Benutzerdefinierte PHP-Funktionen für Datum und Zeit

Dr. Volker Thormählen, 11. Juli 2019

43 benutzerdefinierte PHP-Funktionen und Code-Schnipsel für die Arbeit mit Datumsangaben

```php
function TagImJahr_in_Datum($jahr,$tagesnummer){
   // Aufgabe: Tagesnummer eines Jahres in Datum umwandeln.
   // Algorithmus von Richard A. Stone, Tableless Data Conversion, CACM 398
   // Eingabe: Jahr (4-stellig, numerisch)
   //          Tagesnummer des Jahres (1, ..., 365 bzw. 366 bei Schaltjahr)
   // Ausgabe: Datum im Format Tag.Monat.Jahr
   // Aufruf:  $jahr=2006; $tagesnummr=100; echo TagImJahr_in_Datum($jahr,$tagesnummer);
   // Beispiel:TagImJahr_in_Datum(2006,100) liefert 10.04.2006
   $a = ($jahr % 4 == 0) ? 1 : 0;
   $h = ($tagesnummer > (59 + $a)) ? 1 : 0;
   $b = 0;
   if ($h == 1){
      $b = ($a == 1) ? 1 : 2;
   }
   $c = $tagesnummer + $b + 91;
   $monat = (int)($c / 30.55);
   $tag = $c - (int)(30.55 * $monat);
   $monat = $monat - 2;
   return date('d.m.Y', strtotime($jahr."-".$monat."-".$tag));
}
```

Listing 1: Tagesnummer eines Jahres in Kalenderdatum umwandeln

```php
function Tagesnummer_in_Datum($jahr,$tagesnummer){
   // Aufgabe: Tagesnummer eines Jahres in Datum umwandeln.
   // Eingabe: Jahr (4-stellig, numerisch)
   //          Tagesnummer des Jahres (1, ..., 365 bzw. 366 bei Schaltjahr)
   // Ausgabe: Datum mit dt. Datumsformat.
   // Beispiel: Tagesnummer_in_Datum(2006,100) liefert 10.04.2006
   $datum = date('d.m.Y', strtotime("January 1st ".$jahr." +".($tagesnummer-1)." days"));
   return $datum;
}
```

Listing 2: Tagesnummer eines Jahres in Kalenderdatum umwandeln mit 'strtotime'

```php
function Get_Day_Of_Year($date){
   // Aufgabe: Tagesnummer im Jahr ermitteln (1, ..., 365 bzw. 366 bei Schaltjahr)
   // Eingabe: Kalenderdatum, z. B. '30.11.2020'
   // Ausgabe: Tagesnummer im Jahr, Ergebnis: z. B. 335
   $CumDays = array(0,31,59,90,120,151,181,212,243,273,304,334);
   list($day,$month,$year) = explode('.',$date);
   $isLeapYr = date('L', strtotime($date));
   $DoY = $CumDays[$month - 1] + $day;
   if($isLeapYr and ($month > 2)) $DoY ++;
   return $DoY;
}
```

Listing 3: Tagesnummer im Jahr anhand eines Kalenderdatums ermitteln

```php
function schaltjahr($jahr){
   // Aufgabe: Schaltjahr ermitteln.
   // Jahreszahlen zwischen 1970 und 2099.
   // Eingabe: Jahr (4-stellig, numerisch).
   // Ausgabe: 1=>true oder 0=>false
   // Aufruf:  $jahr=2009; echo schaltjahr($jahr);
   $schaltjahr = date("L", mktime(0,0,0,1,1,$jahr));
   return $schaltjahr;
}
```

Listing 4: Schaltjahr ermitteln

```php
function zeitumstellungen($jahr){
   // Aufgabe: Daten der Zeitumstellung eines Jahres ermitteln.
   // Eingabe: Jahr (4-stellig, numerisch).
   // Ausgabe: Daten der Zeitumstellung eines Jahres.
   // Aufruf:  $jahr = 2019; echo zeitumstellungen($jahr);
   return " März (+1 Std) " . date('d.m.Y', strtotime("last sunday of March ".$jahr)) . ", " .
       " Oktober (-1 Std) " . date('d.m.Y', strtotime("last sunday of October ".$jahr));
}
```

Listing 5: Zeitumstellungen des Jahres 2019 ausgeben

```php
function advent4($jahr){
   // Aufgabe: Datum des vierten Advents eines Jahres ermitteln.
   // Eingabe: Jahr (4-stellig, numerisch).
   // Ausgabe: Datum des 4. Advent (wenn $jahr= 2019, dann 22.12.2019).
   // Aufruf:  $jahr=2019; echo advent4($jahr);
   return date("d.m.Y",strtotime("+4 sunday",mktime(0,0,0,11,27,$jahr)));
}
```

Listing 6: Datum des 4. Advent des Jahres 2019 ermitteln

```php
function feiertage3($jahr){
   // Aufgabe: 3 Feiertage berechnen in Abhängigkeit vom 4. Advent des Jahres.
   // Eingabe: Jahr (4-stellig, numerisch).
   // Ausgabe: Daten von 3 Feiertagen (deutsches Datumsformat)
   // Aufrufe: advent4($jahr);
   // Aufruf:  $jahr=2019; feiertage3($jahr);
   $advent4 = advent4($jahr);
   echo 'Ewigkeitssonntag: '.date('d.m.Y',strtotime("-28 day",strtotime($advent4))).'<br>';
   echo 'Buss- und Bettag: '.date('d.m.Y',strtotime("-32 day",strtoTime($advent4))).'<br>';
   echo 'Volkstrauertag: '  .date('d.m.Y',strtotime("-35 day",strtoTime($advent4))).'<br>';
}
```

