# Kapittel 1

# Kom i gang med PHP

**Læringsmål:**

Dette kapittelet vil fungere som en enkel oppstartsguide for å komme i gang med PHP. Du vil få lære om

- historien bak PHP
- installasjon av nødvendig programvare
- hvordan PHP gir dynamikk, gjennom å lage et enkelt «kom-i-gang-script»
- samspillet mellom tjener og klient

## 1.1 Hvorfor PHP?

Med PHP kan du programmere dynamiske, interaktive websider. Det betyr at det som vises i nettleseren, genereres i samme øyeblikk som noen besøker sidene. PHP er et fullverdig programmeringsspråk og kan dermed brukes til å lage enkle så vel som de mest avanserte webløsninger. Du har kanskje allerede lagd vanlige Internett-sider som er skrevet i HTML? Slike er statiske og passer for informasjon som ikke endres over tid. Endringer er mulige, men krever at en person aktivt går inn for å redigere sidene. Denne boka tar for seg hvordan PHP kan brukes til å lage dynamiske, fleksible, brukervennlige og ikke minst sikre løsninger.

PHP er et relativt nytt programmeringsspråk (lagd i 1994) og faller i kategorien *scriptspråk*. En forklaring på navnet PHP er *Personal Home Page (Tools)*. En annen vinkling er at PHP er et rekursivt definert akronym. Med dette menes at akronymet (forkortelsen) inngår i betydningen – det refererer til seg selv. Dermed kan vi si at *PHP* står for *PHP: Hypertext Preprocessor.*[1]

Mange frivillige har vært aktivt med i prosessen med å utvikle språket, blant annet kan Stig Sæther Bakken fra Trondheim nevnes. Hans navn finner du i den offisielle PHP-manualen på Internett, under adressen

*http://www.php.net/manual/*

Interessen for PHP blir stadig større, og det kommer jevnlig nye, forbedrede utgaver av språket. PHP modnet gjennom versjon 3. Versjon 4 satte søkelyset på sikkerhet. Versjon 5 kom sommeren 2004, og stadig nye versjoner er under utvikling.

PHP er et naturlig valg fordi det kan brukes av både nybegynneren og den avanserte bruker. PHP har nå fått fullstendig støtte for objektorientering, til glede for de som måtte ønske å programmere PHP objektorientert (OOP) fremfor proseduralt. PHP har også en rekke finesser. Disse avanserte mulighetene blir introdusert i denne boka, men merk deg at hovedvekten ligger på generell, basis PHP-programmering. Det er en rekke grunner til at språket er populært:

- *For alle*: PHP er så enkelt at nybegynnere kan komme i gang uten å ha erfaring i programmering fra før av. Muligheten for å blande HTML og PHP-kode gjør at du gradvis kan legge til dynamikk i dine eksisterende Internett-sider, uten å måtte starte på begynnelsen. Syntaksen likner på den i Java, C og Perl, hvilket gjør at det er ekstra lett å lære PHP for de som kan programmering fra før av.
- *Utbredelse*: En hel rekke Internett-leverandører har støtte for PHP. I tillegg til mange hobbyløsninger lagd med PHP, er mange kjente, store nettsteder lagd i PHP. Et nettsted som har lenker som slutter med adressen ***.php***, bruker PHP som teknologi. Stadig flere tar i bruk PHP, og det er derfor ekstra verdifullt for deg å lære å mestre språket.
- *Fleksibel koding*: Du kan enten skrive tradisjonell prosedural kode eller velge en objektorientert tilnærming. I tillegg kan de to fremgangsmåtene blandes,

1. Et annet kjent akronym er *GNU,* som står for *GNU's Not UNIX*.

noe som av og til er hendig. Med PHP versjon 5 er støtten for OOP fullstendig. Denne boka ser i utgangspunktet på vanlig prosedural koding, men tar også opp den objektorienterte muligheten i kapittel 14. Skulle du allerede ha programmert løsninger i Java, kan disse gjenbrukes siden PHP tilbyr funksjonalitet som bruker Java-objekter som om de var PHP-objekter.

- *Hurtighet*: Løsninger bygd på PHP utføres raskt og stabilt og krever lite ressurser på tjenermaskinen.
- *Sikkerhet*: PHP kan settes opp til å tilfredsstille ulike grader av sikkerhet. Dette er et ikke-trivielt tema. Sikkerhetsaspekter med programmeringen vil bli tatt hensyn til og diskutert i mange av eksemplene i boka. Kapittel 12 har en mer grundig gjennomgang av datasikkerhet, og tar opp problematikken rundt og teknikker for å unngå hacking, adgangsbegrensning, og mye mer.
- *Avansert, men enkelt*: Det kommer stadig nye tilleggsmoduler/biblioteker som gjør det mulig å lage for eksempel grafikk, PDF-filer og Flash-filmsnutter dynamisk. Du kan lage handlekurvsystemer, spill, fotoalbum der du presenterer bildene dine på flotte måter, bursdagssystemer, spørrekonkurranser, gjestebøker, distribusjonssystemer, ditt eget nyhetsforum, søkemotorer, oversettelsesprogrammer og så videre. Bare fantasien setter grenser. Fellesnevneren er at selv avansert funksjonalitet blir enkelt når du kan det grunnleggende, noe som skyldes fine eksempler på web og en rekke innebygde funksjoner i PHP.
- *Plattformuavhengighet*: De scriptene du lager, må kjøres av PHP, som igjen må være installert på en webtjener. PHP fungerer med de fleste operativsystemer, både Windows, Mac, Linux og mange varianter av UNIX. Webtjenere som kan benyttes, er Apache, Internet Information Server (IIS) fra Microsoft, Personal Web Server, Oreilly Website Pro-server, Caudium, Xitami, OmniHTTPd og mange flere. Du har altså mulighet til fritt å velge både operativsystem og webtjener. Det er også godt å vite at du som utvikler kan lage løsninger som fungerer for alle og er portable mellom plattformer.
- *Åpen kildekode*: Mange tusen utviklere bidrar til å lage forbedrede versjoner av PHP og moduler for utvidet funksjonalitet. Det betyr at PHP er gratis å bruke for alle parter.
- *Informasjon*: PHP har støtte for alle store databasesystemer og god XML-støtte. Boka tar for seg bruk av databasesystemet MySQL og koder i henhold til API-et som heter mysqli. Forskjellene mellom ulike databasesystemer er relativt små i PHP, så det du lærer, kan overføres ved behov.
- *Tjenester*: PHP støtter kjente protokoller så som IMAP, NNTP, HTTP, LDAP, COM og så videre. Dette er ansett for å være mer avansert programmering, som vi ikke går gjennom i denne boka.
- *Hjelp*: Denne læreboka er et godt utgangspunkt for å lære PHP, men av og til vil du kunne trenge annen hjelp. Det fins per i dag utallige portaler på Internett, e-postlister, nyhetsgrupper/diskusjonsfora og samlinger av kodeeksempler hvor du kan søke hjelp.
- *Samme prinsipp som ASP.NET og JSP*: Når du kan programmere i et av de tre store språkene ASP.NET, JSP eller PHP, kan du enkelt overføre dine kunnska-

per til et annet språk. Hvilket av de tre store du bør velge som førstespråk, er litt avhengig av din bakgrunn, dine interesser og dine fremtidsplaner. PHP er trolig enklest å komme i gang med, og byr dessuten på veldig avanserte muligheter. De foregående punktene bør ikke levne tvil om at du har gjort et godt valg!

