



Avdeling for informatikk og e-læring, Høgskolen i Sør-Trøndelag

# Kom i gang med PHP

Svend Andreas Horgen

30.01.2007

Lærestoffet er utviklet for faget LV197D Webprogrammering med PHP

## 1. Kom i gang med PHP

*Resymé: Denne leksjonen skal gi deg overblikk over noen kjennetegn med PHP, og du skal lage dine første dynamiske websider. For å lykkes med webprogrammering er det viktig å forstå samspillet mellom klient og tjener. Før du leser denne leksjonen må du enten ha installert nødvendig programvare lokalt på egen maskin, eller ha tilgang til å legge ut de scriptene du lager hos for eksempel en Internettleverandør (ISP). En egen støtteleksjon tar seg av installasjon for de som måtte trenge det.*

*Du vil også få en presentasjon av faget og strategier for å jobbe med det – basert på dine forkunnskaper.*

*De tre første kapitlene av boka har to formål: å gjennomgå syntaksen som brukes i PHP, og å introdusere grunnleggende programmering. I løpet av denne første leksjonen vil du få laget noen få script. Dette kurset forutsetter at du allerede har programmert tidligere i et (hvilket som helst) annet språk. Hvis det ikke er tilfelle eller du skulle trenge å gjenoppfriske grunnleggende programmeringskunnskaper, kan du lese de første kapitlene i boka mer grundig enn antydnet i de første leksjonene.*

1.1.	INTRODUKSJON – HVEM ER DETTE KURSET FOR? .....	2
1.1.1.	Eksperter allerede, nybegynner eller et sted midt mellom? .....	2
1.1.2.	Viktig notis om læreboka .....	2
1.1.3.	Samspillet mellom leksjonene og læreboka .....	3
1.1.4.	Kodesnuttene på web .....	3
1.2.	Å PROGRAMMERE I PHP .....	3
1.2.1.	Tjeneren gjør det den blir bedt om .....	3
1.2.2.	Hvorfor script-programmering? .....	4
1.2.3.	Ditt første program .....	5
1.3.	SYNTAKS – VARIABLER, TEKSTSTRENGER OG ANDRE DATATYPER .....	6
1.3.1.	Kommentarer .....	7
1.3.2.	Variabler og tekststrenger .....	7
1.3.3.	Litt mer avansert om tekststrenger .....	7
1.3.4.	Datatyper i PHP .....	8
1.3.5.	Operatorer .....	8
1.3.6.	Konstanter .....	9
1.3.7.	Løsning av oppgave 2-1 fra boka (sist i kapittel 2) .....	9
1.4.	OPPSUMMERING .....	11

## 1.1. Introduksjon – hvem er dette kurset for?

Velkommen til kurset ”Webprogrammering med PHP”! Du vil i dette kurset lære webprogrammering slik at du er i stand til å lage dynamiske og interaktive løsninger med PHP som programmeringsspråk. Du skal lære å lagre data på både filer og i databasetabeller, uten at du må få bakoversveis av den grunn. Nettopp koblingen mot databaser er spesielt enkel i PHP – og du vil snart komme til et stadium der dette er helt naturlig å lære. Du vil også få lære om sikkerhet. Hvordan kan du beskytte informasjonen din, eller stole på at ingen kan gjøre vellykkede hack mot webtjeneren?

Det er mange grunner til å velge PHP som språk for utvikling av dynamiske og interaktive websider. Kapittel 1.1 i boka lister en del fordeler med PHP. Kort fortalt er PHP veldig enkelt å komme i gang med, det er gratis og har stor utbredelse. Både PHP 4 og PHP 5 kan brukes i dette kurset. Webprogrammeringen er den samme, men støtten for mer avanserte teknikker er mye bedre i PHP 5, og det er ingen grunn til å installere PHP 4 når du kan velge versjon 5. Har du en eldre versjon fra før, er det ingen grunn til å oppdatere.

### 1.1.1. Ekspert allerede, nybegynner eller et sted midt mellom?

Kan du PHP fra før og tar dette kurset primært for å dokumentere din kunnskap? Du skal ikke tvinges til å lese noe du allerede kan! Kurset er pedagogisk sett lagt opp slik at en som ikke har web-programmert med PHP før skal lære dette. Estimert arbeidsinnsats er 1 arbeidsdag per uke gjennom 12-15 uker – og dette gjelder som sagt for den som fra før av kan grunnleggende programmering, litt HTML, men ikke noe mer.

