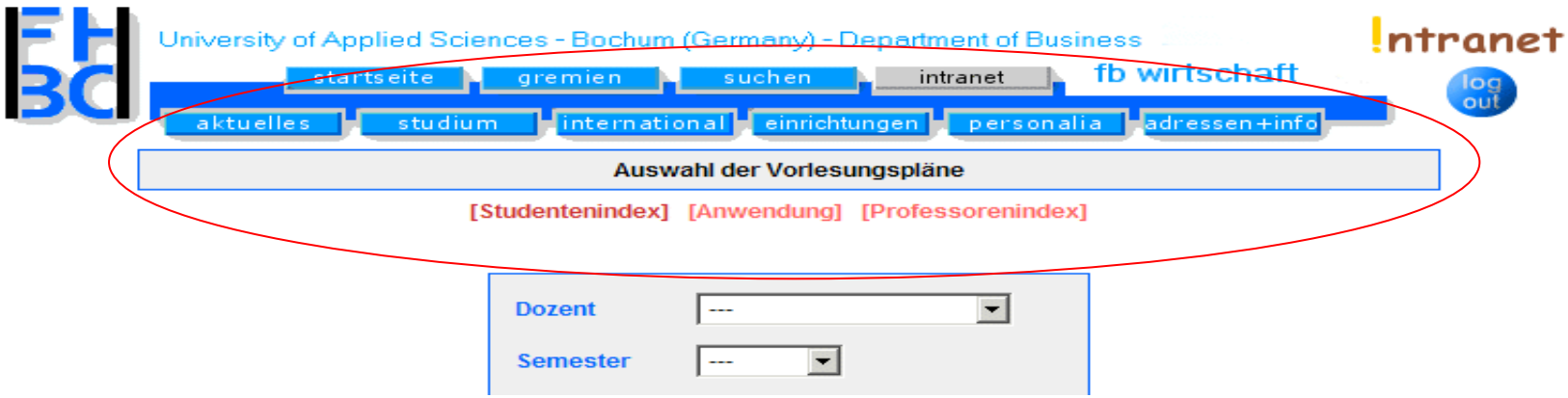
Funktionen

- sind kleine, meist ausgelagerte Programme bzw. Programmfragmente

- können Werte zurückgeben, z.B. Berechnungen

- Wert kann alles sein, was man so in Variablen speichern kann

Praktisches Beispiel:
Intranet



Funktionen

Quellcode:

```
<table border="0" align="center" width=620>
<tr>
	<td>
		<div class="liste3">
			Vorlesungsausf&auml;lle im Fachbereich
		</div>
	</td>
</tr>
</table>
```

Funktionen

```
function someFunction ($parameter1, $parameter2, $parameter3)
{
        anweisung1;
        anweisung2;
        anweisung3;
}
```

Funktionen

Man kann den Quellcode der Kopfzeile jetzt in jede Datei schreiben, muss es aber nicht
Lösung:

eine Funktion, welche einfach aufgerufen wird und dann den Quelltext in das Dokument einfügt

```
function darstellenAlsListe3($text)
{
?>
        <table border="0" align="center" width="620">
        <tr>
                <td>
                <div class="liste3">Vorlesungsausf&auml;lle im Fachbereich</div>
                </td>
        </tr>
        </table>
<?php
}
```

Funktionen - Aufruf

1.
darstellenAlsListe3();
Hier wird kein Parameter übergeben

2.
$text="Seitentitel des Intranets";
darstellenAlsListe3($text);
Hier wird $text als Parameter übergeben

3.
darstellenAlsListe3("Hier wird Text übergeben");
Hier wird der Text direkt als Parameter übergeben

Funktionen - Übung

```
<!-- Das Programm zur Funktionen-Demonstration
Dateiname: funktionen1.php //-->
<?php
function darstellenAlsListe3($text)
{
?>
        <table border='0' align='center' width=620>
        <tr>
        <td><div class='liste3'><?php echo $text ?></div>
        </tr>
        </table>
<?php
}
        darstellenAlsListe3("Der erste Text!");
        darstellenAlsListe3("Der zweite Text!");
        darstellenAlsListe3("Der dritte Text!");
?>
</body>
</html>
```

Funktionen – Übung auslagern

Aufgabe:

Lagern Sie die Funktion des vorherigen Beispiels in eine externe Datei namens darstellungsFunktionen.inc.php aus. Binden Sie anschliessend die Datei in Ihr Hauptdokument ein und rufen Sie von dort die Funktion auf.

