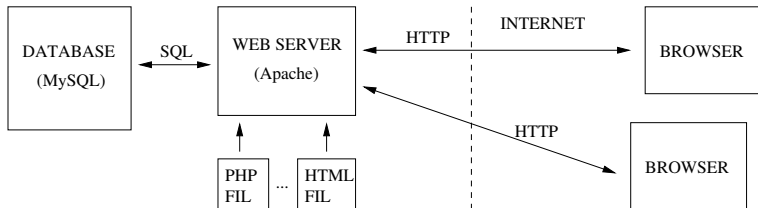


- ▶ Introduktion til webprogrammering
- ▶ Programmeringssproget PHP — det andet script
- ▶ Kodeindlejring og kommentarer
- ▶ Variabler og beregninger med tal
- ▶ Noget om strenge
- ▶ Betingelser — `if`-sætninger
- ▶ Løkker — `while`-løkker
- ▶ Brug af formvariabler
- ▶ PHP scripts på serveren

# Introduktion til Webprogrammering

## Oversigt:



- ▶ Klient (browser) forespørger en side på en web-server
- ▶ Web-server udfører program
- ▶ Web-server programmet tilgår eksempelvis en database på web-serveren
- ▶ Programmets resultat (HTML kode) sendes til klienten

HTTP er den protokol hvormed data sendes imellem klient og web-server

HTTP er fundamentet for "the World Wide Web"

PHP er et programmeringssprog, primært til webprogrammering

Eksempel — `hello.php`:

```
<html><head><title>Hello World</title></head>
  <body>
    <?php echo "Hi There, <p>";
      echo "Greetings from a simple PHP script</p>"; ?>
  </body>
</html>
```

- ▶ PHP kode indlejres i HTML kode ved brug af `<?php` og `?>`
- ▶ PHP kommandoen `echo` sender dets argument (en streng "...") til browseren
- ▶ PHP kommandoer afsluttes med `;`

Når en browser forespørger siden `hello.php` på serveren, da køres PHP koden, hvilket resulterer i HTML kode, som sendes til browseren:

```
<html><head><title>Hello World</title></head>
  <body>
    Hi There, <p>Greetings from a simple PHP script</p>
  </body>
</html>
```

- ▶ Indholdet der sendes tilbage er fortolket, **man kan ikke se PHP ved "view source"**

# Kodeindlejring og kommentarer

Der er et par måder man at indlejre PHP i HTML'en med: —  
indlejre.php:

```
<html><head><title>Indlejring</title></head>
  <body> <h2>Tre former for indlejring</h2>
    <ol>
      <? echo "<li>Den 'korte' form</li>"; ?>

      <?php echo "<li>Den RIGTIGE og 'halvlange' form</li>"; ?>

      <SCRIPT LANGUAGE="PHP">
        // Kommentarer i PHP kode sendes ikke til browseren
        echo "<li>Den 'lange' form</li>";
      </SCRIPT>

    </ol>
  </body>
</html>
```

Som udgangspunkt skal i **ALTID** benytte <?php ?>

- ▶ Kommentarer bruges til..
  - ▶ at gøre koden lettere at læse
  - ▶ at hjælpe andre med at forstå hvorfor
  - ▶ at hjælpe en selv med at forstå hvorfor
  - ▶ at hurtigt fjerne dele af koden mens man 'fejlretter'

Kommentarer over flere linier skrives `/* ... */` Enkelte linier udkommenteres med `#` eller `//`

```
<?php
/*Jeg bliver
aldrig læst af serveren*/

//jeg bliver aldrig læst

#jeg bliver aldrig læst
?>
```

# Variabler i PHP

- ▶ Variabler benyttes til opbevaring af værdier under kørsel af et PHP script
- ▶ Værdier kan bla. være strenge og tal
- ▶ Variabelnavne må indeholde tal, bogstaver og underscores (-)
- ▶ og skal starte med \$ efterfulgt af bogstaver eller \_

```
<h2>
  <?php $navn = "Jonas";
      echo "Hjemmeside for ";
      echo $navn;
  ?>
</h2>
<hr>
Denne side vedligeholdes af <?php echo $navn ?>
```