Hvis du kan PHP fra før, trenger du mindre jobbing for å ta kurset enn standardstudenten, noe som er naturlig siden du gjennom andre kilder har brukt tid på å lære stoffet. Læringsmålene som finnes på ressursiden til hver leksjon vil være til nytte for å bestemme hvorvidt du behersker stoffet eller ikke. Gå så til øvingen. Klarer du å løse øvingen tilhørende en leksjon og får bra tilbakemelding fra din veileder, er det ikke nødvendig at du leser det tilhørende kapittelet i boka eller leksjonen. Du kan eventuelt skimme gjennom leksjon/boka/andre bøker om du vil få en kort repetisjon av stoffet eller snappe opp detaljer.

### 1.1.2. Viktig notis om læreboka

På fagsidene kan du lese at vi forutsetter at grunnleggende programmering er kjent før kurset starter. Du må også forstå enkel HTML. Du trenger ikke å være noen kløpper i programmering, men bør kjenne til bruken av variabler, kontrollstrukturer (if-then-else, while, for), funksjoner og matriser (arrays). Vi vil i kurset ikke foreta noen opplæring i grunnleggende programmering. **Merk:** Dersom du mangler denne nødvendige forkunnskapen, kan du lære det du trenger ved å bruke mer tid på stoffet i bokas kapitler 2, 3, og 4 (samt siste halvdel av kapittel 5) enn det leksjonene gjør.

Uavhengig av om du kan programmering eller ikke skal du **ikke** hoppe over disse kapitlene, fordi de gjennomgår viktige prinsipper i webprogrammering. Av tema som blir gjennomgått i de kommende leksjonene kan nevnes: Syntaks, bruk av variabler i PHP (flere fallgruver å være obs på), skjemabehandling, tips og triks, strukturering og generalisering av kode, og ikke minst bruk av assosiative matriser. Dette stoffet finnes i boka i kapittel 2-5. Kapittel 6 i boka tar for seg behandling av strenger, datoer og matriser. Når denne basisen er på plass introduseres mer avanserte tema innen webprogrammering, som for eksempel

tilstandsbevaring, database- og filbehandling, automatisk produksjon av grafikk, og sikkerhet, noe som i praksis dekkes av resten av læreboka.

### 1.1.3. Samspillet mellom leksjonene og læreboka

Leksjonene og øvingene i dette kurset gir både en oppsummering av stoffet som gjennomgås i boka, og en ny vinkling på pensumstoffet. For eksempel vil leksjonene kunne ha større eksempler som syr sammen kunnskapen som er tilegnet etter å ha lest i boka. Les gjerne boka og leksjonen parallelt, eller eventuelt først de aktuelle sidene i boka og deretter leksjonen. Du vil alltid få rettleiding i leksjonen om hva du bør lese i boka. Det som kan være forvirrende av og til er nummereringen i boka kontra leksjon. Leksjon 3 har kanskje et avsnitt som heter 3.2 og som omtales med kapittel 3.2. mens det samtidig refereres til kapittel 4.2.3 i boka. Sidetall henviser som regel til boka, selv om det ofte står ”side 45” i stedet for ”side 45 i boka”. (Ingen leksjoner er mer enn 10-15 sider). Dersom det er steder der dette ikke kommer tydelig nok fram eller er forvirrende, så ikke nøl med å melde tilbake til faglærer. Alle tilbakemeldinger tas konstruktivt. Ønsket fra vår side er å gjøre faget best mulig for dere.

Dersom du har en tidligere utgave enn nyeste, vil noen henvisninger til sidetall, figurer, kapittelnumre etc. fra leksjonene ikke stemme helt. Du får evt. spørre på forumet til faget dersom dette blir et problem.

