Institutt for informasjonssystemer

# Kom i gang med webutvikling og PHP

IS-115 Webprogrammering i PHP | Peter André Busch

## Innhold

# Leksjon 1: kom i gang med webutvikling og PHP

I denne modulen skal vi komme i gang med HTML/CSS og PHP spesielt. Det innebærer å få satt opp et utviklingsmiljø for PHP og testet at det fungerer. I tillegg er det fordelaktig å bruke webdesignprogramvare for å utvikle webapplikasjonene. Vi legger opp til bruk av Microsoft Visual Studio Code. Vi ser også litt på hvordan HTML og CSS fungerer. I denne modulen skal vi også se på grunnleggende programmeringssyntaks i PHP. Alle språk har sine variasjoner i hvordan de definerer variabler, tekststrenger og konstanter. Videre, har de sine egne måter å skrive tekst ut på skjermen, avslutte tekststrenger på og lage snarveier på. Dette skal vi se nærmere på i denne leksjonen. Denne syntaksen er helt grunnleggende for PHP og noe alle må kunne.

## 1. Introduksjon – hvem er dette emnet for?

Velkommen til emnet IS-115 Webprogrammering i PHP! Du vil i dette emnet 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 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. Støtten for mer avanserte teknikker er bedre i PHP 7 og av sikkerhetsmessige årsaker bør du unngå å bruke eldre versjoner av PHP. De fleste ISPer (for eksempel via et webhotell) benytter alltid siste versjon av PHP, som nå er PHP 8+. Vi anbefaler at du kjører installasjonen lokalt og benytter XAMPP eller Docker til dette. Det vil bli lagt opp til en egen workshop for dette.

### 1.1 Ekspert allerede, nybegynner eller et sted midt mellom?

Kan du PHP fra før og tar dette emnet primært for å dokumentere din kunnskap? Du blir ikke tvunget til å lese noe du allerede kan! Emnet er pedagogisk lagt opp slik at hver enkelt student kan følge en progresjon som passer til en selv (læring i eget tempo). Det vil si at en student som ikke har web-programmert med PHP før skal lære dette, mens den erfarne studenten kan gå fort gjennom modulene. Estimert arbeidsinnsats er ca. 10 timer per uke gjennom 16 uker. Hvis du kan PHP fra før, trenger du mindre jobbing for å ta emnet enn standardstudenten, noe som er naturlig siden du gjennom andre kilder har brukt tid på å lære stoffet. Læringsmålene som finnes på ressurssiden til hver leksjon vil være til nytte for å bestemme hvorvidt du behersker stoffet eller ikke. Det er også disse du vil prøvd på av faglærer/hjelpelærer i gjennomgangen etter hver modul. Dersom du klarer 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 nøye. Du kan eventuelt skumme gjennom leksjon/boka/andre bøker om du vil få en kort repetisjon av stoffet eller snappe opp detaljer. Det er lagt ut en oversikt over minimum fremdrift i emnet. Denne oversikten viser hvordan du bør ligge an tidsmessig i forhold til de ulike modulene for å få en karakter som du er fornøyd med.

### 1.2 Viktig notis om læreboka

Du må ikke bruke den anbefalte læreboka, og du kan også klare deg helt uten bok om du vil det, men emnet er lagt opp for normalstudenten og tar utgangspunkt i at du har læreboka tilgjengelig. På emnesidene kan du lese at vi forutsetter at grunnleggende programmering er kjent før emnet starter. Du må også forstå enkel HTML for å komme i gang med emnet. *Merk:* Dersom du mangler

grunnleggende programmeringskunnskap, kan du lære det du trenger ved å bruke mer tid på stoffet i bokas kapitler 1-6 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 matriser. Dette stoffet finnes i boka i kapittel 2-6. 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.

## 1.3 Komme i gang

Det er noen ting du må huske på for å komme i gang med PHP. Vi bruker tid på dette i begynnelsen av emnet på lab.