Listing 7: Drei Feiertage ausgeben abhängig vom 4. Advent des Jahres 2019

```php
function Muttertag($jahr){
// Aufgabe: Datum des Muttertags eines bestimmten Jahres bestimmen.
// Eingabe: Jahr (4-stellig, numerisch).
// Ausgabe: Datum des Muttertags des übergebenen Jahres 2019: 12.05.2019.
// Hinweis: Muttertag ist der 2. Sonntag im Mai eines Jahres.
//          Wenn dieser auf den Pfingstsonntag fällt, dann wird der Muttertag
//          um 1 Woche verschoben (z. B. in den Jahren 1989 und 2008)
// Aufruf:  $jahr=2019; echo Muttertag($jahr);
   $mothersDay = date('d.m.Y',strtotime('second Sunday of May '.$jahr));
   $easter  = easter_date($jahr);                                  // Ostern
   $whitSunday = date('d.m.Y', strtotime('+49 day', $easter));     // Pfingsten
   if ($mothersDay == $whitSunday){
      $mothersDay = date('d.m.Y',strtotime('-7 day', strtotime($mothersDay)));
   }
   return $mothersDay;
}
```

Listing 8: Muttertag eines Jahres ermitteln

```php
function lebensalter($birthDate){
   // Ausgabe: Aktuelles Lebensalter in ganzen Jahren (hier: 78) o. Fehlermeldung.
   // Eingabe: Gültiges Geburtstagsdatum.
   // Aufrufe: is_valid_date();
   // Aufruf:  $birthDate=date("14.09.1940");echo lebensalter($birthDate);
   if (is_valid_date($birthDate)){ // Gültigkeitsprüfung des Datums
      $today = date('d.m.Y',strtotime('today'));
      list($thisD, $thisM, $thisY) = explode('.',$today);
      list($birthD,$birthM,$birthY)= explode('.',$birthDate);
      // Lebensalter in ganzen Jahren berechnen.
      $age = $thisY - $birthY;
      if (($thisM < $birthM) or ($thisM == $birthM and $thisD < $birthD)){
         $age--;
      }
      return $age;
   }else{
      return "Ungültiges Geburtsdatum!";
   }
}
```

Listing 9: Lebensalter ermitteln

```php
function date_span($date1,$date2){
   // Aufgabe: Zeitspane in Kalendertagen zwischen zwei Daten ermitteln.
   // Eingabe: Zwei Daten im amerik. Format 'Y-m-d'
   // Aufruf mit:
   //    $date1 = date('Y-m-d',strtotime('today'));      // 2019-06-21
   //    $date2 = date('Y-m-d',mktime(0,0,0,6,30,2019)); // 2019-06-30
   // Ausgabe: Zeitspanne in Kalendertagen (im Beispiel: 9)
   // echo Die Zeitspanne beträgt '.date_span($date1,$date2).' Kalendertage.';
   list($j1, $m1, $t1) = explode('-', $date1);
   list($j2, $m2, $t2) = explode('-', $date2);
   $jdn1 = gregoriantojd($m1, $t1, $j1);
   $jdn2 = gregoriantojd($m2, $t2, $j2);
   // Datumsdifferenz.
   $day_diff = $jdn2 - $jdn1;
   return $day_diff;
}
```

Listing 10: Zeitspanne in Kalendertagen zwischen zwei Daten errechnen

```php
function loop_through_date_range($begin,$end){
   // Aufgabe: Kalenderwochen zwischen zwei Daten zählen.
   // Eingabe: Startdatum: $begin = new DateTime('2012-08-05');
   //       Enddatum:   $end   = new DateTime('2012-09-30');
   // Aufruf:  loop_through_date_range($begin,$end);
   // Ausgabe: Zahl der Kalenderwochen im Datumsintervall, Ergebnis: 9
   // Hinweis: Ergebnis ohne Berücksichtigung des letzten Tages des Enddatums: 8
   $end = $end->modify('+1 day');         // Den letzten Tag des Enddatums einbeziehen.
   $duration   = new DateInterval('P1W'); // Jeweils einen Zeitraum von 1 Woche hinzufügen.
   $daterange  = new DatePeriod($begin, $duration ,$end);
   $week_cnt = 0;
   foreach($daterange as $date){
      echo $date->format("d.m.Y")."<br>";
      $week_cnt ++;
   }
   return $week_cnt;
}
```

Listing 11: Kalenderwochen zwischen zwei Daten zählen

```php
function time_span($evtDate,$today){
   // Aufgabe: Zeitspanne bis zu einem best. Ereignisdatum ermitteln.
   // Eingabe: Ereignisdatum,     $evtDate = date('14.09.2019');
   //          Heute (17.06.2019)   $today = date('d.m.Y', strtotime("now"));
   // Ausgabe: Zeitspanne in Kalendertagen.
   // Ergebnis: echo 'Noch '.time_span($evtDate,$today).' Kalendertage bis zum '.$evtDate;
   $dt=date_parse($evtDate);
   $evt_tm = mktime (0,0,0,$dt['month'],$dt['day'],$dt['year']);
   $day = 86400; // s/d = 60 s/min * 60 min/h * 24 h/d
   // Zeitspanne in Kalendertagen.
   $diff =(int)(($evt_tm - time()) / $day);
   return $diff;
}
```

