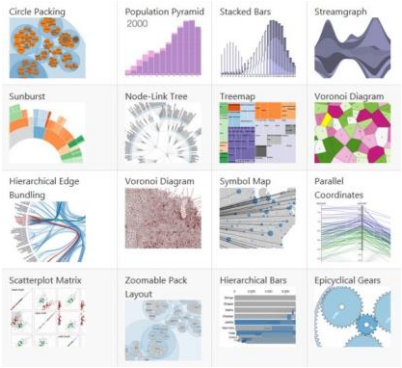
jQuery ist bekannteste JavaScript-Bibliothek – https://jquery.com

- **Funktionsumfang**:
  - DOM-Navigation und -Manipulation
  - erweiterte Event-Mechanismen
  - Hilfsfunktionen wie inArray() und each()
  - AJAX-Funktionalitäten
  - erweiterbar durch Plug-Ins
- Wird von über 75% der meistbesuchten Webseiten verwendet
- Kostenfrei und Open Source (MIT-Lizenz)
- Unterstützt nahezu alle aktuellen Browser

JavaScript Bibliotheken – Beispiele: D3

D3 – Data-Driven Documents – https://d3js.org

- **Funktionsumfang**:
  - Dokumente werden durch Daten verändert
    - Manipulation von HTML, SVG, CSS
  - interaktive HTML-Tabellen
  - interaktive Diagramme in SVG
  - Verarbeitungen von Kartendaten, z.B. GeoJSON
- Kostenfrei und Open Source (MIT-Lizenz)
- Unterstützt nahezu alle aktuellen Browser
- **Beispiele**: https://github.com/d3/d3/wiki/Gallery



JavaScript Bibliotheken – Beispiele: Lit

Lit – https://lit.dev/

- **Funktionsumfang**:
  - Definition von eigenen Tags für HTML, sogenannten WebComponents
    - Beispiel:

```
1    <flexible-rating value="3.6">
2    </flexible-rating>
```

  - Abstraktion von HTML, CSS und JavaScript in eigener Komponente
  - Bindung von Daten und Unterstützung von Events
- Kostenfrei und Open Source (BSD-Lizenz)
- Unterstützt nahezu alle aktuellen Browser
  - Bei Bedarf: Laden von Polyfills zum Erweitern der Kompatibilität mit älteren Browsern

Lit

Web-Framework: Django

Django – https://www.djangoproject.com

- Beliebtes Python-Framework
- Lizenz: BSD, seit 2005
- Schnell von der Idee zur Umsetzung
  - Code-Generatoren
  - Automatisch generierte Administrations-Oberfläche
- Viele eingebaute Features
  - Authentifizierung
  - HTML-Formular-Helper
  - Unterstützung für HTTP-Cache
- „Don‘t Repeat Yourself“ (DRY)
  - Redundanz vermeiden, exakt einen Ort (im Quellcode) für jedes Konzept

Web-Framework: Ruby on Rails

Ruby on Rails (kurz **Rails**)– https://rubyonrails.org

- Wohl berühmtestes Ruby-Web-Framework
- Lizenz: MIT, seit 2005
- Vorreiter für moderne Web-Frameworks in vielen Belangen
  - Model-View-Controller
  - ActiveRecord: einfacher Datenbankzugriff durch Abbilden von Datenbank-Tabellen und -Zeilen in Klassen und Objekten
- Florierendes Ökosystem mit zahlreichen Erweiterungen (sogenannte Gems)
  - manche Gems werden vom De-Facto-Standard zum Teil des Frameworks
- „Convention over Configuration“

Was ist PHP?