### Webområde og MySQL

Du har eget webområde på home.uia.no, men her har du ikke direkte tilgang til en MySQL-database. I tillegg vil du miste tilgang til webområdet når du avslutter studiene. Vi bruker XAMPP eller Docker for å programmere, da gjør du alt lokalt på maskinen din (dette er vår anbefalte løsning). Du kan også bruke et webhotell hos en annen ISP hvis du ønsker det. Noen ISPer lar deg benytte *phpMyAdmin* (andre gjør det ikke av sikkerhetshensyn). Andre ISPer kan være spesielt nyttig om du ønsker å kjøpe et eget domene. Da kan du kjøpe et eget domene med webhotell (eller ev. peke domenet du har kjøpt mot et eksisterende webhotell du har – du trenger ikke et eget webhotell for hvert eneste domene du kjøper).

### HTML- og CSS-kode

Du må selv finne eller lage HTML- og ev. CSS-koden du vil bruke til prosjektsystemet. Dette trenger ikke å være avansert ettersom dette ikke er fokuset i emnet. Jeg har gjort tilgjengelig noen design til deg i Canvas som du kan benytte om du ønsker. Du står fri til å finne noe annet eller lage dette selv.

### Programvare og utstyr

Du kan bruke både PC og Mac, Windows og Linux/Unix i dette faget. PHP fungerer på alle plattformer. Docker kan imidlertid ikke kjøres på alle plattformer (fungerer på Windows/Mac – mer om dette i workshop). For å gjøre øvingene trenger du en teksteditor for å programmere i. Du kan velge for eksempel notisblokk (anbefales ikke), TextPad og phpDesigner (anbefales sterkere) eller mer avanserte verktøy (f.eks. Microsoft Visual Studio Code anbefales sterkt i dette kurset da vi legger opp til dette). Se oversikt i Canvas. Textpad, Visual Studio Code og Dreamweaver (samt andre) viser fargekoder når du programmerer. Dette er en nyttig funksjon. Textpad og phpDesigner er ideelt grunnet deres kraftige funksjonalitet. TextPad er gratis å bruke. Det er mye enklere å designe websider ved hjelp av avansert webdesignprogramvare, men dette må du betale for. Du kan imidlertid bruke Microsoft Visual Studio gratis. Du har gunstige avtaler gjennom universitetet dersom du ønsker der. Sjekk dette ut på UiAs sider og programvareoversikten i Canvas. Det fins også en rekke andre miljø som er spesielt beregnet på PHP. PHP-coder er et eksempel, og flere kommer kanskje på diskusjonsforumet til kurset etter hvert. Se også Canvas for ulik programvare som kan benyttes.

Det finnes en rekke gratisklienter som hjelper deg med å laste opp filene dersom du bruker et webområde. Programvare som FileZilla og WinSCP er egnet til formålet. Hvis du benytter webdesignprogramvare, så kan SSH-/SFTP-klient være innebygd.

## 1.4 Ulike ressurser

Det finnes en flora av ressurssider for å lære seg PHP, alt fra YouTube-videoer til egne nettsted som er laget for formålet. Her kan du fordype deg i mange timer om du ønsker det. Vær også oppmerksom på at introduksjonssiden til hver modul lister en rekke ressurser. Jeg ønsker likevel å gjøre deg oppmerksom på noen få av de viktigste ressursene.

### InLearning

InLearning er det nye navnet på lynda.com hvor dere som studenter har fri tilgang til en rekke kurs-ressurser.

### PHP sine nettsider

PHP har egne nettsider (www.php.net) hvor det finnes mye informasjon om språket slik som innebygde funksjoner. Gjør deg litt kjent her.

### HTML og CSS

W3schools - Er det lenge siden du har holdt på med HTML? Vi anbefaler at du gjenoppfrisker litt før du starter på kurset. Det fins flere alternativer enn denne lenken, blant annet den danske HTML.dk.