Listing 12: Zeitspanne in Kalendertagen bis zu einem best. Ereignisdatum ermitteln

```php
function zinstage_30_360($startdatum,$enddatum){
   // Aufgabe:    Zinstage nach dt. kfm. Methode (30/360) ermitteln.
   // Eingabe:    Datumsbereich: Enddatum > Startdatum.
   // Syntax:     Jahr-Monat-Tag.
   // Rückgabe:   Ermittelte Zinstage.
   // Aufruf:     echo zinstage_30_360($startdatum,$enddatum);
   $tm1 = strtotime($startdatum);
   $tm2 = strtotime($enddatum);
   // Daten vergleichen.
   if ($tm2 > $tm1){
      // Daten zerlegen.
      list($j1,$m1,$t1) = explode("-",$startdatum);
      list($j2,$m2,$t2) = explode("-",$enddatum);
      // 1. Fall: Jahre UND Monate gleich.
      if($j1 == $j2 && $m1 == $m2){
         $zinstage = ($t2 - $t1);              // Tagesdifferenz
      // 2. Fall: Jahre gleich.
      }elseif($j1 == $j2){
         $zinstage  = (30 - $t1 < 0) ? 0 : (30 - $t1) ; // erster Monat
         $zinstage += $t2;                     // letzter Monat
         $zinstage += ($m2-$m1-1)*30;          // andere Monate
      // 3. Fall: Jahre unterschiedlich.
      }else{
         $zinstage  = (30 - $t1 < 0) ? 0 : (30 - $t1);  // erster Monat
         $zinstage += (12-$m1)*30;             // erstes Jahr
         $zinstage += $t2;                     // letzter Monat
         $zinstage += ($m2-1)*30;              // letztes Jahr
         $zinstage += ($j2-$j1-1)*360;         // andere Jahre
      }
   }else{
      $zinstage=0;
   }
   return $zinstage;
}
```

Listing 13: Anzahl der Zinstage zwischen zwei Daten nach dt. kfm. Zinsmethode ermitteln

```php
function is_valid_date($date_str){
   // Aufgabe: Datum mit 'checkdate' auf Gültigkeit prüfen.
   // Eingabe: Datum als Zeichenkette.
   // Ausgabe: TRUE oder FALSE
   $tm = strtotime($date_str);
   if (!is_numeric($tm)){
       return FALSE;
   }
   // Zeitstempel zerlegen.
   $month = date('n',$tm);
   $day   = date('d',$tm );
   $year  = date('Y',$tm );
   if (checkdate($month, $day, $year)){
       return TRUE;
   }
   return FALSE;
}
```

Listing 14: Datum mit 'checkdate' auf Gültigkeit prüfen

```php
function valid_date($date){
   // Aufgabe: Datum mit 'date_parse' und 'checkdate' auf Gültigkeit prüfen.
   // Eingabe: Datum als Zeichenkette, z. B.: $date = "14.09.1940";
   // Ausgabe: TRUE oder FALSE
   $dt = date_parse($date);
   if ($dt["error_count"]==0 && $dt["warning_count"]==0
         && checkdate($dt["month"],$dt["day"],$dt["year"])){
      return TRUE;
   }else{
      return FALSE;
   }
}
```

Listing 15: Datum mit 'date_parse' und 'checkdate' auf Gültigkeit prüfen

```
function GetFiscalMonthForDate($dateToTest, $fyStart){
   // Aufgabe: Geschäftsmonat anhand eines Kalenderdatums ermitteln.
   // Eingabe: Kalenderdatum: $dateToTest = new DateTime('30 June 2019');
   //          Starttag u. -Monat d. GJ: $fyStart='first day of july';
   // Ausgabe: Zugehöriger Geschäftsmonat (Ergebnis: 12)
   $Yr      = date_format($dateToTest,'Y');        // Kalenderjahr, 4-stellig, numerisch
   $Month   = date_format($dateToTest,'m');        // numerischer Kalendermonat
   $Day  = date_format($dateToTest,'d');           // numerischer Kalendertag
   $fyStartDate   = new DateTime($fyStart.' '.$Yr);// Startdatum d. Geschäftsjahres
   $fyMonthStart  = date_format($fyStartDate,'m'); // numerischer Startmonat d. GJ
   $fyDayStart    = date_format($fyStartDate,'d'); // numerischer Starttag d. GJ
   $Month = $Month - $fyMonthStart + 1;
   if($Day < $fyDayStart){
      $Month --;
   }
   if ($Month < 1){
      $Month = $Month + 12;
   }
   return $Month;
}
```

Listing 16: Geschäftsmonat anhand eines Kalenderdatums ermitteln

```
function GetFiscalYearForDate($dateToTest,$fyEnd){
   // Aufgabe: Geschäftsjahr anhand eines Kalenderdatum ermitteln.
   // Eingaben:
   //    $dateToTest- Kalenderdatum: date('09/14/2019');
   //    $fyEnd      - Monat u. Tag f. Ende d. Wirtschaftsjahrs: '06/30'
   // Ausgabe: $fy   - Zugehöriges Wirtschaftsjahr: 2020
   $date = strtotime($dateToTest);
   $year = strftime('%Y',$date);
   $fyEnddate = strtotime($fyEnd.'/'.$year);
   if($date < $fyEnddate){
      $fy = intval($year);
   }else{
      $fy = intval($year + 1);
   }
   return $fy;
}
```

Listing 17: Geschäftsjahr anhand eines Kalenderdatums ermitteln

```
function CalculateFiscalYear($dateToTest,$cutoffMonth){
   // Aufgabe: Start und Ende eines Geschäftsjahres berechnen.
   // Eingabe: Kalenderdatum innerhalb des Geschäftsjahres: $dateToTest = '01.07.2019';
   //          Endmonat des Geschäftsjahres: $cutoffMonth = 6;
   // Ausgabe: Array mit Start und Ende des betrefffenden Geschäftsjahres.
   //          $result = calculate_fiscal_year($date_str,$cutoffMonth);
   //          Das Kalenderdatum 01.07.2019 gehört zum GJ 2019/2020.
   $result  = array();
   $month   = intval(substr($dateToTest,3,2));
r  $fyTo = ($month > cutoffMonth)?date('Y',strtoTime($dateToTest))+1:date('Y',strtoTime($dateToTest));
   $fyFrom = $fyTo - 1;
   $result['start']  = $fyFrom;
   $result['end']    = $fyTo;
   return $result;
}
```