- PHP: Hypertext Preprocessor
- ausschließlich für die Webserver-Programmierung konzipiert
- PHP-Programme sind „Server-Programme“
- erlaubt das Erzeugen dynamischer Webseiten
- unterstützt die einfache Bearbeitung von HTML-Formularen
- unterstützt die Zusammenarbeit mit vielen gängigen Datenbank-Systemen
- systemunabhängig, wird von vielen verschiedenen Typen von Webservern einheitlich unterstützt
- relativ leicht erlernbar
- frei verfügbar

Ein kleines Beispiel

```
<html>
  <head>
  </head>
 <body>
   <?php
     echo "hello world";
   ?>
 </body>
</html>
```

Einbettung von PHP in HTML

- Kurzform innerhalb einer Markierung

<?php [PHP-Anweisungen] ?>

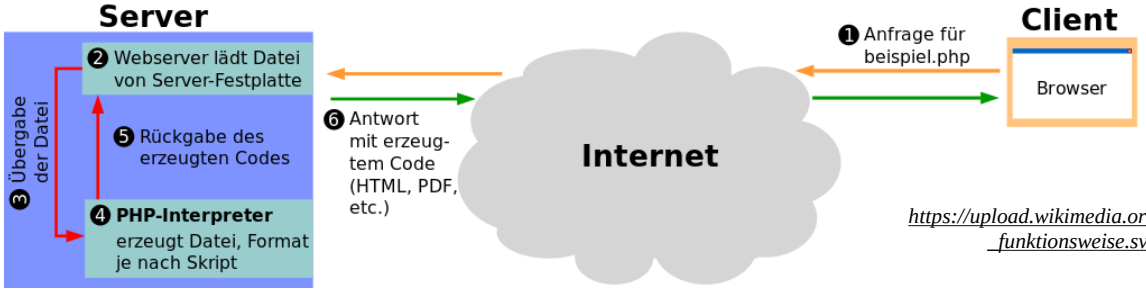
- Langform innerhalb eines Blockes

<script language = "php"> [PHP-Anweisungen] </script>

```
<html>
  <head> <title> Titelzeile der Datei </title> </head>
  <body> <p> Die erste Zeile in HTML </p>
    <?php echo "<p>Die zweite Zeile in PHP</p>"; ?>
    <p> Die dritte Zeile in HTML </p>
    <script language="php">
      echo "<p>Die vierte Zeile in PHP</p>";
      echo "<p>Die fünfte Zeile in PHP</p>";
    </script>
    <p> Die sechste Zeile in HTML </p>
    <?php echo "<p>Die siebte Zeile in PHP</p>"; echo "<p>Die achte Zeile in PHP</p>"; ?>
  </body>
</html>
```

Webserver und PHP-Interpreter

- Um eine PHP-Datei im Rahmen einer Webanwendung ausführen zu können, benötigt man ein System, das mit den in der Datei enthaltenen Anweisungen umgehen kann
- Ein Browser kann dies nicht. Aus diesem Grund setzen wir einen Webserver mit einem PHP-Interpreter ein
- Dazu eignet sich besonders das Software-Paket namens **XAMPP** (https://www.apachefriends.org/de/index.html)
- XAMPP ist eine Zusammenstellung von freier Software und ermöglicht deren einfache Installation und Konfiguration (Webserver Apache, Datenbank MariaDB/MySQL, Skriptsprachen Perl und PHP)
- XAMPP ist nicht für den Einsatz als Produktivsystem gedacht, sondern für Entwickler, die möglichst schnell ein kompaktes Testsystem aufsetzen möchten



https://upload.wikimedia.org/wikipedia/commons/thumb/6/67/PHP_funktionsweise.svg/1024px-PHP_funktionsweise.svg.png

PHP-Interpreter

```
<html>
  <head>
   <title>Informatik</title>
  </head>
  <body>
    <h1>Eine erste PHP-Seite</h1>
    <?php
      echo "Informatik macht Spaß!";
    ?>
  </body>
</html>
```