### Læreboka

Ressursside for læreboka: lærphp.no (blir oppdatert snart)

# 2. Å programmere i PHP

Her er et kort sammendrag av noe av det som står i lærebokas kapittel 1. 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 dette kapitlet etter å ha lest denne og andre leksjoner. Kapitlet oppsummerer nemlig bra hva som skjer ved utføring av et PHP-script, men dette kan være vanskelig å forstå dybden av allerede nå.

## 2.1 Tjeneren gjør det den blir bedt om

Figur 1.4 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 – og dette er noe mange av dere har gått igjennom allerede i emnet IS-105). 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) 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, vil det sendes en forespørsel fra din maskin (klienten) til tjeneren om å få tilsendt informasjon. Siden det som forespørres ikke er en vanlig html-side, men et PHP-script, vil *PHP-tolkeren* (PHP parser) 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.

- Tjenermaskinen som ligger bak domenet 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.
- Merk at PHP-kode alltid kjøres før resten av koden i en fil (f.eks. HTML). PHP er et rekursivt akronym for *PHP Hypertext Preprocessor*.

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. To fordeler er at du slipper å eksponere en uferdig webløsning på Internett (dette finnes det også andre løsninger for å unngå, se pkt. 2.5) samt at du slipper å tenke på tilgang til Internett (men det har de fleste i dag).

## 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 deler av HTML-koden som skal returneres (som regel benyttes PHP sammen med HTML, men ikke alltid)

I dette emnet 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 (modul 3).
- Gir informasjon avhengig av hva brukeren har gjort tidligere (modul 8).
- Kan håndtere de data brukeren fyller ut i et skjema (modul 4).
- Sender e-post med nyhetsbrev til abonnenter eller rapport til systemansvarlig om forsøk på hacking når de måtte inntreffe (modul 10).
- Husker informasjon over tid eller på tvers av forespørsler (modul 8). Nyttig i handlekurvløsninger.
- Er skreddersydde til hver enkelt person (modulene 7-8).
- Gir brukeren mulighet for å laste opp bilder slik at andre kan se bildene (modul 9).
- Har informasjonen lagret i en database, med alle de mulighetene det byr på (modul 7).
- Bare er tilgjengelige for de som er riktig innlogget (modul 8).

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

## 2.3 Ditt første program

Lag en enkel fil og lagre som **hei.php**. Gi den følgende innhold:

```
<html><body>
    <?php
        $hilsen = "Hei verden!";
        echo "<p> " . $hilsen . "</p>";
    ?>
</body></html>
```

I dette eksemplet blander vi HTML og PHP. Først kjøres PHP-koden på tjeneren og deretter returneres hele filen med eksisterende og PHP-generert HTML sammen. Legg merke til semikolonet på slutten av linjene. Semikolon brukes i PHP for å angi slutten av en setning/kommando. Punktumet i `echo`-linje nr. 2 binder sammen tekst. Lagre filen i den katalogen som er angitt som webrot (htdocs i XAMPP), eller last den opp på tjeneren til din ISP. Når du utvikler lokalt, må du skrive inn adressen

http://localhost/hei.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 2 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. Dermed blir resultatet en liten tekst som sier at «Hei verden!». Til slutt skal også `</body></html>` være med. Den komplette kildekoden som sendes til nettleseren ser slik ut:

```
<html><body>
<p>Hei verden!</p>
</body></html>
```

Legg merke til at flere `echo`-setninger som står på hver sin linje, vil resultere i at resultatet i nettleseren og kildekoden står på én linje. Grunnen er at det er HTML som sendes til nettleseren, og for å få linjeskift må en tag brukes. Dersom `echo`-setningen endres til

```
<?php
     echo "<p> " . $hilsen . "<br>Ny linje her.</p>";
?>
```

vil kildekoden endres tilsvarende, og nettleseren vise to linjer med informasjon i stedet for en. Prøv dette eksempelet selv. Se også hva som skjer om du setter hilsenen i fet skrift. Husk at HTML-taggen for fet skrift er `<strong>`.