Listing 18: Start und Ende eines Geschäftsjahres anhand eines Kalenderdatums ermitteln

```
function fiscal_calendar_445($start){
/* Aufgabe: 4-4-5 Fiskalkalender erzeugen.
   Eingabe: Anfängliches Startdatum: $start = '31.12.2018';
   Aufruf:  benötigt Funktion convertDayNm();
   Ausgabe: 4-4-5 Fiskalkalender für 2018/2019
         1. Quartal
   4 Wochen von Mo, 31.12.2018 bis So, 27.01.2019
   4 Wochen von Mo, 28.01.2019 bis So, 24.02.2019
   5 Wochen von Mo, 25.02.2019 bis So, 31.03.2019
         2. Quartal
   4 Wochen von Mo, 01.04.2019 bis So, 28.04.2019
   4 Wochen von Mo, 29.04.2019 bis So, 26.05.2019
   5 Wochen von Mo, 27.05.2019 bis So, 30.06.2019
         3. Quartal
   4 Wochen von Mo, 01.07.2019 bis So, 28.07.2019
   4 Wochen von Mo, 29.07.2019 bis So, 25.08.2019
   5 Wochen von Mo, 26.08.2019 bis So, 29.09.2019
         4. Quartal
   4 Wochen von Mo, 30.09.2019 bis So, 27.10.2019
   4 Wochen von Mo, 28.10.2019 bis So, 24.11.2019
   5 Wochen von Mo, 25.11.2019 bis So, 29.12.2019
   (4 + 4 + 5) Wochen/Quartal * 4 Quartale/Jahr = 52 Wochen/Jahr
*/
   $weeks = array(4,4,5,4,4,5,4,4,5,4,4,5);
   $startDate= date('d.m.Y', strtotime($start));
   $dayShort = date('D',strtotime($startDate));
   $startDay = convertDayNm($dayShort);
   $qrt = 1;
   echo $qrt.'. Quartal <br>';
   foreach($weeks as $week) {
      $days = (7 * $week) - 1;
      $endDate  = date('d.m.Y',strtotime($startDate.'+'.$days.'days'));
      $dayShort = date('D',strtotime($endDate));
      $endDay   = convertDayNm($dayShort);

      echo $week.' Wochen von '.$startDay.', '.$startDate.
                        ' bis '.$endDay  .', '.$endDate.'<br>';
      $startDate= date('d.m.Y', strtotime($endDate.'+ 1 day'));
      $dayShort = date('D',strtotime($startDate));
      $startDay = convertDayNm($dayShort);
      if ($week == 5) {
         $qrt ++;
         if ($qrt < 5) {
            echo $qrt.'. Quartal <br>';
         }
      }
   }
   echo '(4 + 4 + 5) Wochen/Quartal * 4 Quartale/Jahr = 52 Wochen/Jahr <br>';
}
```

Listing 19: 4-4-5 Kalender für das Wirtschaftsjahr 2018/2019 erzeugen

```
function arbeitstage_im_monat($jahr,$monat){
   // Aufgabe: Arbeitstage eines Monats im Jahr ermitteln.
   // Eingabe: Jahr und Monat des Jahres, z. B. $jahr=2019, $monat=7
   // Ausgabe:  Arbeitstage (ohne Wochenfeiertage), Ergebnis: 23
   $anzahl = 0;
   for($i=1;$i<= date('t',mktime(0,0,0,$monat,1,$jahr));$i++){
      // 'N' => Numerischer Wochentag nach ISO-8601 (1=>Montag, ..., 7=>Sonntag)
      if (date('N', mktime(0, 0, 0, $monat, $i, $jahr)) < 6){
         $anzahl++;
      }
   }
   return $anzahl;
}
```

Listing 20: Arbeitstage im Monat eines Jahres ermitteln

```php
function quartal($monat){
   // Aufgabe: Quartal anhand eines Monats ermitteln.
   // Eingabe: Monat (1, ..., 12), z. B.: 9
   // Ausgabe: Quartal (1, ..., 4) Ergebnis: 3
   return (int)(($monat - 1) / 3) + 1;
}
```

Listing 21: Quartal anhand eines Monats ermitteln

```php
function count_calendar_days($today){
   // Aufgabe: Kalendertage zählen von best. Datum bis zum 31.12. des aktuellen Jahres.
   // Eingabe: Aktuelles Datum mit Jahr, Monat und Tag,
   //          z.B. $today = date('Y.m.d',strtotime('today')) => 2019.06.24
   // Ausgabe: Zahl der Kalendertage bis Jahresende. Ergebnis: 190 Kalendertage.
   list($jahr,$monat,$tag) = explode('.',$today);
   $day = 86400;  // Sekunden je Tag: 60 x 60 x 24
   // Differenz in Kalendetagen ermitteln
   return (int)((mktime(0,0,0,12,31,$jahr) - mktime(0,0,0,$monat,$tag,$jahr))/$day);
}
```

Listing 22: Kalendertage zählen von best. Datum bis Jahresende

```php
function arab2roman($jahr){
   // Aufgabe: Arabische Jahreszahl in römischer Schreibweise ausgeben.
   // Eingabe: Arabische Jahreszahl, z. B. 2019
   // Ausgabe: Römische Jahreszahl, Ergebnis: MMXIX
   $rom_zeichen =
array(1000=>"M",900=>"CM",500=>"D",400=>"CD",100=>"C",90=>"XC",50=>"L",40=>"XL",10=>"X",9=>"IX
",5=>"V",4=>"IV",1=>"I");
   $result = "";
   foreach ($rom_zeichen as $val=>$sign){
      $figure = floor($jahr/$val);
      if ($figure > 0){
         $result .= str_repeat($sign,$figure);
      }
      $jahr = $jahr % $val;
   }
   return $result;
}
```