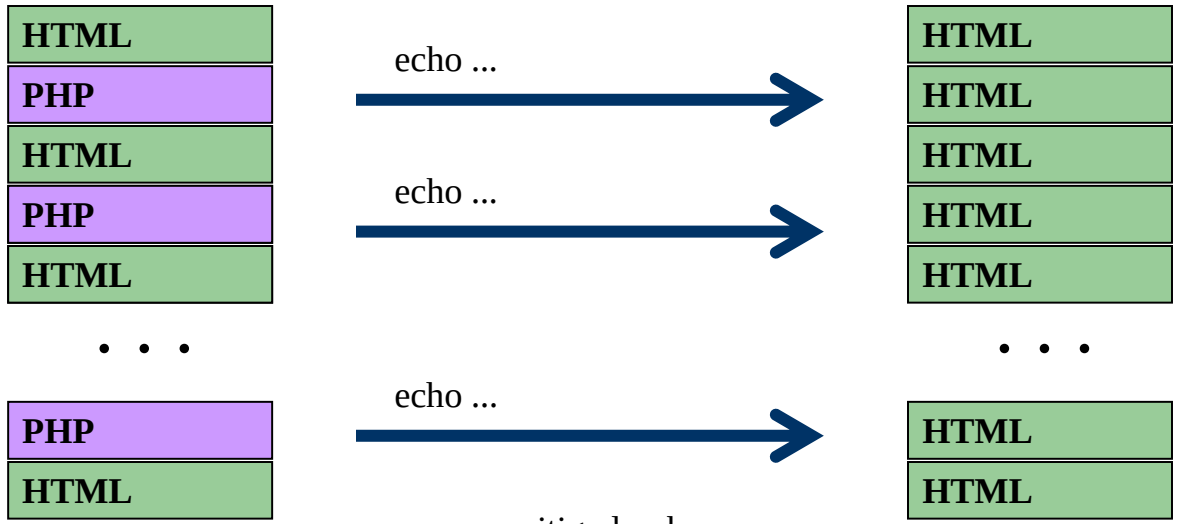
PHP-Interpreter

```
<html>
  <head>
   <title>Informatik</title>
  </head>
  <body>
    <h1>Eine erste PHP-Seite</h1>

    Informatik macht Spaß!

  </body>
</html>
```

Interpreter: PHP → HTML



Aufruf einer Webseite

Dateisystem

C:\xampp\htdocs	\	kurs1\php\	index.html
„Document-Root“		Pfad	Zieldokument

URL

http	://	localhost	/	kurs1/php/	index.html
Protokoll		Name des Rechners (Domainname)		Pfad	Zieldokument

Die URL beginnt mit dem Protokoll, diesem folgt der Rechnername. Der Pfad zum Zieldokument wird relativ zur „Document-Root“ angegeben.

PHP - Variablen und Werte

- Variablen definieren

```
$Hinweis = "Fehler 4711";
$i, $x;
$kontostand = 420;
```

- Werte von Variablen ändern

```
$x = $x + $i;
$Hinweis = 'Alarm';
$a = "Max";
$a = 5;
```

Variablen und Datentypen

- Datentypen richten sich nach dem Zusammenhang der Nutzung
- eine Variable kann ihren Datentyp innerhalb eines Programms wechseln
- keine Variablendeklaration notwendig
- Regeln für Variablennamen:
  - müssen mit einem Dollar-Zeichen beginnen
  - dürfen keine Leerzeichen oder Umlaute enthalten
  - einzig erlaubtes Sonderzeichen: _
  - erste Zeichen sollte Buchstabe sein
  - dürfen nicht identisch mit reservierten Wörtern sein
  - Groß-/Kleinschreibung wird unterschieden

Umwandlung von Zeichenketten in Zahlen

- Automatische Konvertierung
  - Eine gültige Folge von numerischen Zeichen beinhaltet:
    - ein Vorzeichen (optional)
    - eine oder mehrer Ziffern
    - einen Dezimalpunkt (optional)
    - einen Exponent (optional), der Exponent ist ein kleines oder großes „e“, gefolgt von einer oder mehreren Ziffern
  - Eine Zeichenkette wird interpretiert als
    - ganze Zahl, falls sie nur Ziffern beinhaltet
    - Zahl mit Nachkomma, falls sie neben Ziffern die Zeichen „.“, „e“ oder „E“ beinhaltet
- Explizite Konvertierung
  - doubleval(), intval()
  - $a = "435"; → $a = intval($a);
  - $b = "22.6"; → $b = doubleval($b);

PHP - Allgemeine Regeln

- Anweisungen

```
$Zahl = 41;
$Quadrat = $Zahl * $Zahl;
if($Zahl > 1000) $Zahl = 0;
```