## 2.4 Mulige feilmeldinger

Når du programmerer i PHP kan du komme over en rekke feilmeldinger. Noen webhotell viser feilmeldinger som standard, mens andre ikke gjør det. Du er avhengig av å se feilmeldinger for å kunne rette opp ev. feil. For å skru på feilmeldinger kan du inkludere følgende kode:

```
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

Det er imidlertid mulig å endre PHP Config via kontrollpanelet til enkelte ISPer. Du kan støte på en rekke feilmeldinger i PHP og du blir trent på å forstå og adressere dem. Vi kommer ikke til å gå gjennom alle mulige feilmeldinger her, det blir for omfattende. Men her noen mulige feil du kan støte på. Det er ikke

anbefalt å vise feilmeldinger på nettsteder i produksjon (altså på weben), men vi kan gjøre det lokalt når vi programmerer.

**1. Avsluttet funksjon i PHP.** Dersom du benytter en innebygd funksjon som ikke lengre støttes, vil du få en beskjed som denne:

**Deprecated**: mysql_query(): The mysql extension is deprecated and will be removed in the future: use mysqli or PDO instead in

2. Finner ikke filen. Dersom du henviser til en fil som ikke eksisterer på oppgitt filbane, får du følgende feilmelding:

**Warning**: require_once(/home/voll_2/epizy.com/epiz_25154495/htdocs/is217/site/inc/db.inc.php): failed to open stream: No such file or directory in **/home/voll_2/epizy.com/epiz_25154495/htdocs/is115/index.php** on line **9**

## 2.5 Beskytt webområdet ditt mens du utvikler

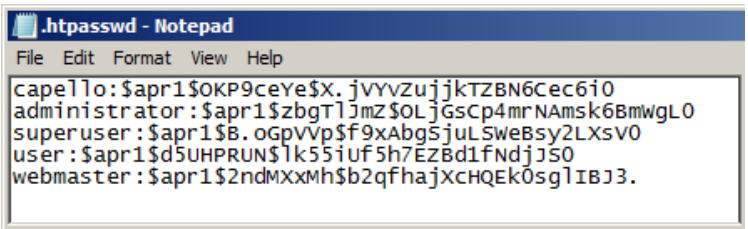
Dersom du laster opp filer til et webområde, kan du beskytte webområdet ditt mens du utvikler det ved å bruke htpasswd og htaccess. Da vil alle som forsøker å laste ditt nettsted bli promptet for brukernavn og passord som du selv setter. Dette er en enkel og sikker måte å beskytte filene dine på. For å bruke disse går du frem på følgende måte:

Du lager to filer i notisblokk og kaller dem henholdsvis .htaccess og .htpasswd (med punktumene først). Deretter skriver du inn følgende innhold i .htaccess-filen din:

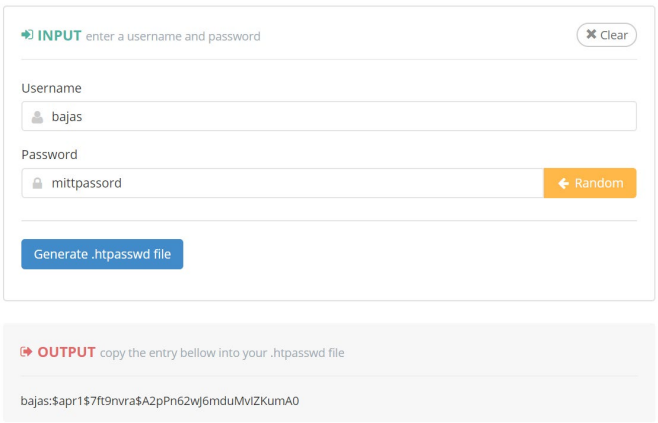
```
AuthName "Under development"
AuthUserFile /path/to/your/website/.htpasswd
AuthType Basic
Require valid-user
```

På linje 1 bak AuthName kan du skrive hva du vil inni anførselstegnene. På linje 2 legger du inn Linux-stien til din .htpasswd-fil (det er lurt å legge denne utenfor webroten hvis du kan). Linje 3-4 står slik de er. Filen .htaccess legger du i roten på den katalogen du ønsker å beskytte (dersom det er hele nettstedet ditt, så er det altså webroten). Denne filen beskytter rekursivt (dvs. alle underkataloger og filer blir beskyttet).

Filen .htpasswd består av en eller flere linjer (avhengig av hvor mange forskjellige brukere det skal være som kan logge seg på). Her er et eksempel på en slik fil:




I eksemplet ovenfor er det fem brukere som har tilgang. For å generere passordet, bruker du et nettsted som f.eks. https://www.web2generators.com/apache-tools/htpasswd-generator og skriver inn brukernavn og passord (ev. flere ganger dersom du skal ha flere brukere). Noe slikt som dette:




Kopier inn teksten i det grå feltet inn i .htpasswd-filen og lagre den der du vil ha den. Last opp filene, og vips så er du ferdig med å beskytte nettstedet ditt. Denne funksjonaliteten kan også brukes som innlogging til et backend av et system om du ikke behøver å lage innloggingsfunksjonalitet i PHP.

# 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.

## 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 */
```