Listing 23: Arabische Jahreszahl in römischer Schreibweise ausgeben

```php
function jahreszeit($tm){
   // Aufgabe: Jahreszeit anhand eines best. Datums ermitteln
   // Eingabe: Zeitstempel, z. B. $tm = mktime(0, 0, 0, 6, 19, 2019)
   // Ausgabe: Zugehörige Jahreszeit, Ergebnis: Frühling (21.3 - 20.6)
   $m = date("n", $tm);
   $d = date("j", $tm);
   if (($m < 3) || (($m == 3) && ($d < 21)) || (($m == 12) && ($d >= 22))){
      return "Winter (22.12 - 20.3)";
   }else if (($m < 6) || (($m == 6) && ($d < 21))){
      return "Frühling (21.3 - 20.6)";
   }else if (($m < 9) || (($m == 9) && ($d < 23))){
      return "Sommer (21.6 - 22.9)";
   }else{
      return "Herbst (23.9 - 21.12)";
   }
}
```

Listing 24: Jahreszeit anhand eines best. Kalenderdatums ermitteln

```
function dt_datum($tag,$monat,$jahr){
   // Aufgabe: Datum mit deutschem Datumsformat ausgeben.
   // Eingabe: Tag, Monat, Jahr, z. B. '19','06','2019'
   // Ausgabe: Datum in deutschem Datumsformat. Ergebnis: Mittwoch, 19. Juni 2019
   $tag     = intval($tag);
   $monat   = intval($monat);
   $jahr    = intval($jahr);
   $Dt      = getdate(mktime(0,0,0,$monat,$tag,$jahr));
   $tage    = array("Sonntag","Montag","Dienstag","Mittwoch","Donnerstag","Freitag",
              "Samstag");
   $monate  = array(" ","Januar","Februar","März","April","Mai","Juni","Juli","August",
              "September","Oktober","November","Dezember");
   return $tage[$Dt["wday"]] . ", " . $tag . ". " . $monate[$monat] . " " . $jahr;
}
```

Listing 25: Datum mit deutschem Datumsformat ausgeben

```
function dt_datumsformat($date){
   // Aufgabe: Kalenderdatum in deutsches Datumsformat umwandeln.
   // Eingabe: Kalenderdatum: $date = date('d.m.Y',strtotime('14.09.2019'));
   // Rückgabe:Deutschsprachige Datumsangabe. Ergebnis: Samstag, 14. September 2019
   // Hinweis: 'setlocale' auf Deutsch umschalten:
   setlocale(LC_TIME,'de_DE.utf8','German_Germany');
   date_default_timezone_set('Europe/Berlin');
   return strftime("%A, %d. %B %Y",strtotime($date));
}
```

Listing 26: Kalenderdatum mit 'setlocale' auf Deutsch zurückgeben

```
function julian_date($tag,$monat,$jahr){
   // Aufgabe: Julianisches Datum ausgeben.
   // Eingabe: z. B. 19 06 2019
   // Ausgabe: Julianisches Datum, Ergebnis: 6.6.2019
   $jd = jdtojulian(unixtojd(mktime(2,0,0,$monat,$tag,$jahr)));
   $jd = explode("/",$jd);
   return $jd[1].'.'. $jd[0].'.'.$jd[2];
}
```

Listing 27: Julianisches Datum ausgeben.

```
function StartAndEndOfWeek($jahr,$kw){
   // Aufgabe: Start- und Enddatum einer best. Kalenderwoche eines Jahres ermitteln.
   // Eingabe: Jahr und Kalenderwoche, z. B. 2019,52
   // Ausgabe: Erster und letzter Kalendertag einer best. Woche im Jahr.
   // Ergebnis: Erster Wochentag: 23.12.2019, Letzter Wochentag: 29.12.2019
   $dt = new DateTime();
   // Die eingebaute PHP-Funktion setISODate() benötigt 3 Argumente: Jahr, Woche u. Tag
   return "Erster Wochentag: ".$dt->setISODate($jahr, $kw, "1")->format('d.m.Y')
      ."; Letzter Wochentag: ".$dt->setISODate($jahr, $kw, "7")->format('d.m.Y');
}
```

Listing 28: Start- und Enddatum einer best. Kalenderwoche eines Jahres ermitteln.

```
function convertDayNm($shortDayNm) {
   // Aufgabe: Kürzel eines Wochentags umschlüsseln.
   // Eingabe: 3-stelliger, englischer Kurzname eines Wochentags (Mon bis Sun)
   // Ausgabe: 2-stelliger, deutscher Kurzname dieses Wochentags (Mo bis So)
   $translate = array('Mon'=>'Mo','Tue'=>'Di','Wed'=>'Mi','Thu'=>'Do','Fri'=>'Fr','Sat'=>'Sa','Sun'=>'So');
   return $translate[$shortDayNm];
}
```

Listing 29: Kurznamen eines Wochentags umschlüsseln

```php
function get_date_one_year_later($dt){
   // Aufgabe: Datum um 1 Jahr erhöhen.
   // Eingabe: '2019-12-31' ... das ist das Ergebnis von:
   //          $month='2019-12'; $dt=date('Y-m-d',strtotime('last day of'.$month));
   // Ausgabe: Datum 1 Jahr später. Ergebnis: 2020-12-31
   if(!empty($dt)){
      if(date("m", strtotime($dt))=="2" && date("d", strtotime($dt))=="29"){
         // Schaltjahr
         return date("Y-m-d",
            strtotime((date("Y", strtotime($dt))+1).
            date("-m-28", strtotime($dt)))
         );
      }else{
         // Gemeinjahr
         return date("Y-m-d",
            strtotime((date("Y", strtotime($dt))+1).
            date("-m-d", strtotime($dt)))
         );
      }
   }
   return false;
}
```