- Anweisungsblöcke

```
$if($Zahl > 1000) {
  $Zahl = 0;
  Neustart();
}
```

- Kommentare

```
//...
```

Operatoren

- Zuweisungsoperator
  - `$x = 5`
- Vergleichsoperatoren
  - == != < <= > >=
- Berechungsoperatoren
  - + ++ += - * / %
- Logische Operatoren
  && || !
- Operatoren zur Zeichenkettungverknüpfung
  - $Name = $Vorname . “, “ . $Zuname

Zeichenketten

```
<?php
  $a = 5;
  $b = 6;
  $c = $a + $b;
  $anrede = "Herrn";
  $vorname = "Max";
  $ganzername = $anrede . " " . $vorname . " ";
  $nachname = 'Mustermann';
  $ganzername .= $nachname;
  echo "Dieses Programm ist von $ganzername<p>";
  echo "Das Ergebnis der Addition ist " . $c . "<p>";
?>
```

Operator-Reihenfolge

Assoziativität	Operator
keine Richtung	new
rechts	[
rechts	! ~ ++ -- (int) (float) (string) (array) (object) @
links	* / %
links	+ - .
links	<< >>
keine Richtung	< <= > >=
keine Richtung	== != === !==
links	&
links	^
links	|
links	&&
links	||
links	? :
rechts	= += -= *= /= .= %= &= |= ^= <<= >>=
rechts	print
links	and
links	xor
links	or
links	,

Verzweigung / Schleifen

```
if (Bedingung) {
  Anweisungen;
} else {
  Anweisungen;
}

for ([Startausdruck]; [Bedingung]; [Iterations-Ausdruck]) {
  Anweisungen;
}

while (Bedingung) {
  Anweisungen;
}

do {
   Anweisungen;
} while (Bedingung);
```

PHP - Funktionen

- Eine Funktion ist ein Block mit Anweisungen, dem ein Name zugewiesen wird.
- Eine Funktion hat eine Liste von Parametern, die auch leer sein kann
- Eine Funktion kann einen Wert zurückgeben

```
function meineFunktion (param1, param2) {
  Anweisungen;
  return Ausdruck;
}
```

Funktionen

- Beispiel:

```
<?php
  function add($z1, $z2){
    $summe = $z1 + $z2;
    return $summe;
  }
?>
...
<?php echo "<p>Summe: " . add(13,2) . "</p>"; ?>
```

Vorhandene PHP-Funktionen

- Mathematische Funktionen
  - M_PI, pi(), exp(1), sqrt($a), pow($a,3), log($a), ln($a)
  - floor($a), ceil($a), round($a), max($a,$b), min($a, $b, $c)
  - sin($a), cos($a), tan($a)

- Zeichenketten-Funktionen
  - strlen($s), strtolower($s), strtoupper($a), strrev($a),
  - substr($s,a,b), strpos($s, „.“, 5)
  - strcmp($s1,$s2), strcasecmp($s1,$s2)

    - → http://de.php.net/manual/de/

Verhalten bei Parameterübergabe

- call-by-value
  - Übergabe der Parameter als Kopie (call-by-value), eine Veränderung der Kopien hat keine Rückwirkung auf das Original: Diese Methode wird angewendet, falls man nur Werte in die Funktion hineinliefern möchte
- call-by-reference
  - Übergabe der Parameter als Verweis (call-by-reference) auf das Original, eine Veränderung hat Rückwirkung auf das Original: Diese Methode wird beispielsweise dann angewendet, wenn mehr als ein Wert aus einer Funktion zurückgeliefert werden soll.

Verhalten bei Parameterübergabe - Beispiel

```
function vtauschen($a, $b){
  $temp = $a;
  $a = $b;
  $b = $temp;
}
```

```
function rtauschen(&$a, &$b){
  $temp = $a;
  $a = $b;
  $b = $temp;
}
```

```
$x = 12;
$y = 18;
echo "Methode 1, vorher: $x, $y";
vtauschen($x,$y);
echo "Methode 1, nachher: $x, $y";
```

```
$x = 12;
$y = 18;
echo "Methode 2, vorher: $x, $y";
rtauschen($x,$y);
echo "Methode 2, nachher: $x, $y";
```