Funktionen - Rückgabewert

```
function countChars($text)
{
        $summe=strlen($text);
        return $summe;
}
$text="Wunderschöner Donnerstag";
$summe=countChars($text);
echo $summe;
```

Obige Funktion gibt $summe als Wert zurück.

Funktionen – Call by Reference

```
<?php
function variablenAenderung($a)
{
        $a=2000;
}
$a=9;
variablenAenderung($a);
$b=21;
variablenAenderung($b);
echo ("Der Wert von \$a: $a <br>");
echo ("Der Wert von \$b: $b <br>");
?>
```

Hier passiert rein gar nichts.

Funktionen – Call by Reference

```
<?php
function variablenAenderung(&$a)
{
        $a=2000;
}
$a=9;
variablenAenderung($a);
$b=21;
variablenAenderung($b);
echo ("Der Wert von \$a: $a <br>");
echo ("Der Wert von \$b: $b <br>");
?>
```

Hier nimmt $a den Wert 2000 an.

Funktionen – JavaScript

Grundsätzlich gilt hier genau das gleiche wie für PHP-Funktionen mit einer Ausnahme:

In Java-Script können

- Variablen die Zahlen, Strings und Wahrheitswerte enthalten nicht per "Call by Reference" übermittelt werden
- Variablen die Arrays und Objekte enthalten werden immer per "Call by Reference" übergeben, sprich durch die JavaScript Funktionen geändert.

Deswegen bei Zahlen, Strings und Wahrheitswerten Return verwenden.

Funktionen – JavaScript

```
function macheInteger(wert)
{
        zahl=parseFloat(wert);
        return zahl;
}
```

Funktionen – JavaScript – Beispiel - head

```
<!-- Programm zur Nutzung von Rueckgabewerten
Dateiname: funktionen2.html //-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Provisionen 1</title>
<script language="JavaScript">
function berechneProvision(umsatzFunktion)
{
        var provisionFunktion;
        if(umsatzFunktion<=0)
        {
                return "Fehler";
        }
        if(umsatzFunktion<10000)
        {
                provisionFunktion=0;
        }
        else
        {
```

Funktionen – JavaScript – Beispiel - body

```
<body>
<h2> Demonstration von return </h2>
<script language="JavaScript">
var umsatz;
var provision;
umsatz=prompt("Bitte geben Sie den Umsatz ein!","");
umsatz=parseFloat(umsatz);
provision=berechneProvision(umsatz);
document.write("Die Provision zu " + umsatz +
" betr¨agt " + provision + " !")
</script>
</body>
</html>
```

Funktionen JS – EuroDollarUmrechnung

Beispiel 7.7 *Euro zum Dritten: Die ausgelagerte Berechnungsdatei*

```
<?php
// Funktion zur Dollar-Euro oder Euro-Dollar Umrechnung
// Datei:euroDollarUmrechnung.inc.php
// Verzeichnis: includes
function euroDollarUmrechnung($zielwaehrung, $betrag)
{
        $kurs=0.9;
        if(($zielwaehrung=="Dollar")||($zielwaehrung=="dollar"))
        {
                $dollarbetrag=$kurs*$betrag;
                return "$dollarbetrag Dollar";
        }
        if(($zielwaehrung=="Euro")||($zielwaehrung=="euro"))
        {
                $eurobetrag=(1/$kurs)*$betrag;
                return "$eurobetrag Euro";
        }
}
?>
```

Funktionen JS – EuroDollarUmrechnung

```
else
{
    if(($zielwaehrung=="Dollar")||($zielwaehrung=="dollar") ||
    ($zielwaehrung=="euro")||($zielwaehrung=="Euro"))
    {
        $ergebnis=euroDollarUmrechnung($zielwaehrung,$betrag);
        echo "Ihre Eingabe entspricht $ergebnis!";
    }
    else
    {
        echo("Falsche Zielw&auml;hrung: <br>" .
        "Erlaubt sind: Euro oder Dollar!");
    }
}
?>
</body>
</html>
```