Oppsummering: Det er mange fordeler med PHP, ikke bare at det er gratis i bruk. Med PHP kan du lage solide, sikre løsninger som fungerer raskt for tusenvis av samtidige brukere. Det er enkelt å komme i gang, og samtidig har språket mange avanserte muligheter du som webutvikler kan benytte til å lage det du ønsker av webløsninger. PHP er i kontinuerlig vekst og blir stadig mer utbredt. Fleksibiliteten er ivaretatt i og med at alle store operativsystemer og webtjenere støttes, i tillegg til at både prosedural og objektorientert tilnærming kan brukes av den som programmerer løsningene.

## 1.2 Hvordan fungerer PHP?

Alle PHP-script må kjøres på en webtjener. Resultatet av kjøringen sendes til den besøkendes nettleser for visning. For å kunne lage dine egne script er det derfor nødvendig at du har tilgang til en *webtjener* som har installert *PHP-programvare*. Dersom du bruker en ISP (Internet Service Provider, eller Internett-leverandør på godt norsk), vil trolig denne kunne gi deg muligheten til å laste opp og kjøre PHP-filer på sine tjenermaskiner. Du kan også installere en webtjener lokalt på din egen maskin.

Dette kapittelet starter med å forklare hvor enkelt installasjon av PHP er ved bruk av ferdige pakkeløsninger. Likevel er det nødvendig å bruke tid i starten på å forklare hvordan PHP fungerer generelt, og litt om filene ***php.ini*** og ***httpd.conf*** spesielt. Dersom du har installert det du trenger og vil gå rett til programmeringen (delkapittel 1.3), så gjør gjerne det, men husk å vende tilbake hit for å få en dypere forståelse av hvordan PHP fungerer og hvorfor.

### 1.2.1 Overordnet om det å programmere i PHP

PHP-programvare må være installert på en webtjener for at tjeneren skal kunne kjøre PHP-script. Når denne forutsetningen er på plass, kan du utvikle webløsninger i PHP. Legg merke til at du kan installere en webtjener på din egen datamaskin, enten du har Windows, Linux eller Mac.

Dersom du legger ut og tester sidene du lager hos en ISP, må du overføre alle endringer du gjør, for eksempel ved å flytte filer med FTP. Dette må gjøres for hver gang du skal teste løsningen din på nytt. Mange tester om ting fungerer, nesten for hver linje med programkode de lager, og da kan slik testing bli tungvint. Et godt alternativ (eller egentlig supplement) er å installere en webtjener og den nødvendige PHP-programvaren lokalt på din egen maskin. Du vil da måtte bruke som adresse (URL) *http://localhost/enSide.php* for å teste scriptene dine. Når løsningen

er fullstendig ferdig (eller når du ønsker det), kan den publiseres på Internett ved å overføre alle filene til riktig ISP. Slik trenger du ikke å ha tilgang til Internett hver gang du skal jobbe med PHP. Samtidig sparer du mye tid siden du unngår filoverføring, og slipper å bekymre deg for om riktig versjon av filene er lastet opp (noe som ofte kan være en øvelse hvor tunga må holdes temmelig rett i munnen).

En annen fordel med å utvikle lokalt og publisere globalt på et senere tidspunkt, er at du da viser «alt-eller-ingenting» til dine besøkende. Det er uheldig fra et sikkerhetsmessig perspektiv å la andre se at sidene dine blir gradvis utviklet. En hacker som får overvåke utviklingsprosessen, vil nemlig kunne avsløre mange særtrekk ved oppbygningen av nettstedet som senere kan brukes mot deg. Brukere som får ufullstendig funksjonalitet, blir også misfornøyde, og data de eventuelt prøver å registrere, kan gå tapt.

**Hva må til for å programmere i PHP?**

I praksis innebærer det å programmere i PHP å gjenta følgende prosess inntil løsningen er ferdig:

- Skrive/endre/videreutvikle kode i en teksteditor.
- Lagre filen(e) med filendelse ***.php*** på riktig sted.
- Teste de siste endringene ved å skrive inn riktig URL i adressefeltet i nettleseren, på formen *http://www.dinISP/enSide.php*. Dersom du bruker din egen maskin under utvikling, brukes *http://localhost/enSide.php*.
- Søke hjelp i lærebøker (for eksempel denne), den offisielle PHP-manualen eller andre steder for å løse konkrete problemstillinger og utvikle deg programmeringsteknisk.

Du kan velge mellom flere webtjenere når du jobber med PHP. *Apache* er mest utbredt og enkel i bruk. Legg merke til at PHP også fungerer med IIS fra Microsoft og mange andre tjenerløsninger. Hvis du allerede har tilgang til PHP og en webtjener, kan du gå til avsnitt 1.2.3 for å få en bedre forståelse av hvordan programvaren oppfører seg (noe som er viktig for å forstå PHP-programmering best mulig).

Hvis du ikke har installert det du trenger for å kjøre PHP, så har du to valg:

- Installere alt du trenger manuelt.
- Installere en pakkeløsning som består av komponentene PHP, Apache og MySQL (samt *phpMyAdmin*).

### 1.2.2 Installasjon

Tidligere utgaver av denne boka (første og andre utgave) la vekt på manuell installasjon. Tanken var å gi innsikt i samspillet mellom PHP, Apache og MySQL på en

best mulig måte. En får unektelig en dypere forståelse av og et optimalt grunnlag for å programmere i PHP gjennom å installere alt en trenger manuelt, siden det krever at en konfigurerer delene og syr dem sammen. På den annen side er faren for feil stor, og en kan bruke mye tid før en kommer i gang med det en faktisk skal, nemlig å lære å programmere i PHP. I tillegg kommer det stadig nye utgaver av operativsystemer og programvare. En oppskrift i ei bok vil da fort bli utdatert og skape mer frustrasjon enn glede hos leseren.

Denne boka beskriver derfor ikke manuell installasjon, men anbefaler heller ferdige pakkeløsninger. Du vil uansett få kjennskap til de underliggende teknologiene og samspillet mellom dem jevnlig utover i boka. Dersom du vil, kan du fortsatt installere PHP manuelt, men må da ty til oppskrifter som du finner på Internett. Det som stod i tidligere utgaver av denne boka om manuell installasjon, vil du også finne på ressurssiden (*http://phpbok.no*).