Globale und lokale Variablen

- Globale Variablen
  - Eine globale Variable wird außerhalb von Funktionen definiert und steht nur außerhalb derselben zur Verfügung. Dies ist ein Unterschied zu vielen anderen Programmiersprachen. Falls man eine globale Variable innerhalb einer Funktion benutzen möchte, so muss sie dort entweder mit dem Schlüsselwort global bekannt gemacht oder als Parameter übergeben werden. Variablen, die ihren Ursprung außerhalb des Programms haben wie z. B. Werte aus Formularfeldern sind immer global.
- Lokale Variablen
  - Eine lokale Variable wird in einer Funktion definiert und steht nur innerhalb dieser Funktion zur Verfügung. Lokale Variablen gleichen Namens in unterschiedlichen Funktionen oder globale Variablen gleichen Namens haben nichts miteinander zu tun. Ein Parameter, der als Kopie an eine Funktion übergeben wird, ist dort lokal.

Globale und lokale Variablen - Beispiel

```
<head>
  <?php    function summiere(){
    echo "Variable z: $z<p>";
    global $x;
    $y = 35;
    $z = $x + $y;
    echo "Variable z: $z<p>";    }
  ?>
</head>
<body>
  <?php
    $x = 6;
    $y = 52;
    $z = $x + $y;
    summiere();
    echo "Variable z: $z<p>";
  ?>
</body>
```

Ausgabe:

Variable z:

Variable z: 41

Variable z: 58

Objekte

- Bis Version 4.x war PHP nur mit Einschränkungen objektorientiert (kein Überladen, keine Zugriffsmodifizierung, keine Default-Aktionen, …)
- Mit PHP 5.x (Release 2004, aktuell 8.x) steigt die Skript-Sprache in die Liga der objektorientierten Sprachen auf. Wer mit den Konzepten von Java, C++ oder C# vertraut ist, wird sich mit PHP schnell zurechtfinden
- Einsteiger und Freunde der strukturellen Programmierung müssen dennoch nicht verzagen. PHP bleibt rückwärtskompatibel.

Klassen

```php
class Person {
  private $name;
  private $firstame;

  public function getFullName() {
    return $this->firstname . " " . $this->name;
  }

  public function setName($newName) {
    $this->name = $newName;
  }
  public function setFirstname($newFirstname) {
    $this->firstname = $newFirstname;
  }
}
```

```php
$myPerson = new Person();
$myPerson->setName("Mustermann");
$myPerson->setFirstname("Max");
echo $myPerson->getFullName();
```

Klassen

```
class Mitarbeiter extends Person {
  private $nr;

  public function setNr($newNr){
    $this->nr = $newNr;
  }

  public function getNr(){
    return $this->nr;
  }
}
```

```
$myPerson = new Mitarbeiter();
$myPerson->setName("Mustermann");
$myPerson->setFirstname("Max");
echo $myPerson->getFullName();
$myPerson->setNr("123");
echo "<br>" . $myPerson->getNr();
```

Klasse PHPMailer

- Oft ist es notwendig, dass PHP Scripte Emails verschicken (für Kontaktformulare, Benachrichtigung, das Versenden von Zugangsdaten, o.ä.)
- PHP bietet mit der Funktion mail() eine entsprechende Möglichkeit. Diese ist jedoch sehr spartanisch und nur für die einfachsten Anwendungsfälle ausreichen
- Mit einem ansehnlichen Funktionsumfang wie die Unterstützung von HTML-Mails mit alternativer Textversion, dem Versand über SMTP-Server mit Authentifizierung oder der Unterstützung für eingebettete Bilder schafft PHPMailer Abhilfe

Siehe auch: https://github.com/PHPMailer/PHPMailer

Klasse PHPMailer

- Klasse einbinden
  require('class.phpmailer.php');
- Objekt erstellen
  $mail = new PHPMailer();
- Attribute setzen
  $mail->From = "max@mustermann.com";
  $mail->FromName = "Max Mustermann";
  $mail->AddAddress("test1@example.com");
  $mail->AddBCC("test2@example.com");
  $mail->Subject = „Eine wichtige Mail";
  $mail->Body = "Hallo Welt!";
- Email senden und auf Fehler prüfen