Funktionen JS – EuroDollarUmrechnung

Beispiel 7.8 *Euro zum Dritten: Der Aufruf*

```
Dateiname: euro3.php //-->
<html>
<head><title> Euro-Dollar Umrechnung Teil 3</title>
<?php require_once("./includes/euroDollarUmrechnung.inc.php"); ?>
</head>
<body>
<?php
if($REQUEST_METHOD!="POST")
{
  echo "<form name='euro2' action='$PHP_SELF' method='post'>";
?>
```

Funktionen JS – EuroDollarUmrechnung

```
<table border>
<tr>
<td>Zielw&auml;hrung</td>
<td><input type="text" name="zielwaehrung" size=12></td>
</tr><tr>
<td>Betrag</td>
<td><input type="text" name="betrag" size=12></td>
</tr><tr>
<td colspan="2" align="center">
<input type="submit" name="Button1" value="Abschicken">
</td></tr>
</table>
</form>
<?php
}
```

Funktionen – Falscher Aufruf

```
<?php
function dividieren ($eins, $zwei)
{
        $ergebnis=$eins / $zwei;
        return $ergebnis;
}

$ersteZahl=20;
$zweiteZahl=2;
$resultat=dividieren($ersteZahl, $zweiteZahl);
echo "Das Ergebnis der ersten Division ist: $resultat<br>";

$resultat=dividieren($zweiteZahl, $ersteZahl);
echo "Das Ergebnis der zweiten Division ist: $resultat<br>";
?>
```

Funktionen – Rakete – ausgelagerte Berechnungsfunktion

```
<?php
// datumsfunktionen
// Datei:datumfunktionen2.inc.php
function tageInMonaten($monat, $tag)
{
        switch($monat)
        {
                case 1:
                $tag=$tag; //ueberflussig, nur der Klarheit weegen
                break;
                case 2:
                $tag=31+$tag;
                break;
                case 3:
                $tag=31+28+$tag;
                break;
                case 4:
                $tag=31+28+31+$tag;
                break;
```

Funktionen – Rakete – ausgelagerte Berechnungsfunktion

```
case 5:
$tag=31+28+31+30+$tag;
break;
case 6:
$tag=31+28+31+30+31+$tag;
break;
case 7:
$tag=31+28+31+30+31+30+$tag;
break;
case 8:
$tag=31+28+31+30+31+30+31+$tag;
break;
case 9:
$tag=31+28+31+30+31+30+31+31+$tag;
break;
case 10:
$tag=31+28+31+30+31+30+31+31+30+$tag;
break;
case 11:
```

Funktionen – Rakete – ausgelagerte Berechnungsfunktion

```
                $tag=31+28+31+30+31+30+31+31+30+31+$tag;
                break;
                case 12:
                $tag=31+28+31+30+31+30+31+31+30+31+30+$tag;
                break;
                } //Schliessen von switch
        return $tag;
} //schliessen der Funktion
```

Funktionen – Rakete – ausgelagerte Berechnungsfunktion

```
//Berechnet die Sekunden
function berechneSekunden($tag, $stunden, $minuten, $sekunden)
{
        return($tag*24*3600+$stunden*3600+$minuten*60+$sekunden);
}
```

Funktionen – Rakete – ausgelagerte Berechnungsfunktion

```
function tageStundenMinutenSekundenAusSekunden(&$tage,&$stunden, &$minuten, &$sekunden)
{
        // zuerst sekunden
        $minuten=floor($sekunden/60);
        $sekunden=$sekunden%60;
        //nun minuten und stunden
        $stunden=floor($minuten/60);
        $minuten=$minuten%60;
        //und tage
        $tage=floor($stunden/24);
        $stunden=$stunden%24;
}
?>
```

Funktionen – Rakete – Aufruf

```
if($flugzeitInSekunden<0)
    {
        echo "Fehleingabe: Landezeit vor Startzeit!";
    }
    else
    {
        //Flugzeit umrechnen
        tageStundenMinutenSekundenAusSekunden($flugzeitTage,
                    $flugzeitStunden, $flugzeitMinuten,
                    $flugzeitSekunden);
        //ausgeben
```

Funktionen – Zusammenfassung

Funktionen
sind ausgelagerte Programmteile
werden erst deklariert

```
function machwas();
function machwas($variable);
function machwas($variable; $foo="bar")
```

und dann aufgerufen

```
machwas();
machwas($variable);
machwas($variable, $foo);
```

Rückgabewerte

per return

```
$rueckGabeWertDerFunktion=machwas($variable);
```

per Call-by-Reference

```
function machwas(&$variable)
```