Bemærk at der kan henvises til en variabel flere gange efter tildeling af en værdi til variabelen

```
<h2>
```

```
    Hjemmeside for Jonas
```

```
</h2>
```

```
<hr>
```

```
Denne side vedligeholdes af Jonas
```

# Beregninger med tal

Der findes to typer af tal i PHP:

1. heltal (integers): 3, 9, 0, -1, ...
2. kommatotal (doubles): 3.0, 9.2, 0.13, -23.2, ...

Sædvanlige regneoperationer (f.eks. +, -, \*, og /) kan benyttes i beregninger. Eksempel — `dollars.php`:

```
<h2>Dollars</h2>
<?php
    $kurs = 8.43;
    $kroner = 1000.0;
    $dollars = ($kroner - 20.0) / $kurs;
    echo "For kr. $kroner modtager du $dollars dollar"; ?>
```

# Mere om strenge

- ▶ For at indsætte et dollar-tegn (\$) i en streng med "dobbelt anførelstegn" skal karakteren \ foranstilles dollar-tegnet
- ▶ Strenge i PHP kan indkapsles enten ved brug af "..." eller '...'
- ▶ I strenge af formen '...' kan man ikke henvise til variabler

```
<?php
$meaningOfLife=21+21;

echo "21+21 = $meaningOfLife";
//Udkriver 21+21 = 42

echo '21+21 = $meaningOfLife';
//Udkriver 21+21 = $meaningOfLife

echo "21+21 = \$meaningOfLife";
//Udkriver 21+21 = $meaningOfLife
?>
```

# Specielle karakterer - escaping

Andre *specialtegn* der kan benyttes i strenge med formen "...":

- ▶ \\ : backslash
- ▶ \" : anførselstegn
- ▶ \\$ : dollartegn

Specielle *specialtegn*

- ▶ \n : ny linie
- ▶ \t : tabulator

```
<?php
//BEMÆRK: fejl i første runde slides

echo "\\ \" \$";
//Udskriver \ $

echo '\\ \' \$';
//Udskriver \\ \' \$

?>
```

Strenge sammensættes ved brug af tegnet '.'

Eksempel — strenge.php

```
<html><head><title>Strenge</title></head>
  <body>
    <?php $course = "DSDS";
          $semester = "E2007";
          $description = $course . " i semesteret " . $semester;
          echo "Du er tilmeldt $description";
    ?>
  </body>
</html>
```

# Aritmetiske operatører og præcedens

Operator	Betydning	Eksempler	
*	Multiplikation	$1.5 * 60$	$24 * 60$
/	Division	$134.0 / 60$	$134 / 60$
%	Rest	—	$134 \% 60$
+	Addition	$1.1 + 60$	$14 + 60$
-	Subtraktion	$1.1 - 60$	$134 - 60$

- ▶ Operatører med høj præcedens binder stærkere end operatører med lav præcedens, og udregnes først.
- ▶ Operatørerne  $*$ ,  $/$  og  $\%$  har højere præcedens end  $+$  og  $-$  og udregnes altså før  $+$  og  $-$ .
- ▶ Når operatørerne har samme præcedens (binder lige stærkt) regnes fra venstre mod højre.

Vi kan lave vilkårligt komplicerede udtryk, f.eks.:

$$22 - 34 + 43 \% 34 * 23 + 122 / 43.22 * 23 + 43$$

- ▶ Uden præcedensregler kan udtryk være tvetydige: vil  $2+4*4$  give 24 eller 18?
- ▶ Eftersom  $*$  har højere præcedens end  $+$ , udregnes  $4*4$  først, og derefter lægges 2 til.
- ▶ Hvis vi vil lægge sammen først, skal vi benytte parenteser:  $(2+4)*4$  giver 24.
- ▶ Eftersom  $*$  og  $/$  har samme præcedens, regnes fra venstre mod højre:  $2*5/2$  giver 5.
- ▶ Vi kan dividere først ved at sætte en parentes:  $2*(5/2)$  giver 4.