  if(!$mail->Send()) { } else { }

Klasse PHPMailer

- Versand über SMTP-Server

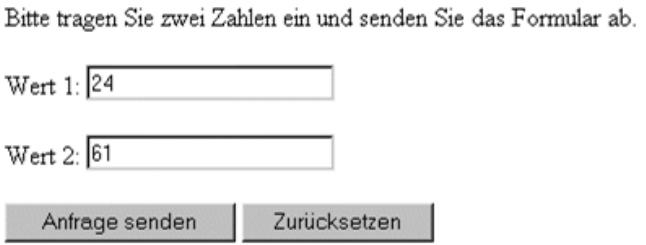
```
$mail->IsSMTP();
$mail->Host = "mail.host.de";

$mail->SMTPAuth = true;
$mail->Username = "benutzername";
$mail->Password = "geheim";
```

Variablen aus Formularen

HTML-Formular:

```
<html>
  <body> Bitte tragen Sie zwei Zahlen ein.<br>
  <form action = „seite5.php" method = „get">
      Wert 1: <input name = "w1"><p>
      Wert 2: <input name = "w2"><p>
      <input type = "submit">
      <input type = "reset">
    </form>
  </body>
</html>
```



Variablen aus Formularen

PHP-Seite (seite5.php):

```
<html>
  <body>
    <?php
      $erg = $_GET["w1"] +  $_GET["w2"];
      echo "Die Summe von " .
           $_GET["w1"] . " und  " .
           $_GET["w2"] . " ist $erg";
    ?>
  </body>
</html>
```

PHP und die MySQL-Datenbank

- Verbindung aufbauen
  - **mysql_connect()**
  - öffnet eine Verbindung zum MySQL-Datenbank-Server. In den Klammern können bis zu drei Parameter stehen: Hostname, Benutzername und Kennwort
  - Falls die Verbindung erfolgreich aufgebaut wurde, so liefert die Funktion eine Verbindungs-Kennung zurück
    - Beispiel: $db = mysql_connect(„localhost“, „max“, „secret“);

- Verbindung abbauen
  - **mysql_close()**
  - schließt eine bestehende Verbindung zum MySQL-Datenbank-Server. In den Klammern muss die Verbindungskennung angegeben werden
    - Beispiel: mysql_close($db)

PHP und die MySQL-Datenbank

- Select-Abfrage (1)
  - **mysql_db_query()**
  - führt eine Abfrage mit *select* in der Datenbank aus. Der Aufbau der Abfrage entspricht der betreffenden SQL-Anweisung, allerdings sollte kein Semikolon am Ende gesetzt werden. Neben der SQL-Anweisung muss der Name der Datenbank angegeben werden.
  - falls die Abfrage erfolgreich war, so liefert die Funktion eine Ergebnis-Kennung zurück. Die Ergebnis-Kennung wird anschließend benötigt, um das Ergebnis zu untersuchen.
    - Beispiel: $res = mysql_db_query("firma", "select * from personen");

  - **mysql_num_rows()**
  - liefert die Anzahl der Datensätze zurück, die mit der Abfrage ermittelt wurde. Dabei wird die Abfrage eindeutig über die zuvor ermittelte Ergebnis-Kennung zugeordnet.
    - Beispiel: $num = mysql_num_rows($res);

PHP und die MySQL-Datenbank

- Select-Abfrage (2)
  - **mysql_result()**
  - Dient zur Ermittlung des Inhalts eines Feldes (name, vorname,...). In den Klammern muss der Verweis auf die Ergebnismenge, die gewünschte Zeile (Datensatz) sowie der Feldname angegeben werden. Die Namen der Felder müssen genau denen der Tabelle entsprechen.
    - Beispiel:

```
$nn = mysql_result($res, $i, "name");
$vn = mysql_result($res, $i, "vorname");
```

PHP und die MySQL-Datenbank

- Select-Abfrage (3) - Beispiel

```
<?php
  $db = mysql_connect(„localhost“, „max“, „secret“);
  $sqlab  = "select name, gehalt from personen";
  $sqlab .= " where gehalt >= 3000";
  $sqlab .= " and gehalt <= 3700";
  $sqlab .= " order by gehalt desc";
  $res = mysql_db_query("firma", $sqlab);
  $num = mysql_num_rows($res);
  echo "$num Datensätze gefunden<br>";
  for ($i=0; $i<$num; $i++) {
    $nn = mysql_result($res, $i, "name");
    $ge = mysql_result($res, $i, "gehalt");
    echo "$nn, $ge <br>";
  }
  mysql_close($db);
?>
```