Listing 30: Datum um 1 Jahr erhöhen

```php
function add_days($date,$days){
   // Aufgabe: Best. Anzahl von Tagen zum aktuellen Datum addieren.
   // Eingabe: Aktuelles Datum, z. B.: '24.06.2019'
   //          => $date = date('d.m.Y',strtotime('today'));
   //          Zahl der zu addierenden Tage, z.B.: 3
   // Ausgabe: Erhöhtes Datum, Ergebnis: '27.06.2019'
   $date = date('d.m.Y',strtotime($date.' +'.$days.' days'));
   return $date;
}
```

Listing 31: Aktuelles Datum um X-Tage erhöhen

```php
function getDatesFromDateRange($start,$end,$format='Y-m-d') {
   // Aufgabe: Daten eines Datumsbereichs in Array speichern.
   // Eingabe: Start- und Enddatum d. Datumsbereichs sowie Datumsformat.
   // Aufruf:  $arr = getDatesFromDateRange('2010-10-01','2010-10-05');
   // Ausgabe: foreach ($arr as $key => $val) {
   //             echo '$key: $val<br>';
   //          }
   $array = array(); // Array initiieren
   $interval = new DateInterval('P1D');
   $realEnd  = new DateTime($end);
   $realEnd->add($interval);
   $period = new DatePeriod(new DateTime($start), $interval, $realEnd);
   foreach($period as $date){
      $array[] = $date->format($format);
   }
   return $array;
}
```

Listing 32: Daten eines Datumsbereichs in Array speichern

```php
function loop_throuh_daterange($start,$end){
   // Aufgabe: Definierten Datumsbereich tageweise durchlaufen.
   // Eingabe: Datumsbereich: von '01.01.2019' bis '31.01.2019'
   // Ausgabe: Schleifenzähler. Ergebnis: 31
   $loopCnt=0;
   while (strtotime($start) <= strtotime($end)){
      echo $start.'<br>';
      // Schleifenzähler erhöhen.
      $loopCnt++;
      // weitere PHP-Anweisungen im Schleifenkörper.
      // Startdatum erhöhen.
      $start = date ('d.m.Y', strtotime('+1 day', strtotime($start)));
   }
   return $loopCnt;
}
```

Listing 33: Definierten Datumsbereich tageweise durchlaufen

```
function get_week_ranges($start,$end){
   // Aufgabe: Definierten Datumsbereich wochenweise durchlaufen.
   // Eingabe: Datumsbereich
   //    von:  $start = new DateTime('31.12.2018');   Startdatum
   //    bis:  $end   = new DateTime('30.12.2019');   Enddatum
   // Aufruf:  function quartal($monat) - Quartal anhand eines Monats ermitteln.
   // Ausgabe: Quartal, Wochennummer sowie Start- und Enddatum der jeweiligen Woche.
   //    z.B.: Quartal 4, Woche 01: von 31.12.2018 bis 07.01.2019
   //          ...
   //          Quartal 4, Woche 52: von 23.12.2019 bis 30.12.2019
   while($start < $end){
      $qrt = quartal($start->format('m'));
      echo 'Quartal '. $qrt . ', Woche '.$start->format('W').': von '.$start->format('d.m.Y')
      .' bis ';
      // Startdatum erhöhen.
      $start->modify('+1 week');
      echo $start->format('d.m.Y').'<br>';
   }
}
```

Listing 34: Definierten Datumsbereich wochenweise durchlaufen

```
function count_workdays($start,$end,$wochenfeiertage){
   // Aufgabe: Zahl der Arbeitstage im April 2919 ermitteln.
   // Eingabe: Start- u. Enddatum sowie Datenbereich mit 2 Wochenfeiertagen
   //    $start     = new DateTime('2019-04-01' );          Startdatm
   //    $end    = new DateTime('2019-04-30' );             Enddatum
   //    $wochenfeiertage = array('2019-04-19','2013-04-22');Karfreitag u.Ostermontag
   // Ausgabe: Zahl der Arbeitstage im April 2019 bei sog. 5-Tage-Woche
   // Ergebnis: 20 Arbeitstage
   $interval = new DateInterval('P1D'); // Zeitspanne, wobei P=Periode, D=Tag
   $daterange= new DatePeriod($start,$interval,$end); // Datumsbereich definieren
   $workdays = 0;    // Arbeitstagezähler initiieren.
   foreach($daterange as $date){
      // 'N' => Numerischer Wochentag gemäß ISO-8601: (1=Mo,...,6=Sa,7=So)
      if($date->format('N') < 6 and !in_array($date->format('Y-m-d'),$wochenfeiertage)){
         $workdays ++;
      }
   }
   return $workdays;
}
```

Listing 35: Zahl der Arbeitstage im April 2019 ermitteln

```
function get_ISO_weeknumber($date_str){
   // Aufgabe: ISO-Wochenmummer eines Datums ermitteln.
   // Eingabe: $date_str = "2019-06-25";
   // Ausgabe: Wochennummer, Ergebnis: 26
   $jahr    = substr($date_str,0,4);
   $monat   = substr($date_str,5,2);
   $tag     = substr($date_str,8,2);
   if (!checkdate($monat,$tag,$jahr)){
      return "falsch";
   }
   // "W" => ISO-8601 Woche des Jahres (1 = Montag)
   return date("W", strtotime($date_str));
}
```