# Eksempel: Hvilken type og værdi har følgende udtryk?

<b>Aritmetisk udtryk</b>	<b>Resultattype</b>	<b>Resultatværdi</b>
$1.5 * 60$		
$150.0 / 60$		
$150 / 60$		
$134 \% 60$		
$1.1 + 60$		
$1.1 - 60$		

# Boolske værdier (Booleans)

Boolske værdier er værdier der er enten sande eller falske (TRUE/FALSE)

PHP har et meget afslappet forhold til den slags:

- ▶ 1 er true
- ▶ 0 er false
- ▶ -2 er true
- ▶ 'a' er true
- ▶ true er true
- ▶ false er false
- ▶ 'false' er true!!!!

Eller... Næsten alt andet end false, 0 og '' (tom streng) er true.  
Boolske værdier bruger vi ofte til betingelser:

If-sætninger bruges til at udføre forskellig kode på baggrund af en *betingelse*

Eksempel — `kroner.php`:

```
<?php $kroner = 8;
    if ( $kroner == 1 ) { // if-gren
        echo "Du har 1 krone på din bankbog";
    } else { // else-gren
        echo "Du har $kroner kroner på din bankbog";
    }
?>
```

En betingelse er enten FALSK (værdien 0) eller SAND (forskellig fra 0)

Hvis betingelsen (`$kroner == 1`) er FALSK (værdien 0) så udføres `else-grenen`. Ellers udføres `if-grenen`

- == lig med
- < mindre end
- <= mindre end eller lig med
- > større end
- >= større end eller lig med
- != forskellig fra

Generelt format:

```
if ( betingelse1 ) {  
    kommandoer1  
} elseif ( betingelse2 ) {  
    kommandoer2  
} else {  
    kommandoer3  
}
```

Forløb:

1. beregn værdien af *betingelse1*.
2. hvis SAND så udfør *kommandoer1*, ellers
3. beregn værdien af *betingelse2*
4. hvis SAND så udfør *kommandoer2*, ellers
5. udfør *kommandoer3*

Man kan have et vilkårligt antal `elseif`-sekvenser

Det er muligt at undlade den afsluttende `else`-konstruktion

```
<h2>Body Mass Index</h2>
<?php
    $vaegt = 70.0;
    $hoejde = 180.0;
    echo "Vægt: $vaegt kg. <br>Højde: $hoejde cm.<br>";
    $bmi = $vaegt / (($hoejde/100.0) * ($hoejde/100.0));
    echo "Dit BMI er $bmi<br>";
    if ( $bmi < 20.0 ) {
        echo "Dit BMI er for lavt.";
    } elseif ( $bmi < 25.0 ) {
        echo "Dit BMI er normalt.";
    } else {
        echo "Dit BMI er for højt.";
    }
?>
```

# Sammenligningsoperatører og deres præcedens

Operator	Betydning	Eksempel
<	Mindre end	$\$x < 60$
<=	Mindre end eller lig med	$\$x \leq 60$
>	Større end	$\$x > 60$
>=	Større end eller lig med	$\$x \geq 60$
==	Lig med	$\$x == 60$
!=	Forskellig fra	$\$x != 60$

- ▶ De aritmetiske operatører  $*$ ,  $/$ ,  $\%$ ,  $+$ ,  $-$  binder alle stærkere end sammenligningsoperatørerne  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ .
- ▶ Sammenligningsoperatørerne  $<$ ,  $\leq$ ,  $>$ ,  $\geq$  binder alle stærkere end  $==$  og  $!=$ .
- ▶ Hvorfor bruges  $=$  **ikke** til sammenligning?