PHP und die MySQL-Datenbank

- Datensätze erzeugen
  - **mysql_db_query()**
  - führt eine *insert* in der Datenbank aus. Der Aufbau entspricht der betreffenden SQL-Anweisung, allerdings sollte kein Semikolon am Ende gesetzt werden. Neben der SQL-Anweisung muss der Name der Datenbank angegeben werden.
    - Beispiel: mysql_db_query("firma", “insert into person (name, vorname, gehalt) values (‚$nn‘, ‚$vn‘, $gh)“;
  - **mysql_affected_rows()**
  - kann bei Aktions-Abfragen eingesetzt werden. Die Funktion ermittelt die Anzahl der von der Aktion betroffenen (=affected) Datensätze.
    - Beispiel: $num = mysql_affected_rows($res);
- Datensätze ändern / löschen
  - Analog zu „Datensätze erzeugen“

Zugriff auf Textdateien

- Öffnen einer Datei
  - **fopen()**
  - der erste Parameter gibt den Namen der Datei an, der zweite Parameter den Öffnungsmodus
    - r → read
    - w→ write
    - a → append
  - Rückgabewert der Funktion ist ein so genannter „Dateizeiger“. Dieser Dateizeiger wird für weitere Zugriffe auf die Datei benötigt. Sollte die Datei am angegebenen Ort nicht existieren, so gibt die Funktion den Wert *false* zurück.
    - Beispiel: $fp = fopen(„test.txt","r");

- Schließen einer Datei
  - **fclose()**
  - Der Parameter gibt an, welche Datei geschlossen werden soll. Dabei muss es sich um den Dateizeiger einer zuvor geöffneten Datei handeln.
    - Beispiel: fclose($fp)

Zugriff auf Textdateien

- Lesen von Zeichen
  - **fgets()**
  - dient zum Lesen einer Zeichenkette aus einer Datei
  - Der erste Parameter gibt an, aus welcher Datei gelesen werden soll. Dabei muss es sich um den Dateizeiger einer zuvor geöffneten Datei handeln. Der zweite Parameter gibt die Leselänge an. Es werden entweder (Leselänge – 1) Zeichen aus der Datei gelesen (hier 99) oder bis zum Zeilenumbruch oder bis zum Ende der Datei. Dies gilt je nachdem, was zuerst eintritt. Man sollte zum Lesen ganzer Zeilen eine Leselänge wählen, die auf jeden Fall für die betreffende Datei ausreicht.
  - Rückgabewert der Funktion ist die gelesene Zeichenkette (einschließlich des Zeilenumbruches).
    - Beispiel: $zeile = fgets($fp, 100);

Zugriff auf Textdateien

- Lesen aller Zeilen einer Textdatei
  - **feof()**
  - dient dazu, das Ende einer Datei anzuzeigen
  - der Parameter gibt an, welche Datei geprüft werden soll. Dabei muss es sich um den Dateizeiger einer zuvor geöffneten Datei handeln.
  - Rückgabewert der Funktion ist *true*, falls das Ende der Datei festgestellt wurde, bzw. *false* wenn dies nicht der Fall ist.
    - Beispiel: feof($fp)

```
<?php
  $fp = fopen(„test.txt","r");
  if ($fp){
    while (!feof($fp)){
      $zeile = fgets($fp, 100);
      echo "Zeile: $zeile<p>";
    }
    fclose($fp);
  } else {
     echo "Datei wurde nicht gefunden";
  }
?>
```

Zugriff auf Textdateien

- Lesen einer gesamten Datei
  - **file_get_contents()**
  - dient dazu, eine gesamte Datei auszulesen
  - der Parameter gibt an, welche Datei gelesem werden soll. Ein zuvor erzeugter Dateizeiger ist dafür nicht notwendig
  - Rückgabewert der Funktion sind die gelesenen Daten, falls das Ende der Datei festgestellt wurde, bzw. *false* bei Fehlern
    - Beispiel:
      - file_get_contents(“test.txt“)
      - file_get_contents(“https://example.com/file“)

