4.2 Fehlermeldungen

PHP ist äußerst restriktiv was seine Syntax betrifft. Wenn z. B. in HTML einmal ein unkritischer Tag nicht geschlossen wurde, parsed der Browser trotzdem den Code. PHP schickt sofort eine Fehlermeldung, selbst wenn nur ein Semikolon mit einem Beistrich verwechselt wurde (neben einer nicht geschlossenen Klammer, der meiste Fehler). In PHP gibt es zahlreiche Methoden um ein Debugging zu betreiben.

PHP

`error_reporting(E_ALL);`

Bestimmt das Verhalten, falls es zu einem Fehler oder einer Warnung kommt. Das Level der Meldungen (z. B. **`E_ERROR | E_WARNING`**) kann genauer bestimmt werden. In der DEV Phase der Programmierung sollten alle Fehler und Warnungen mit **`E_ALL`** aktiviert werden.
Der Code sollte noch vor dem **`<!doctype>`** Tag, ganz am Anfang stehen!

```
<?php
  error_reporting(E_ALL);
  ini_set('display_errors', 1);
?>

<!doctype html>
  <html lang="de"> ...
```

ini_set('display_errors', 1); sorgt dafür, dass alle Fehler und Warnungen in der Webseite dargestellt werden! Es setzt einen Eintrag in die php.ini

PHP

`error_reporting(0);`

In der Produktion sollte man alle Fehlermeldungen abschalten, da diese Hackern Informationen zum PHP Code geben (z. B. bei einer Division durch 0)!

```
<?php
  //Fehlermeldungen ausschalten
  error_reporting(0);
  ini_set('display_errors', 0);
?>
```

Nur weil das Error-Reporting deaktiviert wurde, bedeutet noch lange nicht, dass auch die Fehler verschwunden sind. So kann ein Syntax-Fehler noch immer zu einem Parse_Error führen.

Um einen Denkfehler im Code zu finden, bietet sich die gute alte Methode des Auskommentieren an. Zwei Schrägstriche erzeugen ein einzeiliges Kommentar. Mehrzeilige Kommentare werden mit einem Schrägstrich und einem Stern eingeleitet und mit Stern und Schrägstrich wieder geschlossen!

```
// Kommentiert eine Zeile
/* Kommentiert
   mehrere Zeilen aus */
```