# Eksempel

Lad  $x$  have værdien 2 og  $y$  have værdien 4.

$x=2$ ;  $y=4$ ;

Logisk udtryk	Resultatværdi
$1 == 1$	
$x != y$	
$x < 3 + y$	
$(x + y > 3) == 0$	
$0 != x < 3$	
$x == y == 0$	
$x = y$	

Husk: Tallet 1 svarer til SAND og tallet 0 svarer til FALSK.

# Logiske operatører og deres præcedens

Operator	Betydning	Eksempel
!	Ikke (Negation)	!(\$x == 60)
&&	Og (Konjunktion)	0 <= \$x && \$x < 60
	Eller (Disjunktion)	\$x < 0    \$x >= 60

- ▶ Operatoren ! binder stærkere end && som binder stærkere end ||.
- ▶ ! binder også stærkere end sammenligningsoperatorerne og de aritmetiske operatører.
- ▶ && og || binder svagere end sammenligningsoperatorerne og de aritmetiske operatører.
- ▶ Der udregnes fra venstre mod højre.
- ▶ Hvis *udtryk1* er FALSK i *udtryk1 && udtryk2* så udregnes *udtryk2* ikke.
- ▶ Hvis *udtryk1* er SAND i *udtryk1 || udtryk2* så udregnes *udtryk2* ikke.

# Eksempel: Hvad er resultatet af følgende logiske udtryk?

`$x=2; $y=4;`

Boolsk udtryk	Resultat
<code>! 0</code>	
<code>! 1</code>	
<code>! 1 == 0</code>	
<code>! (1 == 0)</code>	
<code>1 &amp;&amp; 0</code>	
<code>0    1</code>	
<code>\$x + \$y &gt; 3 &amp;&amp; \$x &lt; \$y</code>	
<code>\$x + \$y == 3    \$x &lt; 4</code>	

Husk, at 1 svarer til SAND og 0 til FALSK.

Hvorledes kan vi udskrive "I love Web programming" 20 gange?

## Dårlig løsning:

```
echo "I love Web programming<br>";  
...18 gange...  
echo "I love Web programming<br>";
```

**Bedre løsning:** benyt en `while`-løkke til repetition — `love.php`:

```
$counter = 20;  
while ( $counter >= 1 ) {  
    echo "I love Web programming<br>";  
    $counter = $counter - 1;  
}
```

Kommandoen `echo` udføres 20 gange, hvor `counter = 20, 19, ..., 1`

Det er vigtigt at variabelen `counter` nedtælles i hvert gennemløb af løkken

Hvad sker der hvis variabelen `counter` ikke nedtælles?

Generelt format:

```
while ( betingelse ) {  
    kommandoer ;  
}
```

Den virker sådan:

- (1) *udregn betingelse*
- (2) *hvis forskellig fra 0 (dvs. SAND), så udfør kommandoer, og fortsæt ved (1)*

Dvs. `while`-løkken kører så længe *betingelse* er SAND

Indmaden i `while`-løkken kaldes *løkke kroppen*

Ofte anvendt `while`-konstruktion — `love.php`:

```
initialisering ;  
while ( betingelse ) {  
    kommandoer ;  
    optælling ;  
}
```

## Og til de dovne

- ▶ `$counter = $counter+1` er det samme som `$counter++`;
- ▶ `$counter = $counter-1` er det samme som `$counter--`;

# Eksempler på while-løkker — lokker1.php

Standard-løkke, hvor i gennemløber -----:

```
$i = 10;
while ( $i >= 1 ) {
    $i = $i - 1;
}
echo "Løkkeeksempel 1: $i <br>";
```

Standard-løkke, hvor i gennemløber -----:

```
$i = 1;
while ( $i <= 10 ) {
    $i = $i + 2;
}
echo "Løkkeeksempel 2: $i <br>";
```

Standard-løkke, hvor i gennemløber -----:

```
$i = 1;
while ( $i <= 100 ) {
    $i = $i * 2;
}
echo "Løkkeeksempel 3: $i <br>";
```

Hvilken værdi indeholder i efter hver løkke?