Listing 36: ISO-Wochennummer eines Kalenderdatums ermitteln

```
function get_ISO_WeeksInYear($year){
   // Aufgabe: Zahl der Kalenderwochen im Jahr nach ISO
   // Eingabe: Jahr, 4-stellig, numerisch     => 2020
   // Ausgabe: Zahl der Kalenderwochen im Jahr => 53
   $date = new DateTime;
   $date->setISODate($year, 53);
   return ($date->format("W") === "53" ? 53 : 52);
}
```

Listing 37: Zahl der Kalenderwochen im Jahr nach ISO ermitteln

```php
function business_days_from_date($days,$direction,$date=NULL){
   // Aufgabe: Best. Zahl von Arbeitstagen ab Datum vorwärts o. rückwärts zählen.
   // Eingabe: $days:   Zahl d. Arbeitstage, zum Vorwärts- o. Rückwärtszählen.
   //       $direction: 1=>vorwärts, 0=>rückwärts.
   //       $date:      Datum (falls nicht angegeben, das Tagesdatum).
   // Ausgabe: Datum mit Format 'd.m.Y'.
   if(!$date){
      $date = date('d.m.Y'); // Datum ist das Tagesdatum.
   }
   while ($days > 0){
      $direction==1 ? $tm=strtotime($date.' +1 day') : $tm=strtotime($date.' -1 day');
      $date = date('d.m.Y', $tm);
      // 'N' => Numerischer Wochentag nach ISO-8601 (1=>Montag, ..., 7=>Sonntag)
      if(date('N', strtotime($date)) < 6){ // ... ist ein Arbeitstag
         $days--;
      }
   }
   return $date;
}
```

Listing 38: Bestimmte Zahl von Arbeitstagen ab Datum vorwärts o. rückwärts zählen

```php
function getStarSignFromDate($date){
   // Aufgabe: Sternzeichem eines Datums ermitteln.
   // Eingabe: Datum, entweder als  Zeichenkette o. DateTime-Objekt.
   // Ausgabe: Sternzeichen als Text.
   if (is_string($date)) $date = new DateTime($date);
   if (!$date instanceof DateTime) return false;
   $MonthDay = $date->format('md'); // string(4)
   // Assoziativen Datenbereich einrichten:
   $signs = array(
      '0120'=>'Steinbock','0219'=>'Wassermann','0320'=>'Fische',
      '0420'=>'Widder',   '0520'=>'Stier',     '0621'=>'Zwillinge',
      '0722'=>'Krebs',    '0823'=>'Löwe',      '0923'=>'Jungfrau',
      '1023'=>'Waage',    '1122'=>'Skorpion',  '1221'=>'Schütze',
      '1231'=>'Steinbock');
   foreach ($signs as $key=>$val){
      if ($MonthDay <= $key) return $val;
   }
   return false;
}
```

Listing 39: Sternzeichen eines Datums ermitteln

```php
function StartOfWeek($Date){
   // Aufgabe: Startdatum der Woche ermitteln, in die ein best. Kalenderdatum fällt.
   // Eingabe: Kalenderdatum: z. B. $Date  = date('d.m.Y', strtotime ('04.01.2019'));
   // Ausgabe: Startdatum der entsprechenden Kalenderwoche: 31.12.2018 => Montag
   return date("d.m.Y", strtotime('monday this week', strtotime($Date)));
}
```