Neste ramme lister opp noen alternativer for å installere nødvendig programvare ved hjelp av totale pakkeløsninger på ulike plattformer. Dersom du skulle ha for eksempel Apache installert fra før av, kan det være lurt å avinstallere denne før du installerer en pakkeløsning.

### Pakkeløsninger for å installere PHP

Kombinasjonen av PHP, Apache og MySQL er veldig populær og anbefales, også på Windows. Disse komponentene spiller godt sammen, og databasesystemet MySQL er etter hvert blitt veldig modent, med mye avansert funksjonalitet. De fleste pakkeløsninger består derfor av disse tre og syr dem sammen slik at du kan installere alt du trenger med noen få klikk.

Windows-brukere kan med fordel benytte en pakke som heter WAMP. Forkortelsen står for *Windows, Apache, MySQL and PHP*, og kan lastes ned fra adressen *http://www.wampserver.com/en/* (eller søk på WAMP i din favorittsøkemotor). En annen populær pakkeløsning er XAMPP, tilgjengelig fra adressen *http://www.apachefriends.org/en/index.html*.

Mac-brukere kan med fordel bruke en pakke som heter MAMP. Forkortelsen står for *Mac, Apache, MySQL and PHP*, og kan lastes ned fra adressen *http://www.mamp.info/* (eller søk på MAMP). Vi nevner også at Mac OSX faktisk kommer med PHP ferdig installert, så i utgangspunktet trenger en bare å aktivere PHP. Du bør likevel vurdere en pakkeløsning (se avsnitt 1.2.3).

WAMP og MAMP er fine navn, men det er egentlig Linux som har æren. Det er nemlig kombinasjonen Linux, Apache, MySQL og PHP (også kalt LAMP) som tradisjonelt har vært brukt på webtjenere. Mange Linux-distribusjoner kommer nå med alt en trenger ferdig installert, men pakkeløsninger fins også.

Det fins en rekke pakkeløsninger, og det kan også være at noen veletablerte alternativer forsvinner. Søk i så fall på Internett på for eksempel *Install Apache MySQL PHP,* og du vil trolig få mange gode alternativer.

Merk at ressurssiden (*http://phpbok.no*) har video og tekst som er nyttig, enten du velger installasjon ved hjelp av en pakkeløsning eller manuell installasjon.

### 1.2.3 Forstå hvorfor PHP fungerer som det gjør

For at PHP-scriptene dine skal fungere når du utvikler og tester dem ut lokalt, må webtjeneren (typisk Apache) kjøre. Webtjeneren er en tjeneste som må startes i operativsystemet, og den håndterer alle forespørsler som kommer mot en viss port (normalt port 80). I tillegg må MySQL-tjeneren kjøre (dersom PHP-scriptene du lager skal benytte informasjon i en database) fordi det er MySQL som håndterer og administrerer tilgangen til informasjonen i databasen(e). Det er mulig å la en webtjener og en databasetjener kjøre hele tiden på din lokale maskin, men du kan også starte disse manuelt, noe som egentlig er fordelaktig, fordi du da sikrer deg at disse tjenestene kjører bare når det er behov for det.

Pakkeløsninger som WAMP eller MAMP tilbyr deg å starte og stoppe webtjeneren på en enkel måte. Når du har flere tjenester, er dette fordelaktig. Du velger rett og slett bare å starte alle tjenester fra et menyvalg i selve pakkeløsningen, og så fungerer alt som det skal, uten at du trenger å navigere deg dypt innover i operativsystemets kontrollpanel (innstillinger). Når du er ferdig, slår du enten av tjenestene fra et menyvalg i pakkeløsningen, eller du avslutter selve pakkeløsningen. Dette er enkelt og trygt, og anbefales.

Når du programmerer i PHP, kan du for eksempel lage kode som tegner grafikk dynamisk. Dette gir deg mulighet til å vise brukeren statistikk basert på innholdet i en database slik det er akkurat nå (noe som dekkes i kapittel 13), men det krever typisk at en viss tilleggsmodul er aktivert først. En stor fordel med pakkeløsninger er at de mest vanlige tilleggsmoduler kommer ferdig klargjort – samtidig som du har mulighet til lett å endre på oppsettet.

Det er noen sentrale tekstfiler, også kalt konfigurasjonsfiler, som styrer hvordan PHP og Apache oppfører seg. PHP styres i hovedsak av filen ***php.ini***. For at for eksempel dynamisk generering av grafikk skal fungere, må det henvises til et grafikkbibliotek i ***php.ini***. Hvis du skal se feilmeldinger når du programmerer, må en linje med teksten `display_errors = On` stå i filen ***php.ini***. Dersom denne linjen ikke står i ***php.ini***, vil du ikke se feilmeldinger når du programmerer.

Hvis du installerer PHP manuelt, må du gjøre alle endringer ved å redigere direkte i ***php.ini*** (ved å åpne denne i en teksteditor). Pakkeløsninger, derimot, tilbyr deg å gjøre endringer ved å krysse av i en dialogboks eller gjøre et menyvalg. Det som egentlig skjer, er at pakkeløsningen endrer ***php.ini*** for deg når du gjør endringer i menyvalg. Grunnen til at dette kan være gunstig, er at faren for feil er større om du selv åpner ***php.ini*** manuelt. Du kan egentlig se på ***php.ini*** som en fil som har innstillingene for hvordan PHP skal oppføre seg. Den beste måten å forklare hvordan pakkeløsningene fungerer på dette området, er gjennom demonstrasjon. Se derfor på videofilmene som du finner på bokas ressursside (*http://phpbok.no*).

Hvis du tenker deg om, har de fleste programmer en rekke innstillinger du kan gjøre (egenskaper). Du er derimot trolig vant til å stille inn på slike egenskaper ved hjelp av en egenskapsboks som aksesseres fra menyvalg (fra Windows/Mac og grafiske brukergrensesnitt under Linux). I praksis endres bare en konfigurasjonsfil (eller liknende) som du ikke ser. Det er flere grunner til at PHP har en såpass synlig konfigurasjonsfil, og at det er verdt å vite om ***php.ini*** sin eksistens for den som skal lære PHP:

– For det første er det en rekke veiledere på Internett som omtaler ***php.ini*** og beskriver hvilke innstillinger som må gjøres, og andre konfigurasjonsmessige forutsetninger for å kunne programmere en bestemt løsning.
– For det andre vil innstillingene påvirke hvordan du programmerer, for eksempel hvorvidt du ser feilmeldinger eller ikke. Det er også en rekke innstillinger som får konsekvenser for sikkerhet.
– For det tredje er det ofte slik at ferdige PHP-løsninger ligger på en webtjener på Internett. Dersom du har utviklet løsningen din lokalt og brukt en pakkeløsning, har du trolig brukt de innstillingene som kom med pakkeløsningen. Det er vel og bra, men hva om webtjeneren (på Internett) har et annet oppsett, og for eksempel ikke har konfigurert for bruk av grafikkbiblioteket? Da vil all kode der du prøver å tegne grafikk dynamisk, slå feil, og brukere vil ikke se den fine statistikken du har brukt tid på å lage. Du må i så fall ta kontakt med systemansvarlig og be denne endre på innstillingene i ***php.ini***. Mange webtjenere kjører Unix eller Linux, og systemansvarlige er vant til å endre konfigurasjonsfiler via en teksteditor. Dersom alt fungerer lokalt på din maskin, men ikke på web, vil systemutviklere typisk gå rett til ***php.ini*** og se på den. Du kan i så fall lokalisere din ***php.ini*** og sende den til systemansvarlig, som kan sammenlikne med webtjenerens fil.
– Sist, men ikke minst, er det mulig å overkjøre innstillingene i ***php.ini*** ved å plassere en egen mini-konfigurasjonsfil i for eksempel samme mappe som PHP-scriptene du har lagd. Dette kan også brukes som strategi for å løse problemene nevnt i forrige punkt. Hvis du bruker Apache som webtjener, er det en fil som heter ***.htaccess*** (med punktum først i filnavnet), som må ha slike innstillinger.

Det er også viktig å nevne at selve innstillinger i en eller flere konfigurasjonsfiler bestemmer hvordan selve webtjeneren oppfører seg. For Apache sitt vedkommende er det i hovedsak filen ***httpd.conf*** som er aktuell å kjenne til, og siden Apache er så utbredt som webtjener, diskuteres den her. Pakkeløsninger tillater deg å endre på innstillinger. Du bør se ressurssiden til boka på http://phpbok.no for tekstlig og videobasert informasjon om dette. Her er likevel en oppsummering av de viktigste elementene du bør kjenne til for ***httpd.conf***:

– Grunnen til at Apache snakker med PHP, er at det ligger informasjon i ***httpd.conf*** om at PHP skal lenkes sammen med Apache ved oppstart. I tillegg har ***httpd.conf*** informasjon om at filnavn som slutter på ***.php***, skal behandles av PHP-programvaren. Det betyr at PHP vil få beskjed om å kjøre alle forespørsler som Apache mottar som går til en fil med ***ettellerannetnavn.php***.

– Hvor skal scriptene dine lagres? I hvilken mappe (katalog) på maskinen må du legge alt du lager i PHP? Dette er viktig, fordi når du i nettleseren skriver inn en adresse, må denne kobles opp mot riktig mappe på maskinen din slik at filen blir funnet. For å være helt nøyaktig er det innstillingen i en linje som starter med `DocumentRoot` i filen ***httpd.conf*** som bestemmer koblingen. Pakkeløsninger angir gjerne denne koblingen automatisk, for eksempel til mappen ***C:\wamp\www***. Da må du legge PHP-scriptene dine i denne mappen og skrive *http://localhost/filnavn.php* for å kjøre PHP-scriptet som heter ***filnavn.php***. Vær derimot varsom med å endre `DocumentRoot` med mindre pakkeløsningen tilbyr menyvalg for det. Endringer krever uansett restart av Apache før de trer i kraft.

Legg deg på minnet at standardinnstillingene i ***php.ini*** og ***httpd.conf*** kan være mangelfulle og hindre deg i å gjøre det du ønsker. Endringer i disse kan løse problemer. Husk derfor på disse to filenes rolle for PHP sin virkemåte!

### 1.2.4 Test om alt virker

Du er nå klar til å teste om alt fungerer som det skal, slik at du kan begynne med virkelig kreativ PHP-programmering. Lag en helt ny fil som har innholdet:

```
<?php
   phpinfo();
?>
```

og lagre filen som ***informasjon.php*** i mappen (katalogen) som står oppført som `DocumentRoot` i ***httpd.conf***. Dersom du bruker den ferdige pakkeløsningen WAMP (Windows), så skal du legge filen i den mappen du kan se fra menyvalget *www directory*. Bruk adressen *http://localhost/informasjon.php*.

Dersom du bruker MAMP (Mac), så kan du selv angi hva som er DocumentRoot fra en innstilling (Preferences). Du må legge filen din i denne mappen. Som standard bruker MAMP port 8888, så du må derfor (om du ikke stiller inn annerledes) bruke adressen *http://localhost:8888/informasjon.php*.

Dersom du har lastet opp filene på en webtjener, må du oppgi adressen til denne tjeneren. De som studerer data ved Høgskolen i Sør-Trøndelag (HiST), får eksempelvis et eget underområde under webtjeneren, og det samme er vanlig for mange andre skoler og høgskoler. En HiST-student med brukernavn *lisenordmann* vil typisk måtte bruke *http://www.stud.aitel.hist.no/~lisenordmann/informasjon.php*.

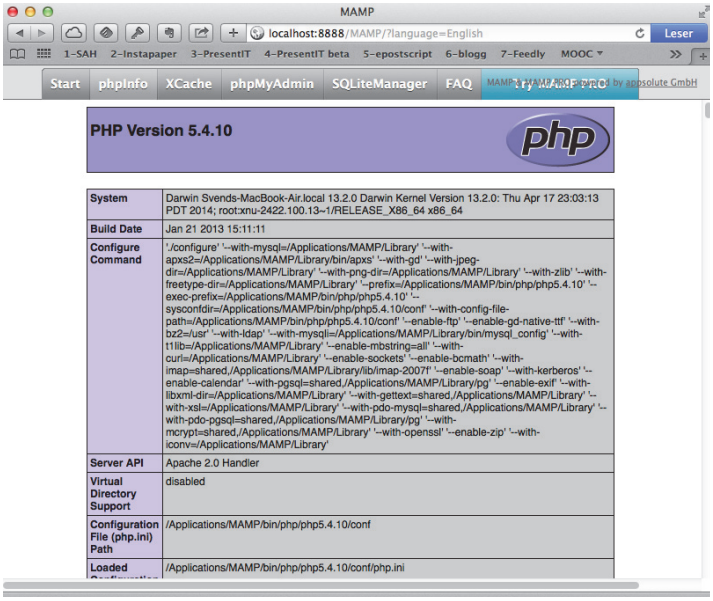
Når du i nettleseren skriver inn adressen

```
http://startForDinDocumentRoot/informasjon.php
```

der *startForDinDocumentRoot* kan være *localhost*, *localhost:8888* eller *www.stud.aitel.hist.no/~lisenordmann*, vil Apache kunne behandle forespørselen. Apache ser at ***informasjon.php*** har endelsen ***.php*** og følgelig skal behandles av PHP (ifølge ***httpd.conf***). Derfor gis kommandoen over til PHP, som kjører scriptet.

Mer presist kan vi si at PHP åpner filen, ser på innholdet og utfører alle kommandoene i den.

