|  |  |
| --- | --- |
| **PHP** | **Allgemein** |
| Groß-/Kleinschreibung | PHP unterscheidet zwischen Groß- und Kleinbuchstaben. |

|  |  |  |
| --- | --- | --- |
|  | **PHP** | **Erläuterung** |
| **1** | <?php ... ?> | Begrenzungszeichen (Separator). Sie können überall in der PHP-Seite stehen und trennen HTML-Tags von PHP-Skript-Befehlen. |
| **2** | //  /\* ... \*/ | einzeiliger Kommentar, auch am Zeilenende  mehrzeiliger Kommentar |
| **3** | + - \* /  % | Rechenoperatoren: Plus, Minus, Mal, Geteilt  Modulo (Divisionsrest), Beispiel: $Rest = 20 % 3; $Rest hat den Inhalt 2, weil 20 / 3 = 6 Rest 2 ergibt |
| **4** | and  or  !  xor | Logische Operatoren:  logisches Und  logisches Oder  logisches Nicht  logisches Entweder-Oder |
| **5** | ==  <>, !=  <, <=  >, >= | Vergleichsoperatoren:  Gleichheit  Ungleichheit  kleiner, kleiner oder gleich  größer, größer oder gleich |
| **6** | Verkettung . | Aus mehreren Zeichenketten wird eine zusammengesetzte Zeichenkette gebildet.  Bsp: echo "Arrayelement ist:" . $woche["SO"]; |

|  |  |  |
| --- | --- | --- |
|  | **PHP-Funktionen** | **Erläuterung** |
| **7** | count($array) | Zählt die Anzahl der Elemente in einem array  Beispiel: count($woche) ergibt für ein array, das die Wochentage enthält, den Wert 7. |
| **8** | date("Formatierungsstring", ["Unix-Timestamp"]) | Formatierungsstring ist eine Zeichenkette, in der mit Schaltern (siehe AB Einführung PHP) bestimmt wird, wie welche Zeit- und Datumsinformationen ausgegeben werden.  Der Unix-Timestamp ist eine Zahl, die die Anzahl der Sekunden seit dem 1.1.1970 angibt. Er ist optional (kann weggelassen werden), dann wird die aktuelle Zeit des Servers verwendet. |
| **9** | echo "Text"; | Schreibt den Text in die Ausgabe, die der Server an den Browser weiterleitet:  Beispiel: echo “Hallo“ erzeugt die Ausgabe Hallo |
| **10** | end($array) | Gibt den Wert des letzten array-Elements zurück  Beispiel: end($woche) ergibt den Wert Samstag für das array der Wochentage. |
| **11** | empty(*Ausdruck*) | Gibt einen booleschen Wert zurück, der angibt, ob die Variable leer ist. |
| **12** | include "dateiname"; | Fügt eine .php Datei ein; Verhält sich wie hereinkopieren. |
| **13** | implode("Trennzeichen",  $array) | Zur Ausgabeaufbereitung von Array-Elementen.  Bsp: $fussballmannschaft=array("FC Bayern München","Hamburger SV“);  $komma\_getrennt = implode(", ", $fussballmannschaft);  Ergebnis: echo $komma\_getrennt liefert  FC Bayern München, Hamburger SV |
| **14** | is\_numeric(*Ausdruck*) | Gibt den booleschen Wert TRUE zurück, wenn ein Ausdruck als Zahl oder als ein numerischer String ( +0123.45e6) ausgewertet werden kann. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **15** | isset() | Prüft, ob eine Variable existiert.  Beispiel: isset($\_POST["cbBad"]) liefert true, wenn $\_POST["cbBad"] existiert und sonst false. |
| **16** | is\_string(*Ausdruck*) | Gibt den booleschen Wert TRUE zurück, wenn ein Ausdruck als String ausgewertet werden kann. |
| **17** | number\_format  (*Variable*, Stellenzahl,  Dezimalzeichen,  Tausenderzeichen) | Formatiert eine Zahl und gibt sie als String zurück.  Beispiel: number\_format($zahl, 2, ",", ".") liefert für $zahl=1234567.123 beim obigen Befehl das Ergebnis 1.234.567,12 |
| **18** | pow(*Basis, Exponent*) | Potenziert die Basis mit dem Exponenten Beispiel: pow(2, 3) ergibt 23 = 2 \* 2 \* 2 = 8 |
| **19** | round(*Ausdruck, Stellenzahl, mode)* | Rundet eine Fließkommazahl auf die angegebene Stellenzahl. |
| **20** | str\_replace(string1, string2, string3) | Ersetzt alle String1 durch den String2 im String3.  Beispiel: str\_replace(",", ".", $zahl) liefert für $zahl=23,56 den Wert 23.56 |

|  |  |  |
| --- | --- | --- |
|  | **PHP-Kontrollstrukturen** | **Erläuterung** |
| **21** | if (Bedingung)  { *Anweisungsblock*  } | Bedingte Verarbeitung:  Ist die Bedingung wahr, so werden die Befehle im Anwei­sungsblock ausgeführt. |
| **22** | if (Bedingung)  { *Anweisungsblock 1*  }  else  { *Anweisungsblock 2*  } | einfache Alternative:  Ist die Bedingung wahr, so wird Anweisungsblock1 ausgeführt, wenn sie nicht erfüllt ist, wird Anweisungsblock2 ausgeführt. |
| **23** | switch(*Variable*)  { case(*Bedingung1*):  *Anweisungsblock 1*  break;  case(*Bedingung2*):  *Anweisungsblock 2*  break;  ...  default:  *Anweisungsblock n*  } | mehrfache Alternative (Fallabfrage) |
| **24** | while(*Bedingung/Ausdruck*)  { *Anweisungsblock* } | Solange-Schleife:  Solange die Bedingung/der Ausdruck wahr ist, wird der Anweisungsblock wiederholt. |
| **25** | for (Init Schleifenvariable;   Bedingung; Veränderung)  { *Anweisungsblock* } | Zählerschleife:  Bei jedem Schleifendurchlauf wird Zählvariable verändert. Abbruch der Schleife, wenn Bedingung nicht mehr wahr ist. |
| **26** | foreach (array as $wert)  { *Anweisungsblock* } | Schleifenausgabe für Arrays (ohne Abbruchbedingung) |

|  |  |  |
| --- | --- | --- |
|  | **PHP-Variable** | **Erläuterung** |
| **27** | $variablenname | Der Variablenname beginnt immer mit einem $-Zeichen |
| **28** | $variablenname = *Ausdruck*; | **Wertzuweisung:** Der Wert des Ausdrucks rechts vom Gleichheitszeichen wird der Variablen links vom Gleichheitszeichen zugewiesen.  Beispiel: $liter = 14; |
| **29** | $\_POST | Enthält alle Daten aus *methode = "post"* (Datenübernahme) Beispiel:  $vorname = $\_POST["tfVorname"]; |
| **30** | Datentypen | Boolean (wahr/falsch)  Integer (ganzzahlig)  Fließkommazahl (float, real, double)  Zeichenkette (String) |
| **31** | Typkonvertierung | Ein bedeutender Unterschied zwischen PHP und Sprachen wie Java liegt darin, dass PHP nicht typisiert ist, d.h. dass Variablenwerte beliebige Datentypen enthalten können.  In Anführungszeichen " " oder ' ' eingeschlossene Zeichen werden als String interpretiert.  Eine Zahl ohne Punkt wird als Ganzzahl interpretiert.  Eine Zahl **mit Punkt** (nicht wie bei uns üblich mit Komma) wird als Fließkommazahl interpretiert, z.B. 1.53 nicht 1,53. |