Listing 40: Startdatum einer Woche ermitteln, in die ein best. Kalenderdatum fällt.

```php
function is_week_holiday ($datum,$land=null){
   // Aufgabe: Wochenfeiertag nach Datum und Bundesland ermitteln.
   // Eingabe: Kalenderdatum und Kürzel für das betreffende Bundesland: BW, BY, ..., TH
   // Ausgabe: Wochenfeiertag oder anderer Status ('Wochenende' bzw.'Arbeitstag')
   // 1. Prüfen,ob das übergebene Datum skalar ist.
   if(!is_scalar($datum)){
      echo "Datum ist nicht skalar";
      return false;
   }
   // 2. Datum prüfen.
   if(count(explode('-', $datum)) != 3){
      echo "Datum ist nicht gültig";
      return false;
   }else{
      // Übergebenes Datum aufteilen in Jahr, Monat und Tag.
      list($jahr,$monat,$tag) = explode('-',$datum);
   }
   // 3. Das Jahr muss 4-stellig sein.
   if(!strlen($jahr) == 4){
      echo "Jahr ist nicht 4-stellig";
      return false;
   }
   // 4. Monat bzw. Tag ggf. linksseitig mit '0' auffüllen.
   if(strlen($monat) < 2){
      $monat = str_pad($monat,2,"0",STR_PAD_LEFT);
   }
   if(strlen($tag) < 2){
      $tag = str_pad($tag,2,"0",STR_PAD_LEFT);
   }
   // 5. Übergebenes Datum mit der PHP-Funktion 'checkdate' auf Gültigkeit prüfen.
   if(!checkdate(intval($monat),intval($tag),intval($jahr))){
      echo "Übergebenes Datum ist falsch";
      return false;
   }
   // 6. Ostersonntag und -monat bestimmen.
   $ostern  = easter_date($jahr);
   $osterSo    = date("d",$ostern); // Ostertag
   $osterMnth  = date("m",$ostern); // Ostermonat
   // 7. Datums- und Zeitinformationen mit der PHP-Funktion 'getdate' ermitteln.
   $datum_arr = getdate(mktime(0,0,0,$monat,$tag,$jahr));
   // 8. Tag der Woche und zugehörigen Status bestimmen.
   $status =
   ($datum arr['wday']==0||$datum arr['wday']==6)?'Wochenende':'Arbeitstag';
   // 6. Monat und Tag verketten und prüfen.
   $md = $monat.$tag;
   if (strlen($md) != 4){
      echo "Konkatenierung Monat.Tag misslungen";
      return false;
   }
   // 7. Ermitteln, ob das verkettete Datum auf einen Wochenfeiertag fällt.
   if ($md=='0101'){
      return 'Neujahr';
   }elseif ($md=='0106' && ($land=='BW'||$land=='BY'||$land=='ST')){
      return 'Heilige Drei Könige';
   }elseif ($md=='0308' && $land == "BE"){
      return 'Frauentag';
   }elseif ($md==date("md",mktime(0,0,0,$osterMnth,$osterSo-2,$jahr))){
      return 'Karfreitag';
   }elseif ($md==date("md",mktime(0,0,0,$osterMnth,$osterSo+1,$jahr))){
      return 'Ostermontag';
   }elseif ($md=='0501'){
      return 'Tag der Arbeit';
   }elseif ($md==date("md",mktime(0,0,0,$osterMnth,$osterSo+39,$jahr))){
      return 'Christi Himmelfahrt';
   }elseif ($md==date("md",mktime(0,0,0,$osterMnth,$osterSo+50,$jahr))){
      return 'Pfingstmontag';
   }elseif ($md==date("md",mktime(0,0,0,$osterMnth,$osterSo+60,$jahr))
      && ($land=='BW'||$land=='BY'||$land=='HE'||$land=='NW'||$land=='RP'||$land=='SL')){
      return 'Fronleichnam';
   }elseif ($md=='0808' && $land=='BY'){
      return 'Augsburger Friedensfest';
   }elseif ($md=='0920' && $land=='TH'){
      return 'Weltkindertag';
   }elseif ($md=='0815' && ($land=='BY'||$land=='SL')){
      return 'Mariä Himmelfahrt';
   }elseif ($md=='1003'){
      return 'Tag der deutschen Einheit';
```

```
   }elseif ($md=='1031' && ($land=='BB'||$land='HB'||$land='HH'||$land=='MV'||$land=='NI'
                    ||$land=='SH'||$land=='SN'||$land=='SH'||$land=='TH')){
      return 'Reformationstag';
   }elseif ($md=='1101' && ($land=='BW'||$land=='BY'||$land=='NW'||$land=='RP'||$land=='SL')){
      return 'Allerheiligen';
   }elseif ($md==date('md',(strtotime("-11 days",strtotime("1 sunday",
         mktime(0,0,0,11,26,$jahr))))) && $land=='SN'){
      return 'Buß- und Bettag';
   }elseif ($md=='1225'){
      return '1. Weihnachtstag';
   }elseif ($md=='1226'){
      return '2. Weihnachtstag';
   }else{
      return $status;
   }
}
```

Listing 41: Wochenfeiertag ermitteln und zurückgeben

```
function GermanDateToMySQL($date){
   // Aufgabe: Deutsches Kalenderdatum umwandeln in ein MySQL-Datumsformat.
   // Eingabe: Kalenderdatum:  $date = date('d.m.Y', strtotime('14.09.2019'));
   // Ausgabe: Datum mit MySQL kompatiblem Datumsformat. Ergebnis: 2019-09-14
   $dt = explode('.',$date); // Datum zerlegen in Jahr, Monat u. Tag
   return sprintf("%04d-%02d-%02d", $dt[2], $dt[1], $dt[0]);
}
```

Listing 42: Deutsches Kalenderdatum umwandeln in ein MySQL-Datumsformat

```
function MySQLDateToGerman($date){
   // Aufgabe: MySQL-Datum in ein deutsches Kalenderdatum umwandeln.
   // Eingabe: MySQL-Datum: date('Y-m-d', strtotime('2019-09-14'));
   // Ausgabe: Deutsches Kalenderdatum . Ergebnis: 14.09.2019
   $dt = explode('-',$date); // Datum zerlegen in Tag, Monat u. Jahr
   return sprintf("%02d.%02d.%04d", $dt[2], $dt[1], $dt[0]);
}
```

Listing 43: MySQL-Datum umwandeln in ein deutsches Kalenderdatum

Eingebaute PHP-Funktionen für Datumsabgaben

Zeitstempel erzeugen mit ...	**strtotime('...')**
1 Monat addieren	$today.'+ 1 month'
14. September 1940	14th september 1940
14. September 1940	september 14, 1940
3 Jahre subtrahieren	$today.'- 3 years'
3 Kalendertage subtrahieren	$today.'- 3 days'
7 Kalendertage addieren	$today.'+ 7 days'
Erster Mittwoch im Dez. 2015	first wednesday 2015-12
Erster Montag im Folgemonat	first monday of next month
Erster Montag im Jahr 2019	first monday 2019-01
Erster Tag im Folgemonat	first day of next month
Letzter Freitag	last friday
Letzter Freitag im aktuellen Monat	last friday of this month
Letzter Montag im Jahr 2019	last monday 2019-12-31
Letzter Tag im Vormonat	last day of last month
Letzter Tag in diesem Monats	last day of this month
Montag dieser Woche	monday this week
Nächster Arbeitstag	$today .' +1 weekday'
Nächster Arbeitstag nach $days Tagen	$today .' +' .$days .'weekday'
Nächster Donnerstag	next thursday
Samstag der Vorwoche	saturday last week
Schaltjahr	echo date('L', strtotime($today));
Tagesnummer im Jahr	'january 1st +'.($days-1).' days'
Wochennummer (1=>Mo.)	echo date('W', strtotime($today));
Zahl der Kalendertage eines Monats	echo date('t', strtotime($today));
Zweiter Freitag im Vormonat	second friday of last month