Resultatet som vises i nettleseren ved kjøring av ***informasjon.php***, ser du i figur 1.1. Der vises en omfattende oversikt over innstillingene på tabellform som gjelder for PHP-versjonen som er installert. Kommandoen `php_info();` gjør nemlig at PHP lister ut denne informasjonen. Du vil kunne kjenne igjen mange innstillinger fra filen ***php.ini***. Trolig er `display_errors` påslått, men hvis ikke er det veldig lurt å sørge for at den blir påslått ved å enten endre ***php.ini*** eller velge fra menyene i pakkeløsningen du bruker. Du vil også finne informasjon om Apache, både portinformasjon, hvor roten til webtjeneren befinner seg, og hvilken versjon av Apache som kjører.




Figur 1.1 Scriptet kjøres av PHP, siden både PHP og Apache er satt riktig opp.

Vi kommer tilbake til ***php.ini*** jevnlig utover i boka. Koden som vi nettopp har skrevet inn og testet, tilbyr en enkel måte for å sjekke at alt fungerer som det skal, men utover det bør du ikke plassere slik kode på nettstedet ditt. Det er nemlig ikke gunstig at brukere av dine nettsider får vite hvilke innstillinger PHP er satt opp med!

Du kan skrive hvilken som helst PHP-kode i dine PHP-script, og denne boka handler nettopp om hvilken kode som må til for å løse helt konkrete problemer, som for eksempel å lage en nettbutikkløsning, et registreringssystem og liknende. Etter hvert får du mange script, og da er det lurt å organisere disse i undermapper. Hvis du under webroten har en mappe som heter ***morsomt/*** og en undermappe som heter ***filmer/*** og du i denne plasserer en fil som heter ***liste.php***, så vil adressen du må skrive inn som URL, bli

```
http://startForDinDocumentRoot/morsomt/filmer/liste.php
```

Nå er det på tide å starte programmeringen for fullt!

## 1.3 Samspillet mellom klient og tjener

HTML står for Hyper Text Markup Language og har hatt en enorm betydning for veksten av Internett. Dette skyldes at HTML er enkelt å lære og logisk oppbygd. «Markup» betyr at formateringen av elementer angis ved å markere disse med spesielle *tagger* som nettleseren forstår. Med HTML kan Internett-sider med grafikk, lyd, tekst og andre elementer lenkes sammen. Når slike sider lagres på en tjenermaskin, blir de tilgjengelige for hele verden. Bruk av HTML passer bra til statisk informasjon som ikke endres ofte.

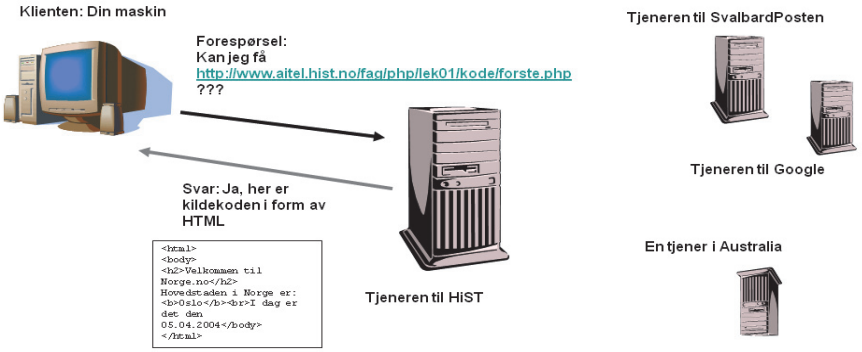
HTML spesifiserer altså *hvordan* informasjonen skal presenteres. Utvidet funksjonalitet, dynamikk og interaksjon med brukeren krever imidlertid programmering. PHP bygger på to ting: at du forstår HTML, og at du forstår programmering. Denne boka skal øve opp programmeringsforståelsen og vise hvordan PHP kan brukes til å lage fine webløsninger. Dersom du ikke allerede kan grunnleggende HTML, er det lurt å gjennomgå det først. Ressurssiden til boka (*http://phpbok.no*) har tips til gode websider.

PHP og HTML er ikke konkurrenter. PHP samhandler med HTML og bruker HTML. I denne boka benyttes for øvrig en utvidet HTML-syntaks som heter XHTML, fordi det gir bedre kode som lettere kan brukes av ulike systemer og inngå i automatiseringsprosesser.

### 1.3.1 Klienten ber en tjener om å sende informasjon

*Klienten* er som regel den datamaskinen du (eller andre) benytter ved surfing på Internett. *Tjeneren* (ofte brukes det engelske navnet *server*) er en datamaskin med spesiell programvare som er plassert hos for eksempel Høgskolen i Sør-Trøndelag (HiST). Tjeneren har den egenskapen at den har lagret informasjon som kan deles med andre. På nettet er det mange flere klienter enn tjenere.

Har du noen gang tenkt over hva som skjer når du besøker et nettsted? Du skriver enten inn en såkalt URL (Uniform Resource Locator) i nettleserens adressefelt eller klikker på en lenke (hyperreferanse). Det er nå opp til nettleseren å ta kontakt med den tjeneren hvor riktig nettsted ligger lagret, og be om å få tilsendt ønsket side. Vi sier at nettleseren på klienten sender en *forespørsel* til tjeneren, som så vil behandle forespørselen og returnere riktig informasjon. Til slutt vil nettleseren kunne vise innholdet for deg.




Figur 1.2 Basert på innskrevet URL vil klienten ta kontakt med riktig tjener på Internett, som forhåpentligvis kan returnere ønsket side.

### 1.3.2 Et enkelt PHP-script

Et PHP-script kan skrives i en teksteditor så som «Notisblokk», «Textpad», «FrontPage», «Dreamweaver», «vi», «pico» og liknende. Microsoft Word skal aldri brukes til å programmere websider. Du vil finne lenker til flere gratis editorer på bokas hjemmeside.

Egentlig er det et skille mellom et script og et program, men i denne boka (og stort sett ellers i websammenheng) er dette skillet uvesentlig, og begrepene vil derfor brukes om hverandre. Et script vil tolkes linje for linje fra start til slutt. Hver gang scriptet skal kjøres, må det tolkes på nytt. PHP er et scriptspråk hvor koden blir tolket hver gang den kjøres. Det betyr at koden kan produsere et resultat som kan vises selv om det skulle være feil i koden, i motsetning til kompilert kode.

Koden skal lagres i en vanlig fil med etternavnet ***.php***. I Windows velger du «Lagre som ...» i din editor. Dersom du ikke kan velge php som filendelse (dette er for eksempel ikke blant valgene i programmet Notisblokk), velger du «Alle filer» fra nedtrekksmenyen og skriver inn ønsket filnavn og etternavnet ***.php***. (Statiske HTML-sider har vanligvis ***.html*** eller ***.htm*** som etternavn, dette gjelder også om XHTML brukes.)