Zugriff auf Textdateien

- Überschreiben einer Textdatei
  - **fputs()**
  - dient zur Ausgabe von Zeichenketten in eine Datei
  - der erste Parameter gibt an, in welche Datei ausgegeben werden soll. Dabei muss es sich um den Dateizeiger einer zuvor geöffneten Datei handeln. Der zweite Parameter beinhaltet die auszugebende Zeichenkette.
    - Beispiel: fputs ($fp, „Peter Pan");
  - **flock()**
  - dient zum Sperren einer Datei
  - der erste Parameter gibt an, welche Datei gesperrt werden soll. Dabei muss es sich um den Dateizeiger einer zuvor geöffneten Datei handeln. Der zweite Parameter gibt die Art der gewünschten Operation an: 2 : exklusiv sperren, 3: Sperre aufheben
    - Beispiel:

```
flock($fp,2);
$nl = chr(13) . chr(10);
fputs ($fp, „Peter Pan$nl");
flock($fp,3);
```

Arrays

- Eindimensionalen numerischen Arrays
  - **array()**
    - Beispiel: $temp = array(21.5, 22.2, 25.8, 24.4, 22.7);
  - Mit Hilfe der Funktion *array()* wird die Variable *$temp* zu einem Array mit fünf Elementen. Diese Elemente sind automatisch durchnummeriert worden, beginnend bei 0.
  - Arrays können auch einfach durch die Zuweisung einzelner Elemente erzeugt oder vergrößert werden → *$temp[5] = 20.2;*
  - Zugriff auf einzelne Elemente:

```
for($i=0; $i<=6; $i = $i+1) {
  echo "<p>$temp[$i]</p>";
}
```

  - Sortierung: *sort($temp);* oder *rsort($temp);*

Arrays

- Eindimensionale assoziative Arrays
  - Die Elemente eines solchen Arrays werden nicht über eine laufende Nummer, sondern über eine Schlüssel-Bezeichnung (Key) identifiziert. Dadurch wird es möglich, die Array-Elemente eindeutigen Begriffen zuzuordnen und die Suche nach bestimmten Array-Elementen zu vereinfachen.

```
<?php
  $tp = array("Montag"=>22.4,"Dienstag"=>21.1,"Mittwoch"=>21.8);
  $tp["Donnerstag"] = 21.6;
  $tp["Freitag"] = 22.5;
  $tp["Samstag"] = 21.2;
  $tp["Sonntag"] = 23.6;
?>
```

Arrays

- Eindimensionale Arrays und Datenbanken
  - **mysql_fetch_array()**
  - holt aus dem Ergebnis einen Datensatz und legt ihn als assoziativen Array ab. Die Keys des assoziativen Arrays sind die Namen der Datenbank-Felder, die Values des assoziativen Arrays sind die Werte aus den Datenbank-Feldern.
  - Nach dem Aufruf der Funktion mysql_fetch_array() wird ein interner Zeiger auf den nächsten Datensatz gestellt. Folglich wird beim nächsten Aufruf der nächste Datensatz gelesen. Nach dem letzten Datensatz gibt die Funktion mysql_fetch_array() den Wert false zurück

```
<?php
 $db = mysql_connect();
 $res = mysql_db_query("firma","select * from personen");
 mysql_close($db);
 while($zeile = mysql_fetch_array($res)){
   echo $zeile["vorname"] . " ";
   echo $zeile["name"] . "<br/>";
 }
?>
```

Arrays

- Parameterübergabe aus Formularen
  - HTTP-GET-Variablen: $_GET
    - Dieses Array wurde mit PHP 4.1.0 eingeführt.
    - Ein assoziatives Array mit Variablen, das an das aktuelle Skript mit der GET-Methode übergeben wurde.
    - Es ist automatisch global in allen Gültigkeitsbereichen.
    - Beispiel: $zuname = $_GET[“name“];
  - HTTP-POST-Variablen: $_POST
    - Vergleiche $_GET

FH Salzburg

VO Web-Technologien

29.11.2023, Oliver Jung

Technik
Gesundheit
Medien