### 1.1.4. Kodesnuttene på web

Når du skal teste ut kode fra læreboka, kan du enten skrive inn all koden selv i en egen fil, eller du kan benytte lenken du finner under bokas ressurside på <http://www2.tisip.no/boker/php/>. Du kan både teste hver enkelt kodesnutt (med visse unntak), du kan se på kildekoden, og du kan laste ned en zip-fil med alle kodesnuttene fra et kapittel. Ønsket er at du som programmerer skal få velge selv om du vil teste kodesnutter mens du lærer stoffet, om du vil skrive inn alt for egen hånd og dermed få programmeringen lettere i fingrene, eller om du vil modifisere koden fra eksemplene og dermed prøve ut alternative løsninger på egen hånd.

## 1.2. Å programmere i PHP

Her er et kort sammendrag av det som står i lærebokas kapittel 1.3. Bruk tid på å forstå samspillet mellom klient og tjener, for da er det mye lettere å kode smart, sikkert og finne feil. Gå gjerne tilbake til bokas figurer 1.6 og 1.9 etter å ha lest denne og andre leksjoner. Figurene oppsummerer nemlig bra hva som skjer ved utføring av et PHP-script, men dette kan være vanskelig å forstå dybden av allerede nå.

### 1.2.1. Tjeneren gjør det den blir bedt om

Figur 1.6 i boka viser hva som skjer når en person besøker en nettside: *Klienten* sender en forespørsel til riktig *tjener* om å få tilsendt websiden. (Selve prosessen bak det å finne fram til riktig tjener, er ikke viktig for forståelsen av webprogrammering). Tjeneren behandler så forespørselen, og sender tilbake informasjonen som det ble spurt etter. En tjener kan håndtere mange forespørsler, tilsynelatende samtidig.

Mer konkret betyr det at hvis Kari skriver inn adressen til VG i sin nettleser ([www.vg.no](http://www.vg.no)) vil hennes maskin ta kontakt med tjeneren til VG og be om å få tilsendt forsiden. Tjeneren sender forsiden tilbake til Karis maskin i form av vanlig HTML, slik at hun kan lese dagens nyheter.

Det fins en rekke internettsider som har etternavn *.php*. Når du som bruker klikker en lenke eller skriver inn en adresse, for eksempel

<http://www.aitel.hist.no/fag/php/lek01/kode/forste.php>

vil det sendes en forespørsel fra din maskin (klienten) til tjeneren ved AITeL om å få tilsendt informasjon. Siden det som forespørres ikke er en vanlig html-side, men et PHP-script, vil *PHP-tolkeren* på tjeneren utføre koden i scriptet og *returnere resultatet* i form av *vanlig HTML*. Det som skjer er veldig viktig og oppsummert i punktene under:

- Klienten forespør en tjener om en side, for eksempel [www.tjener.no/ettEllerAnnet.php](http://www.tjener.no/ettEllerAnnet.php).
- Tjenermaskinen som ligger bak domenet [www.tjener.no](http://www.tjener.no) ser at klienten ber om et PHP-script. Dermed sendes kontrollen over til PHP-tolkeren.
- Tolkeren utfører *koden* som ligger i filen `ettEllerAnnet.php` på tjeneren.
- Resultatet av kjøringen blir (som regel) vanlig HTML.
- HTML-informasjonen sendes tilbake til klienten og vises i nettleseren.

Ved å installere tjenerprogramvare lokalt på din egen maskin, kan du *simulere* kommunikasjonen mellom tjenermaskin og klientmaskin. Dermed kan du utvikle og teste PHP-scriptene du lager selv om du ikke er tilkoblet Internett, noe som har flere fordeler. Først og fremst kan du spare tellerskritt (ved bruk av modem). Med dagens utbredte bruk av bredbånd, er det et viktigere poeng at du slipper å eksponere en uferdig webbløsning på Internett.

### 1.2.2. Hvorfor script-programmering?

Det er altså grovt sett to muligheter for tjeneren når informasjon skal sendes til nettleseren:

1. Returnere en statisk HTML-fil.
2. Kjøre et program/script som lager HTML-koden som skal returneres.

I dette kurset skal du lære å gjøre det siste, altså lage kode med PHP. Ved å programmere det som skal returneres åpner det seg en rekke interessante muligheter. Du kan lage websider som for eksempel:

- Viser forskjellig innhold avhengig av hvilket tidspunkt nettstedet besøkes (leksjon 2).
- Gir informasjon avhengig av hva brukeren har gjort tidligere (leksjon 6).
- Kan håndtere de data brukeren fyller ut i et skjema (leksjon 3).
- Sender e-post med nyhetsbrev til abonnenter eller rapport til systemansvarlig om forsøk på hacking når de måtte inntreffe (leksjon 5).
- Husker informasjon over tid eller på tvers av forespørsler (leksjon 6). Nyttig i handlekurvløsninger.
- Er skreddersydd til hver enkelt person (leksjon 4).
- Gir brukeren mulighet for å laste opp bilder slik at andre kan se bildene (leksjon 7).
- Har informasjonen lagret i en database, med alle de mulighetene det byr på (leksjon 8 og 9).
- Gjør det vanskelig for en hacker å ødelegge (leksjon 10).

- Bare er tilgjengelige for de som er riktig innlogget (leksjon 11).
- Har grafiske fremstillinger av informasjon, der grafikken lages i det øyeblikk siden besøkes (leksjon 12).

Dette er bare noen av de tingene kurset kommer til å gå gjennom, og etter at kurset er fullført vil du beherske nok PHP og webprogrammering til å utforske mer av de avanserte mulighetene med PHP på egen hånd.

### 1.2.3. Ditt første program

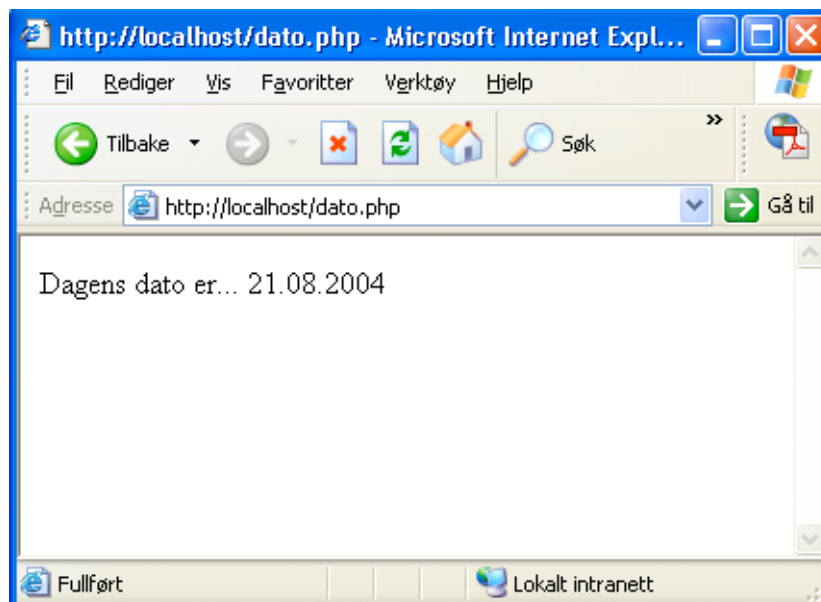
Lag en enkel fil og lagre som `dato.php`. Gi den følgende innhold:

```
<html><body>
<?php
    echo "Dagens dato er... ";
    echo date("d.m.Y");
?>
</body></html>
```

Legg merke til semikolonet på slutten av linjene som starter med `echo`. Semikolon brukes i PHP for å angi slutten av en setning/kommando. Lagre filen i den katalogen som er angitt som webrot, eller last den opp på tjeneren til din ISP. Når du utvikler lokalt må du skrive inn adressen

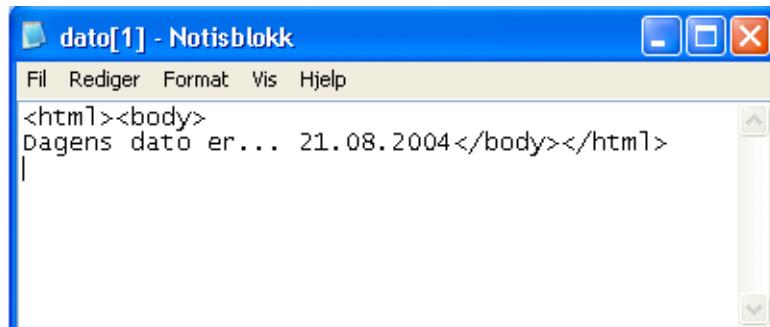
<http://localhost/dato.php>

i nettleseren. Fungerer dette, så er du i gang!