Her er et eksempel fra et nettsted hvor koden for å koble til databasen ligger i en include-fil (mer om det senere) helt i begynnelsen av filen (før HTML):

```
1 ▼ <?php
2       //Connect to database
3       require_once 'site/inc/db_connection.inc.php';
```

## 3.2 Variabler og tekststrenger

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

- Variabler i PHP skal ikke deklareres 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 deklareres. 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";
```

## 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-standarden 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>
<img src="bilde.jpg" height="100" width="70">
<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 a la

”**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 teksttrengen 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>";
```

## 3.4 Datasnutter 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 deklareres. 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.

## 3.5 Operatorer

Stoffet i bokas kapittel 2.3 har du trolig vært borti før, så det eneste du trenger å gjøre er å skumme 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/redusere 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.

## 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.

### 3.7 Oppgaveeksempler

Sjekk læreboka for ulike eksempler som illustrerer bruken av variabler o.l.

## 4. Tips, triks og snarveier

På dette stadiet skal vi kun introdusere dere for enkelte løsninger som er gjort for å gjøre livet til PHP-programmereren enklere. PHP har en rekke slike løsninger som kan virke ulogiske ved første øyekast, men som man lærer seg å sette pris på etter hvert som man blir mer dreven i programmeringen. Det tar for lang tid å gå gjennom alle her, men vi begynner med noen ofte brukte løsninger. Første eksempel er fra en teller hvor vi enkelt kan øke verdien med én:

```
<?php
	$i++;
?>
```

Kortformen `$i++` benyttes istedenfor å skrive `$i = $i+1`. Dette er en rent praktisk forordning for noe som benyttes ofte. Et annet eksempel er en løsning for å skrive ut riktig substantivform (entall eller flertall) – noe som ser mye mer elegant ut (og faktisk er grammatisk riktig) enn for eksempel å skrive «du lastet opp 1 filer».

```
<?php
	$antall = 4;
	$dokOrd = ($antall==1) ? "dokument" : "dokumenter";
	echo $antall . " " . $dokOrd . " ble funnet.";
?>
```

Her kan du ev. også lage en egen funksjon (mer om dette i modul 2) som du kaller hver gang du ønsker å sjekke riktig substantivform (her må du da ev. tenke på hvordan du løser problemet med forskjellige substantiv). Dette var to smakebiter på hva du kan gjøre av tips og triks.

## 5. Oppsummering

I denne leksjonen har vi sett på fordeler med PHP, og utforsket samspillet 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 appetitt med PHP – fra nå av er det forresten du som blir kokken!