Neste script skal skrive ut (det vil si "vise") navnet på hovedstaden vår i en nettleser. Hvis du vil prøve selv, kan du skrive inn den etterfølgende koden i en ny fil i den teksteditoren du bruker, og lagre denne filen som ***forste.php***. Om du vil plassere disse i en passende mappe, så sørg for at den er en undermappe av det som er satt som `DocumentRoot`. Vi forklarer først samspillet mellom tjener og klient, og deretter syntaksen i koden. Dersom du ikke klarer å kjøre dette scriptet, kan det skyldes at PHP ikke er installert på din maskin. Du må i så fall installere nødvendig programvare (se avsnitt 1.2.2).

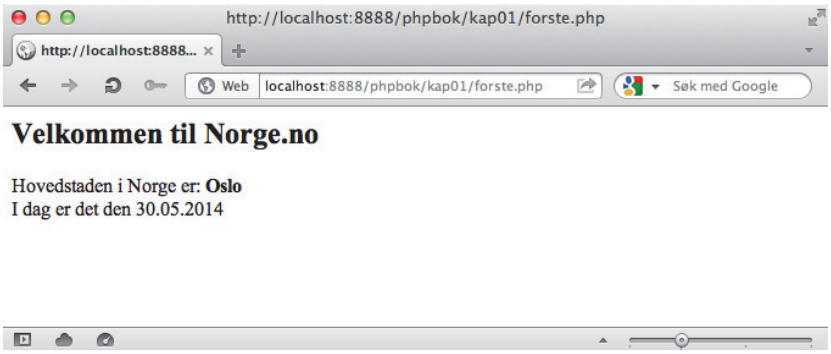
**Kodesnutt 1.1 I scriptet forste.php er HTML og PHP blandet sammen**

```
<html>
<body>
<h2>Velkommen til Norge.no</h2>
<?php
  echo "<p>Hovedstaden i Norge er: ";
  echo "<strong>Oslo</strong>";
  echo "<br />";
?>

I dag er det den <?php echo date("d.m.Y");?>
</p>
</body>
</html>
```

### 1.3.3 Resultatet vises i nettleseren

Legg merke til at HTML-taggene `<strong>` og `</strong>` forekommer inne i PHP-koden. Dersom du lagrer filen som beskrevet og skriver inn riktig adresse i nettleseren din, vil du få frem følgende resultat:




Figur 1.3 Resultatet av scriptet i kodesnutt 1.1.

En titt på kildekoden bak denne Internett-siden (Velg «View Source» eller liknende funksjonalitet fra nettleseren, gjerne fra en kontekstsensitiv meny) viser at nettleseren bare har behandlet vanlig HTML (egentlig XHTML). Selv om en PHP-side etterspørres i nettleseren, vises altså bare vanlig HTML-kode. Hvor er det blitt av PHP-koden? Dessuten brytes ikke linjene i kildekoden fra nettleseren selv om vi lagde linjeskift i PHP-scriptet vårt. Hva har skjedd?


Figur 1.4 Kildekoden avslører at bare HTML ble sendt tilbake fra tjeneren.

### 1.3.4 Tjeneren sender HTML-kode til klienten

Scriptet fra kodesnutt 1.1 befinner seg på en lokal maskin, og kan derfor nås på adressen

*http://localhost/forste.php*

Dersom scriptet hadde vært lagt på for eksempel AITeL sin tjener hos HiST i Norge, i et fag som heter PHP, ville adressen kunne vært slik:

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

Klienten ber uansett tjenermaskinen som ligger bak adressen, om å utføre koden til filen ***forste.php***.

Hemmeligheten til at koden *utføres på tjeneren som et script*, ligger som tidligere nevnt i etternavnet ***.php***. Filer med etternavn ***.html*** vil returneres direkte slik de er, mens alle PHP-script vil prosesseres før resultatet returneres. Etter at koden er utført enten lokalt, på HiST sin tjener eller tilsvarende, vil altså resultatet sendes tilbake til klienten i form av vanlig HTML, og det er dette vi får frem på skjermen i nettleseren. Det er viktig å forstå at det er *resultatet*, og ikke selve scriptet, som returneres.

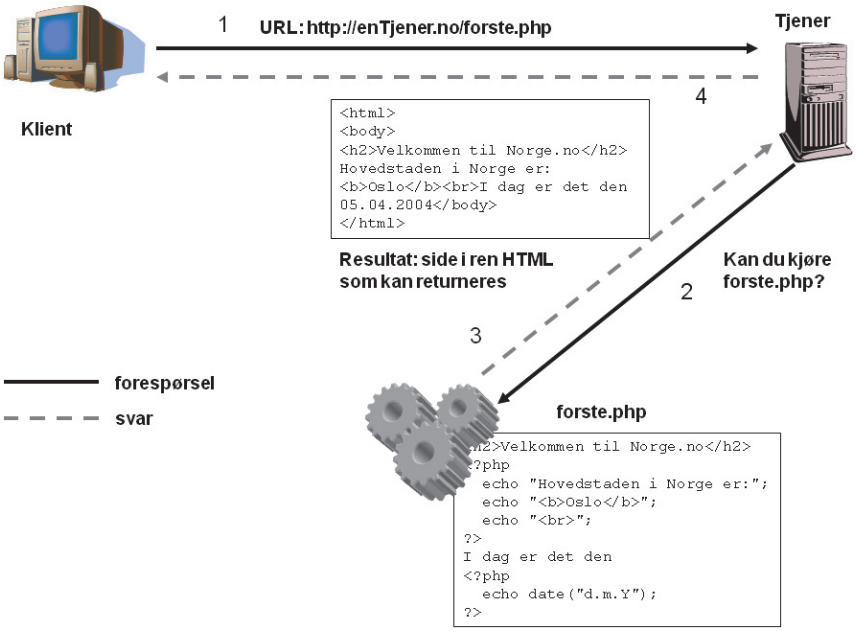
Den observante leser aner kanskje allerede nå at nettopp her ligger nøkkelen til dynamikk: Dersom tjeneren kjører programmet rett etter at en person har besøkt siden, og resultatet vises umiddelbart deretter, er det jo mulig å programmere Internett-sider slik at hver enkelt bruker får forskjellig side presentert på skjermen basert på visse kriterier som angitt i programmet! Ja, dette er riktig, og PHP har i tillegg mange andre interessante bruksområder, som vi skal se utover i boka.

Det kan være nyttig bevisst å tenke på samspillet mellom klient og tjener når en programmerer. I gjennomgangen av installasjonen ble det påpekt at en form for PHP-programvare må være installert på tjeneren, og at tjeneren, for eksempel Apache, vil videresende forespørsler etter PHP-script til PHP-programvaren (egentlig PHP-tolkeren, eller bare *tolkeren*, som vi ofte vil si utover i boka).