Det som skjer når brukeren besøker siden, er at scriptet kjøres *linje for linje* av tolkeren. Både PHP og HTML kan blandes i ett og samme script. Tolkeren bygger gradvis opp et resultat som til slutt skal sendes til klienten. Først treffer tolkeren på linjen `<html><body>`. Siden dette bare er vanlig HTML, gjøres ikke noe mer enn å la resultatet bestå av nøyaktig det samme. For å komme i PHP-modus brukes taggen `<?php` men også andre kan brukes (se kapittel 1.3.6 ”Noen detaljer” i boka). For å avslutte PHP-modus brukes taggen `?>`. I linje 2 i scriptet ser tolkeren derfor at nå kommer det programkode som skal utføres. Koden består av

to setninger, der det er semikolonet som skiller setningene fra hverandre, ikke linjeskiftene. Kodeordet `echo` skriver ut en tekststreng. Funksjonen `date()` returnerer dagens dato i form av en tekststreng. Dermed blir resultatet en liten tekst som sier at "Dagens dato er... 21.08.2004". Til slutt skal også `</body></html>` være med. Den komplette kildekoden som sendes til nettleseren ser slik ut:

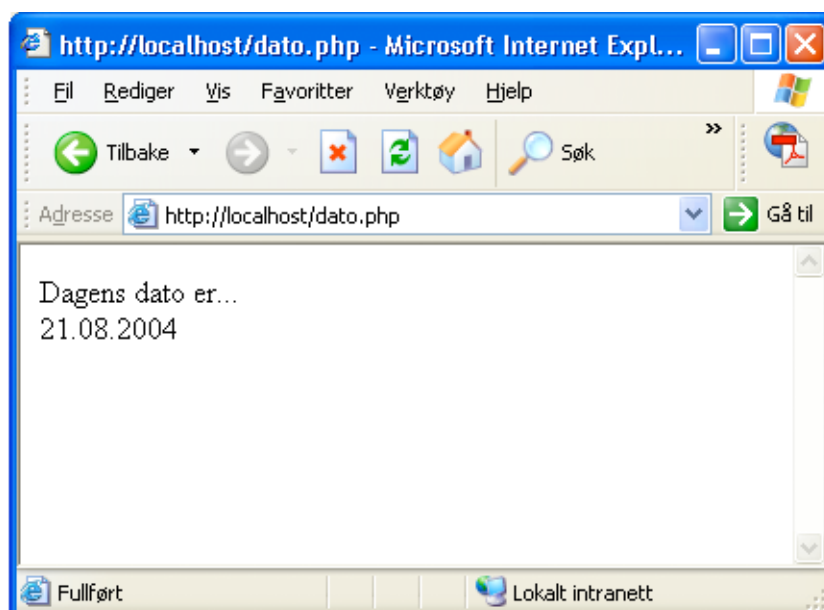


```
<html><body>
Dagens dato er... 21.08.2004</body></html>
```

Legg merke til at selv om de to `echo`-setningene står på hver sin linje, blir resultatet i nettleseren og i kildekoden at alt står på én linje. Grunnen er at det er HTML som sendes til nettleseren, og for å få linjeskift må en `<br>` tag brukes. Dersom den første `echo`-setningen endres til

```
echo "Dagens dato er... <br>";
```

vil kildekoden endres tilsvarende, og nettleseren vise to linjer med informasjon i stedet for en.



Prøv dette eksempelet selv, og se hva som skjer om du setter ordet "dato" i fet skrift. Husk at HTML-taggen for fet skrift er `<b>` (det noe mindre utbredte `<strong>` anbefales for å sikre kompatibilitet i fremtiden).

### 1.3. Syntaks – variabler, tekststrenger og andre datatyper

Du har nå en god basis for å forstå hva som skjer i kulissene når en webside laget i PHP besøkes. La oss ta fatt på litt mer programmering. Kapittel 2 i boka bør leses selv om du har

programmert tidligere, for her gjennomgås karakteristiske trekk ved PHP som det er vel verdt å merke seg.