Skriv en while-løkke der udskriver 64, 32, 16, 8, 4, 2, 1:

```
$i = __ ;  
while ( __ >= __ ) {  
    echo "$i, ";  
    $i = __ / __ ;  
}
```

Skriv en while-løkke der udskriver 2, 4, 6, 8, ..., 100:

```
$i = __ ;  
while ( __ <= __ ) {  
    echo "$i, ";  
    $i = $i + __ ;  
}
```

Skriv en while-løkke der udskriver 100, 110, 120, ..., 200:

```
$i = __ ;  
while ( $i <= __ ) {  
    echo "$i, ";  
    $i = $i + __ ;  
}
```

Det er muligt for PHP kode at benytte data som indtastes af brugere i en form. Eksempel: veksle.html

```
<h2>Vekslebank</h2>
```

```
<form action="veksle.php" method='get'>  
  <p>Indtast beløb i kroner: </p>  
  <p><input name="kroner"></p>  
  <p><input type="submit" value="Beregn Dollarbebeløb"></p>  
</form>
```

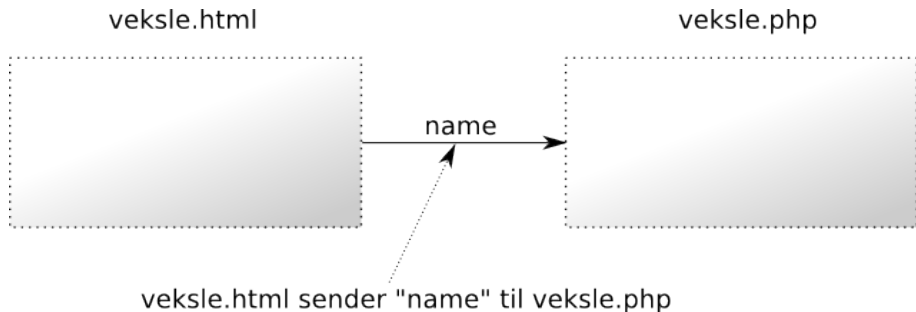
Eksempel: veksle.php

```
<h2>Vekslebank</h2>
<?php
    $kurs = 5.43;
    $gebyr = 20.0;
    $kroner=$_REQUEST['kroner'];
    $dollars = ($kroner - $gebyr) / $kurs;
    $dollars = number_format($dollars, 2, ",", ".");
    echo "For kr. $kroner modtager du \$$dollars
        (når en dollar koster $kurs)"; ?>
    <p><a href="veksle.html">Ny beregning</a></p>
```

Hvilke problemer er der ved denne veksleservice? Er servicen robust?

# Hvad var det lige der skete der?

1. veksle.html sender værdien af  
`<input name="kroner">`  
til `<form action='veksle.php'>`
  2. veksle.php spørger efter værdien af  
`<input name="kroner">`  
ved hjælp af `$_REQUEST['kroner']`
- 
1. `$_REQUEST['navn']` benyttes til både POST og GET data
  2. Derudover findes
    - ▶ `$_GET['name']` der kun henter GET data
    - ▶ `$_POST['name']` der kun henter POST data



Når et PHP-script (en fil med endelse `.php`) lægges under en brugers bibliotek udføres scriptet når en klient forespørger filen fra hjemmesiden

*`http://itu.dk/stud/f2008/DSDS/login/test.php`*

- ▶ Temperaturomregning
- ▶ Gangeservice
- ▶ Æblegrødsservice

Det er en rigtig god ide at lave sitemaps / flowcharts

Vi fortsætter med PHP:

- ▶ Teknologier for sites der er programmer
- ▶ for-løkker
- ▶ Indbyggede funktioner
- ▶ Brugedefinerede funktioner
- ▶ Genbrug af kode