Først sendes en forespørsel etter siden ***forste.php*** på riktig tjener (steg 1). Tjeneren mottar forespørselen og ser at dette er en oppgave for PHP-tolkeren (basert på for eksempel innstillingene i ***httpd.conf*** hvis Apache brukes). Tolkeren kjører scriptet

(steg 2). Resultatet av kjøringen samles i en strøm av HTML (steg 3), som tjeneren sender tilbake til klienten (steg 4). På grunn av at all prosessering skjer på tjeneren, trenger ikke nettleseren å forstå noe annet enn vanlig HTML og kan uten problem vise siden fra figur 1.3. Sett med nettleserens øyne er dette statisk informasjon, men brukeren vil oppleve siden som høyst dynamisk i og med at dagens dato vises.

Teorien vi har gått igjennom til nå, er summert opp i figur 1.5.




Figur 1.5 Samspillet mellom klient og tjener.

Hvis du bruker localhost, blir prinsippene for kommunikasjonen likedan. Forskjellen er at både klienten og tjeneren ligger på én og samme maskin. Det er derfor det er nødvendig å starte opp riktig programvare (for eksempel Apache og PHP) før du tester dine PHP-script.

### 1.3.5 PHP og HTML kan blandes

En forklaring på scriptet ***forste.php*** er på sin plass. Koden i kodesnutt 1.1 så slik ut:

```
<html>
<body>
<h2>Velkommen til Norge.no</h2>
<?php
  echo "<p>Hovedstaden i Norge er: ";
  echo "<strong>Oslo</strong>";
```

```
  echo "<br />";
?>

I dag er det den <?php echo date("d.m.Y");?>
</p>
</body>
</html>
```

Først i scriptet kommer helt vanlig HTML-kode, og dette gir akkurat samme resultat tilbake. Når tolkeren støter på taggen `<?php` vet den at alt som forekommer inntil `?>` er PHP-kode. Merk at det er vanlig å rykke inn koden med tabulator eller mellomrom for å få bedre oversikt, men at dette ikke er noe krav.

Den første setningen etter `<?php` starter med kodeordet `echo` og utgjør scriptets første *instruksjon*, eller *linje*, som vi også sier. Hver instruksjon i PHP avsluttes av et semikolon, og dermed ser vi at scriptet totalt sett består av fire instruksjoner mellom HTML-koden.

Det som står omsluttet av anførselstegn, er tekst. Tekst som står etter `echo` blir skrevet ut, og det betyr at PHP i første instruksjon skriver ut teksten `"<p>Hovedstaden i Norge er: "`, der både kolon og mellomrommet etter kolonet er en del av tekststrengen.

Neste instruksjon skriver ut teksten `"<strong>Oslo</strong>"`, hvilket vi gjenkjenner som HTML-taggene for å lage fet skrift. Her er essensen i samspillet mellom PHP og HTML: Når tolkeren er ferdig med å prosessere scriptet den har mottatt, vil siden som er klar for retur, bestå av ren HTML (tekst og tagger), og elementet innrammet i taggen `<strong>` vil deretter føre til at klienten korrekt viser fet skrift i nettleseren.

Det er enda en detalj som er verdt å merke seg. Selv om det er linjeskift i koden, blir det ikke linjeskift mellom `"Hovedstaden i Norge er: "` og `"Oslo"` når resultatet skal vises i nettleseren. Grunnen til dette er at filen ikke tolkes som en tekstfil med skjulte linjeskift, men som en HTML-fil hvor linjeskift må markeres eksplisitt med for eksempel taggen `<br />`. I eksempelet vårt har vi forutsatt at dette ville bli et problem mellom Oslo og teksten `"I dag ..."`.

Når tolkeren møter på taggen `?>` er den ferdig med å utføre PHP og hopper tilbake til HTML-modus. Linjen `I dag er det den` vil derfor vises som vanlig HTML direkte, men så støter tolkeren på `<?php` og må hoppe tilbake til PHP-modus enda en gang, selv om dette står midt inne i en linje som ikke startet med PHP. Vi kan altså bruke PHP der det passer best. Formularet

```
echo date("d.m.Y");
```

betyr at det som står bak echo, skal skrives ut til skjerm. Hva er det? Funksjonen `date()` returnerer informasjon om dagens dato. Denne er innebygd i PHP, og når `"d.m.Y"` står inne i parentesene, vil resultatet bli at dagens dato på formen dag.måned.år, skrives ut. Se tilbake på figur 1.3 om du er i tvil. Du vil lære mer om datofunksjoner i kapittel 6.

### 1.3.6 Noen detaljer

Følgende tre setninger gir akkurat samme resultat:

```
echo "tekst her";
echo("tekst her");
print("tekst her");
```

Hvilken av de tre du velger å bruke, er en smakssak, og du må gjerne bruke alle i samme kodesnutt.

En fin ting med PHP er at det er så visuelt, noe vi har illustrert med echo-funksjonaliteten. Resultatet vises i nettleseren. Det er imidlertid også mulig å programmere funksjonalitet som ikke gir noe synlig resultat tilbake i nettleseren. Dette er nyttig hvis trafikk om antall besøkende og deres handlinger skal logges. Slik informasjon kan lagres i databaser eller filer og bli nyttig for å legge strategi for videreutvikling av et nettsted. Databaser og filer brukes også for skape interaktivitet og brukervennlighet. Detaljene kommer senere i boka.

Det bør også nevnes at det er mulig å bruke følgende syntaks for å markere hva som er PHP-kode:

```
<script language="php"> PHP-koden kommer her </script>
```

Tidligere var det for øvrig lov å bruke samme tagg som ASP, nemlig `<%` og `%>`, men fra og med PHP 6 er det ikke lov, så det er ingen grunn til å benytte den metoden. Det betyr at du i PHP-script på Internett kan komme over følgende tre måter å gjøre ting på:

```
<?php kode her, standard måte å gjøre det på ?>
<? kode her, dette er kortnotasjonen ?>
<script language="php"> PHP-koden kommer her </script>
```

PHP-tolkeren aksepterer at det veksles mellom HTML- og PHP-modus, og det gir mulighet til å skrive for eksempel

```
<h2>Velkommen hit, <?php echo $navn ?></h2>
```

Følgende hurtigsyntaks er mulig å bruke dersom `<?` er satt opp som gyldig starttag i ***php.ini***:

```
<h2>Velkommen hit, <?= $navn ?></h2>
```

Med andre ord betyr `<?=` og `<?php echo` akkurat det samme. Det er fristende å bruke kortnotasjonen, men du kan risikere at en fremtidig ISP som skal ha Internett-sidene dine, ikke har slått på dette valget i ***php.ini***, og dermed vil ikke scriptene kjøre som de skal.

Vi bruker notasjonen

```
<?php
koden kommer her...
?>
```

gjennomgående i denne boka, med noen få unntak.