### 1.3.1. Kommentarer

Det fins mange former for kommentarer. Bruk av // og /\* \*/ er vanligst, men også # er mulig.

```
// dette er en kommentar som gjelder til neste linjeskift.  
/* her er en kommentar  
over flere  
linjer  
*/
```

### 1.3.2. Variabler og tekststrenger

Her er en kort oppsummering om bruken av variabler og tekststrenger i PHP:

- Variabler i PHP skal **ikke** deklarerer før bruk. Dette er en kilde til feil, men gjør også programmeringen enklere. Eksempel:

```
$navn = "Kari Olsen";  
echo $navn;
```

- Et dollartegn brukes foran alle variabelnavn, hver eneste gang variabelen brukes. Dette er nødvendig som en følge av at variablene ikke kan deklarerer. Dersom dollartegnet utelates, kan logiske feil oppstå. Se for eksempel figur 2.2 i boka.
- Sammenslåing av tekststrenger, gjøres med bruk av punktum-operatoren:

```
$navn = "Kari Olsen";  
$alder = 23;  
echo "Alderen til " . $navn . " er " . $alder . " år";
```

- For å tydelig markere de ulike tekststrengene kan en setning gå over flere linjer. Dette er mulig siden det er semikolon, ikke linjeskift, som markerer skillet mellom ulike kommandoer. Denne teknikken er spesielt hendig i forbindelse med databaser og SQL.

```
echo "Alderen til " .  
    $navn .  
    " er " .  
    $alder .  
    " år"  
    ;
```

- Variabler kan brukes i tekststrenger, uten at strengen må avsluttes først. Dette er mulig som en følge av at dollartegn brukes som prefiks. Eksempel:

```
$navn = "Kari Olsen";  
$alder = 23;  
echo "Alderen til $navn er $alder år";
```

### 1.3.3. Litt mer avansert om tekststrenger

For å få skrevet ut en lenke, må anførselstegn brukes i HTML. Det samme gjelder attributter i andre tagger. Det er valgfritt i henhold til HTML-standardene om en vil bruke anførselstegn eller fnutter. De to første setningene viser anførsler, mens de to siste bruker fnutter (Vi bruker ordet fnutt i stedet for apostrof, som du kanskje bruke i engelsk-fag, fordi fnutt er noe lettere å

skille fra anførselstegn enn apostrof. Heretter vet du at fnutt er enkel ' og anførselstegn er dobbel ". Boka bruker samme navngiving).

```
<a href="lenke.html">Her er en lenke</a>

<a href='lenke.html'>Her er en lenke</a>
<img src='bilde.jpg' height='100' width='70'>
```

Dersom en lenke lages slik med en echo-setning:

```
echo "<a href="lenke.html">Her er en lenke</a>";
```

vil ikke koden fungere som forventet. PHP oppfatter nemlig at tekststrengen bare går mellom de to første anførselstegnene:

```
"<a href="
```

Resultatet blir en feilmelding ala

```
"Parse error: parse error, unexpected T_STRING, expecting ',' or ';' in filnavn.php".
```

Det fins flere løsninger på dette problemet:

- Tekststrenger kan enten være omsluttet av anførselstegn, slik: "tekststreng" eller fnutter, slik: 'tekststreng'. Forskjellen er at *det bare letes etter variabler i strenger med anførselstegn*. En løsning på problemet med feilmeldingen over, er derfor å bruke både anførselstegn og fnutter i echo-setningen. Motsatt vei går også, men da vil eventuelle variabler inne i tekststrengen ikke tolkes.

```
echo "<a href='lenke.html'>Her er en lenke</a>"; //tolker variabler
echo '<a href="lenke.html">Her er en lenke</a>'; //ingen tolkning
```

- Det er mulig å angi at enkelte tegn ikke skal tolkes av PHP, men bare vises som et helt vanlig tegn. Du kan lese mer om såkalte escape-characters i boka på side 47.

```
echo "<a href=\"lenke.html\">Her er en lenke</a>";
```

### 1.3.4. Datatyper i PHP

I websammenheng er det aller meste tekst. Alt du ser i nettleseren, vil være tekstlig informasjon (hvis vi utelater bilder). For å kunne regne med tall, er det derimot nødvendig for PHP å benytte datatyper. Variabler skal som nevnt ikke deklarerer. Det er dermed PHP selv som sørger for hvilken datatype variablene har.

