

11 – Funktionen mit Übergabeparameter

Infoblatt



ChatGPT erklärt: Was sind Übergabeparameter?

Stell dir folgendes vor: Funktionen sind wie kleine Helfer, die Aufgaben für dich erledigen. Manchmal brauchen diese Helfer Informationen, um besser zu verstehen, was du von ihnen willst. Hier kommen die Übergabeparameter ins Spiel.

Die Übergabeparameter sind wie Nachrichten, die du den Helfern gibst. Wenn du zum Beispiel möchtest, dass ein Helfer die Summe von zwei Zahlen berechnet, gibst du ihm diese Zahlen als Übergabeparameter.

```
<?php
    function addieren($a, $b) {
        echo $a + $b;
    }
```

Braucht man das wirklich?

Wir schauen uns das mal in einem Beispiel an. Die folgende Funktion soll prüfen, ob das geheime Passwort „Marshmallow“ korrekt eingegeben wurde:

```
<?php
    $password = $_POST['tfPasswort'];
    passwordPruefen();

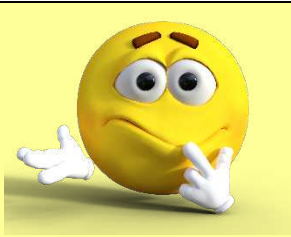
    function passwordPruefen()
    {
        if($password == "Marshmallow")
        {
            echo "Passwort korrekt :-)";
        }
    }

?>
```

Code – *Achtung, fehlerhaft!*

Probiere das mal aus – das vollständige Formular liegt unter *1-passwort-pruefen.html* vor, die Auswertungsdatei hat den Namen *1-passwort-pruefen-auswertung.php*.

Passwort eingeben:



Was ist denn hier los?

Wenn du das ausprobierst, wirst du feststellen, dass das nicht funktioniert! Du wirst eine Fehlermeldung erhalten – entweder schon in der Entwicklungsumgebung oder spätestens, wenn du das Skript ausprobierst.

Fehlermeldung in Visual Studio Code:

```
<?php
$password = $_POST["tfPassword"];

password // Undefined variable '$password'
@var unset $password
function // Funktion anzeigen (\F8) Keine Schne
{
    if ($password == "Marshmallow")
    {
        echo "Passwort korrekt :-)";
    }
}
```

Fehlermeldung im Browserfenster:

Warning: Undefined variable \$password in /Users/bnetz/my_documents/dateien/1-password/1-password-pruefen-auswertung.php on line 17

Das Problem liegt hier:

Eine Variable (in unserem Beispiel: `$password`) ist nur dort sichtbar, wo sie erstellt wurde.

Das heißt: Eine Funktion „sieht“ die Variablen nicht, die im Hauptprogramm erstellt wurden. Ebenso gilt: Das Hauptprogramm sieht die Variablen nicht, die in einer Funktion erstellt werden.

```
10 <?php
11
12 // Hauptprogramm
13 echo "<p>Programmstart!</p>";
14 $password = $_POST["tfPassword"];
15 passwordPruefen();
16
17 // Unterprogramm
18 function passwordPruefen()
19 {
20     if ($password == "Marshmallow")
21     {
22         echo "<p>Passwort korrekt :-)</p>";
23     }
24 }
25 ?>
```

Die Lösung: Übergabeparameter

Wenn eine Funktion nicht auf die Variable zugreifen kann, die wir außerhalb der Funktion erstellt haben, dann müssen wir der Funktion den Wert der Variablen eben „in die Hand drücken“. So etwas nennt man einen Übergabeparameter.

Grundsätzlicher Aufbau einer Funktion mit Übergabeparameter in PHP:

```
function nameDerFunktion(Übergabeparameter)
{
    // Hier passiert irgendwas;
}
```

Code

Und so sieht das dann aus:

```
12 // Hauptprogramm
13 1 $password = $_POST["tfPassword"];
14 2 passwordPruefen($password);
15
16
17
18
19
20 // Unterprogramm
21 function passwordPruefen($pPassword)
22 {
23     if ($pPassword == "Marshmallow") { 4
24         echo "<p>Passwort korrekt</p>";
25     } else {
26         echo "<p>ACCESS DENIED</p>";
27     }
28 }
```

Funktionsaufruf mit Parameter

Wert von **\$password** wird an **\$pPassword** übergeben

1 – Eine Variable **\$password** wird deklariert und mit dem Wert aus dem Formular initialisiert.

2 – Die Funktion **passwordPruefen (...)** wird aufgerufen. Sie bekommt einen Wert als Übergabeparameter.

Hinweis: Wir könnten auch direkt den Wert übergeben, etwa **passwordPruefen("123")**. Aber das würde natürlich nicht wirklich Sinn ergeben.

3 – Die Funktion ist so eingerichtet, dass sie erwartet, einen Wert zu bekommen. Diesen Wert wird sie als **\$pPassword** verwenden können.

Hinweis: **\$pPassword** wurde gewählt, um zu verdeutlichen, dass es sich um einen Parameter handelt. Natürlich könnte der Parameter auch anders heißen, z.B. **\$password** oder **\$meinPassword** oder **\$p**.

4 – Der Wert kann jetzt in der Funktion beliebig verwendet werden.