### 1.3.7 Kommentarer i PHP

Alt som står på en linje bak `//` oppfattes som en kommentar av tolkeren og vil dermed ikke bli utført. Det betyr at verdifulle kommentarer kan skrives for å dokumentere et script. Kommentarer over flere linjer følger for øvrig samme standard som C, C++ og Java, slik:

```
/*  kommentar over
mange linjer eller midt i en linje,
tolkes ikke av PHP
******!!#**** alle tegn kan brukes, helt til
sluttsekvensen kommer, nemlig */

//Her er en kommentar på bare en linje
echo "Velkommen"; //kommentar etter semikolon er også lov
#Dette er også en kommentar som det er lov å bruke
```

### 1.3.8 Viktig notis om XHTML

Dersom du kan HTML fra før, er endringene til XHTML små, men viktige, og en forklaring er derfor på sin plass. XHTML krever at alle tagger (også kalt elementer) skrives med små bokstaver, og at alle tagger avsluttes.

I tillegg er noen vanlige HTML-tagger som `<b>` og `<i>` erstattet med `<strong>` og `<em>`, og linjeskift som før kunne skrives bare som `<br>`, må skrives som `<br />` for å være gyldig XHTML. Rekkefølgen på tagger har også betydning. Der HTML godtar feil rekkefølge på nøsting av for eksempel

```
<b>en fin <i>dag</b></i>
```

krever XHTML at nøstingen skjer og avsluttes i riktig rekkefølge:

```
<strong>en fin <em>dag</em></strong>
```

Det er altså ikke bare `strong` og `em` som er endret her, men rekkefølgen på hva som avsluttes først og sist.

Mange gamle HTML-sider har brukt `<p>` for å få luft (avstand) mellom avsnitt av tekst. Det samme vil XHTML gjøre, men en må huske å avslutte med en `</p>`. Faktisk må all tekst stå inne i en tagg, og det å plassere tekst utenfor en `<p>` er ikke lov. I forbindelse med CSS (stilsett) er det veldig fordelaktig å bruke de generelle taggene `<div>` og `<span>`. Vi kommer relativt lite inn på CSS i denne boka, men understreker at CSS passer utmerket sammen med XHTML (og PHP).

Det meste av øvrig koding er nokså likt mellom HTML og XHTML. Grunnregelen er å avslutte alle tagger og bruke bare små bokstaver. For å sjekke at kode er gyldig XHTML kan en for eksempel bruke webbaserte validatorer. W3C har lagd en flott en, som du finner på *http://validator.w3.org*.

Boka bruker XHTML. I henhold til standarden skal en side starte slik:

```
<!DOCTYPE bla bla bla (tas ikke med her)>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Tittelen her...</title></head>
<body>... og så videre, innholdet her</body></html>
```

Koden i boka kan fort bli veldig lang, og det er ofte ønskelig å vise bare deler av en webside. Derfor utelates som regel starten og også `</body>` og `</html>`. Det er for å spare plass, men også for å sette fokus på den viktige PHP-koden. Når du lager egne websider, bør du derimot ta med fullstendig kode slik at det du lager, blir XHTML-korrekt.

Hvorfor skrive XHTML og fullstendig kode? Det har blant annet med integrasjonen av andre webteknologier, for eksempel Ajax, å gjøre. Ren og ryddig kode er viktig, og HTML er ikke rent nok til slike formål siden det blant annet godtar feil nøsting, ikke avslutning av tagger og liknende. Mange løsninger starter enkelt, men utvider seg senere, og derfor er det lurt å benytte XHTML først som sist.

### 1.3.9 PHP versus JavaScript og Ajax

HTML spesifiserer *hvordan* informasjonen skal presenteres. Vi har nevnt at løsninger som bygger på (X)HTML, er statiske. Utvidet funksjonalitet, dynamikk og interaksjon med brukeren krever programmering.

*JavaScript* er et scriptspråk som tilbyr en del interaktivitet og dynamikk. Alle script som er skrevet i JavaScript, *utføres på klienten*, i motsetning til script skrevet i PHP, som *utføres på tjeneren*. Det vil si at koden i scriptfilen blir kjørt av et program på tjenermaskinen hver gang siden blir forespurt. Tjeneren vil returnere *resultatet* til klienten for presentasjon, og dette resultatet vil typisk være i form av HTML.

Hva er best, å utføre et script på tjeneren eller klienten? Begge metodene har fordeler og ulemper, og helt klart egnede bruksområder. Du vil i løpet av boka få se mange eksempler hvor JavaScript inngår i PHP-scriptene for å gi ytterligere dynamikk og utnytte de sterke sidene til kjøring på klientsiden. Det er ikke bare mulig, men ofte veldig nyttig å bruke begge teknologiene sammen.

Ajax blir stadig mer populært, og er en av suksessfaktorene bak en rekke Web 2.0-tjenester. Ajax er i praksis JavaScript kombinert med en asynkron tjenerkommunikasjon og XML som utvekslingsformat. Ajax og PHP passer utmerket sammen og konkurrerer ikke. Mange eksisterende PHP-løsninger kan ajaxifiseres, og dermed bli mye bedre. Ajax gir nemlig mulighet for å lage responsive brukergrensesnitt der informasjon oppdateres uten at siden må lastes inn på nytt. Websider kan dermed bruke Ajax på klientsiden og foreta tung databehandling på tjenersiden med PHP. Denne boka tilbyr ingen teori om Ajax, men det fins mange gode bøker, veiledere og fag om Ajax.

## 1.4 Oppgaver og kontrollspørsmål

### Oppgave 1–1: Innstillinger

Utforsk ***php.ini*** dersom du har tilgang til denne.

– Sjekk hva innstillingen `display_errors` er satt til.
– Kan du finne innstillingen på en annen måte enn ved å åpne ***php.ini*** i en tekst-editor? Kan du i så fall se for deg noen situasjoner hvor dette kan være nyttig?

### Oppgave 1–2: Forståelse av hvordan PHP fungerer

Sørg for at setningen

```
AddType application/x-httpd-php-source .phps
```

er aktivert/lagt til i konfigurasjonsfilen til Apache (***httpd.conf***). Kopier koden fra eksempelet i kodesnutt 1.1 og lagre to versjoner med samme filnavn, der den ene for eksempel heter norge.php og den andre norge.phps. Filene skal ha nøyaktig likt innhold.

– Skriv inn adressen til de to scriptene i to nettleservinduer og sammenlikn både resultatet og kildekoden (velg funksjonalitet for å vise HTML-kildekode i nettleseren).
– Forklar forskjellen, gjerne med basis i samspillet mellom klient og tjener, og si noe om når denne teknikken med visning av scriptkoden kan være nyttig.
– Legg til passende kommentarer i koden du nettopp har lagd.

### Oppgave 1–3

Gå til et hvilket som helst nettsted. Se deg ut en side som du vil kopiere, vis kildekoden i nettleseren og lagre din egen variant av siden på din maskin/ditt område.

– Legg til dagens dato og ditt navn på et passende sted og observer resultatet.
– Lokaliser en side på Internett som slutter på .php. Hvorfor er det ingen PHP-kommandoer i kildekoden? Forklar.
– Hva er fordelene med PHP som programmeringsspråk?
– Kan JavaScript og PHP brukes sammen? Hvorfor (ikke)?