De som har programmert tidligere vil kanskje få tanker om at "her mister jo programmereren all kontroll". I 95% av tilfellene er det ikke noe minus at PHP selv bestemmer datatypene, og en trøst er at du alltid selv har mulighet til å overkjøre PHP ved å bruke såkalt casting. I tillegg fins flere funksjoner for å finne datatypen til en variabel og konvertere mellom ulike datatyper. Vi skal komme mer tilbake til praktisk bruk av bokas kapittel 2.2.4 "Lese eller endre datatypen" senere, så du trenger ikke å lese disse sidene så nøye i første omgang.

### 1.3.5. Operatorer

Stoffet i bokas kapittel 2.3 har du trolig vært borti før, så det eneste du trenger å gjøre er å skimme gjennom sidene for å se hvilken syntaks PHP bruker. Bruk av tilordningsoperatoren inne i betingelser, skal vi komme mer tilbake til i forbindelse med databaser og filer. Legg merke til hurtignotasjonen for å øke/reducere verdier:

```
<?php
    $tall = 24;
    $tall ++; //har nå innholdet 25
    $tall /= 2; //har nå innholdet 12.5
    echo $tall; //skriver ut 12.5
?>
```

De logiske operatorene har en litt annen syntaks enn du kanskje er vant med fra før.

- `&&` betyr **AND** (tastetrykk shift + 6 på PC for å få &-tegnet)
- `||` betyr **OR** (finnes ved siden av ett-tallet på PC-tastaturet)
- `!` betyr **NOT** (vanlig utropstegn)

Det er viktig å bruke dobbel `&&`. Et enkelt `&` betyr noe annet, det samme gjelder for `||`. Ved testing på likhet er det også viktig å bruke dobbel `==`, siden enkel `=` bare betyr tilordning.

### 1.3.6. Konstanter

Legg merke til at konstanter ikke skal ha dollartegn foran seg. Konstanter må opprettes før bruk, det gjøres ved kodeordet `define`. Vi kommer tilbake til praktisk bruk av konstanter senere. Det er lurt å bruke store bokstaver i konstanter.

### 1.3.7. Løsning av oppgave 2-1 fra boka (sist i kapittel 2)

For å vise nytten av variabler, kommer løsningen på første oppgave i slutten av kapittel 2 her. Oppgaven går på å lage et script som har to variabler, ditt navn og din alder. Informasjonen skal skrives ut i en tabell, en nummerert liste, en punktmerket liste og inne i en paragraf.

1. Det første du må gjøre er å lage en ny fil. Husk å kalle denne med etternavn `.php`, for eksempel `oppgave2-1.php`
2. For å lage en variabel må du først fortelle PHP-parseren at den skal bytte til PHP-modus. Det gjøres med taggen `<?php`
3. Skriv følgende kode og husk semikolonene:

```
<?php
    $alder = 45;
    $navn = "James Bond";
?>
```

4. Dersom du avslutter PHP-modus før du skriver tabellen, kan du starte PHP-modus akkurat der du trenger informasjonen innsatt:

```
<table border="1">
    <tr>
        <th>Navn</th> <th>Alder</th>
    </tr>

    <tr>
        <td><?php echo $navn; ?></td> <td><?php echo $alder; ?></td>
    </tr>
</table>
```

5. Når mindre HTML-kode, som for eksempel den nummererte listen, skal skrives ut, er det kanskje like enkelt å bruke php-modus. Det er opp til deg hva du liker best:

```
<?php
    echo "<ul>";
    echo "<li>Navnet er $navn</li>";
    echo "<li>Alderen er $alder</li>";
    echo "</ul>"; //avslutter punktmerket liste
?>
```

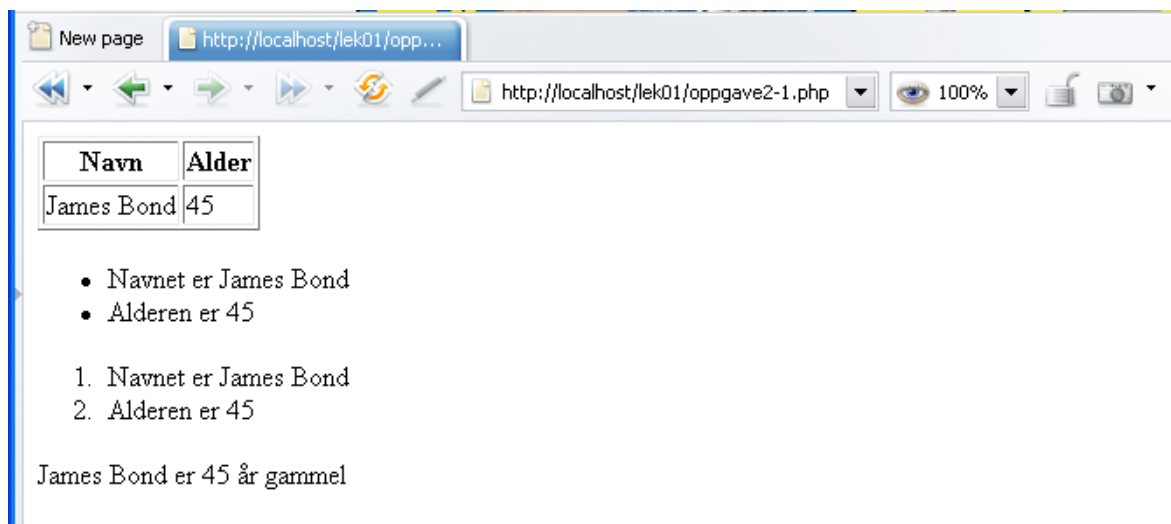
6. Nummerert liste kan nå enkelt lages ved å kopiere den punktmerkede, men endre fra ul til ol.

```
<?php
    echo "<ol>";
    echo "<li>Navnet er $navn</li>";
    echo "<li>Alderen er $alder</li>";
    echo "</ol>"; //avslutter punktmerket liste
?>
```

7. Paragrafen er også rett fram:

```
<?php
    echo "<p>$navn er $alder år gammel</p>";
?>
```

8. Lagre filen (i en underkatalog av det som er satt som rot i webtreet jmf. side 25 i boka)
9. Skriv inn riktig URL i nettleseren, for eksempel <http://localhost/lek01/oppgave2-1.php>



Som du ser vil variablene som ble opprettet øverst i scriptet, tilgjengelig helt til siste slutt, selv om PHP-tolkeren bytter flere ganger mellom HTML- og PHP-modus underveis. Det er ikke nødvendig å hoppe inn og ut av PHP-modus hver gang som gjort i punkt 5 og 7, når det bare er PHP-kode som kjøres hele tiden. Følgende kode gir samme resultat, og er også mer lesbart:

```
<?php
    echo "<ul>";
    echo "<li>Navnet er $navn</li>";
    echo "<li>Alderen er $alder</li>";
    echo "</ul>"; //avslutter punktmerket liste

    echo "<ol>";
    echo "<li>Navnet er $navn</li>";
    echo "<li>Alderen er $alder</li>";
    echo "</ol>"; //avslutter punktmerket liste

    echo "<p>$navn er $alder år gammel</p>";
?>
```

Du kan ta det som en ekstra øvelse å lage tabell-biten med PHP-kode, og listene i HTML-modus.

## 1.4. Oppsummering

I denne leksjonen har vi sett på fordeler med PHP, og utforsket samspeillet mellom klient og tjener. Du skal nå ha en konseptuell forståelse av hva som skjer når et PHP-script utføres. Bokas kapittel 2 tar for seg både grunnleggende begreper innen programmering, og syntaksen som brukes i PHP. Det kan være en liten omstilling å gå fra et annet språk til å lære PHP, men du vil merke at det grunnleggende stort sett er det samme. Vær obs på særegenhetene ved PHP, og ikke nøl med å stille spørsmål på fagets diskusjonsforum hvis du står fast eller lurer på noe. Det er helt naturlig å oppleve små eller større problemer i starten, og ofte vil det være en liten fillefeil som produserer feilmeldingen. Det er bra å få friske øyne til å se på tilsynelatende umulige problemer!

Du har, etter å ha jobbet med denne leksjonen, forhåpentligvis fått et godt grunnlag for å programmere i PHP og kan gå i gang med øvingen. Du finner innleveringsfristen og selve øvingen på fagets nettsider. Håper det smakte og at du har fått lyst på mer. Bon appetit med PHP – fra nå av er det forresten du som blir kokken!