

Informatik



Dette er en eBogs-pdf af iBogen informatik.systime.dk . Bokse og skemaer kan ombrydes uhensigtsmæssigt på siderne. eBogen indeholder kun tekster og billeder fra iBogen. Du finder derfor ikke videoer eller andet interaktivt materiale fra iBogen i denne eBog.

INFORMATIK

© 2021 Martin Damhus, Jesper Buch, Elisabeth Husum m.fl. og Systime

Kopiering og anden gengivelse af dette værk eller dele deraf er kun tilladt efter reglerne i gældende lov om ophavsret eller inden for rammerne af en aftale med Copydan. Al anden udnyttelse forudsætter en skriftlig aftale med forlaget.

ISBN 9788761689122

Indholdsfortegnelse

| | |
|---|-----------|
| Informatik | 1 |
| Introduktion til underviseren | 5 |
| Projekter | 7 |
| Animation af redoxreaktion | 8 |
| Beregning af LIX | 9 |
| Computerens fysik | 11 |
| Drone-simulator | 11 |
| Dyrekllinik | 13 |
| Firmahjemmeside | 15 |
| Førstehjælpsguide | 17 |
| Glosetræner | 19 |
| Internethandel | 21 |
| It-politik i en virksomhed | 22 |
| Nudging | 24 |
| Pandaria Musik | 26 |
| Partitest | 27 |
| Reklamespil | 29 |
| Sikkerhed på internettet | 30 |
| Træningsprogram | 31 |
| 1. Fra idé til færdigt it-system | 34 |
| Arbejdsmetoden Scrum | 36 |
| 2. Planlægning af et it-system | 51 |
| Brainstorm | 51 |
| Analyse | 53 |
| Kravspecifikation | 55 |
| Målgrupper | 57 |
| Gallup Kompas og Minerva-modellen | 59 |
| conzoom® | 62 |
| Data mining | 63 |
| 3. Udarbejdelse af et it-system | 65 |
| Interaktionsdesign | 65 |
| Metoder til design af brugerflader | 68 |
| Metoder til vurdering af brugerflader | 84 |
| Opgave til interaktionsdesign | 88 |
| Gestaltlovene | 90 |
| Modellering | 104 |
| Algoritmer | 105 |
| Databaser | 108 |
| Analyse | 109 |
| E/R-diagram | 112 |
| Nøgler | 114 |
| Tabelskitser | 116 |
| Normalformer | 118 |
| Programmering | 127 |

| | |
|--|------------|
| Syntaks og semantik | 129 |
| Kontrolstrukturer og funktioner | 139 |
| Sekvenser | 141 |
| Forgreninger | 148 |
| While-løkker | 158 |
| For-løkker | 170 |
| Funktioner | 178 |
| Data og operationer | 190 |
| Tal | 192 |
| Strenge | 194 |
| Sandhedsværdier | 205 |
| Arrays | 215 |
| Datatyper | 222 |
| Konstanter | 223 |
| Kommentarer i koden | 228 |
| Opgaver til programmering | 231 |
| 4. Evaluering af et it-system | 240 |
| Test | 240 |
| Tænke-højt-test | 242 |
| Brugervejledning | 245 |
| Dokumentation | 246 |
| 5. Andet materiale | 249 |
| Billeder og farver | 250 |
| Bitmap-billeder | 251 |
| Vektorgrafik | 257 |
| Komprimering | 258 |
| Farver | 262 |
| Computeren | 266 |
| Processor (CPU) og hukommelse (RAM) | 267 |
| Hardware-opbygning | 268 |
| Opbygning af RAM | 270 |
| Harddisk og andet lager | 271 |
| Filer | 271 |
| Liste af kendte filtyper | 272 |
| Bits og bytes | 274 |
| Opgaver til Computeren | 276 |
| Deleøkonomi | 277 |
| Digital selvbetjening | 279 |
| HTML og CSS | 281 |
| Innovation i IT | 285 |
| 4p-modellen for innovation | 287 |
| Radikal og inkrementel innovation | 292 |
| Interaktion med databaser | 293 |
| IT-historie | 296 |
| IT-sikkerhed | 301 |
| Fortrolighed, integritet og tilgængelighed | 303 |
| Privacy | 305 |
| Brugere og hackere | 306 |

| | |
|--|------------|
| Kodeord og Adgangskontrol | 307 |
| Kommunikation over netværk | 309 |
| Kryptografi | 311 |
| Kryptering | 313 |
| Hashing | 318 |
| Digital signatur | 319 |
| Kryptografisk opsummering | 322 |
| Antivirus-software og Firewalls | 323 |
| Opgaver til it-sikkerhed | 324 |
| Klient-server arkitektur | 327 |
| Kommunikation | 329 |
| Laswells kommunikationsmodel | 330 |
| Støj og kommunikation | 332 |
| Moderne kommunikations-modeller | 334 |
| AIDA og HYSO | 335 |
| Model-View-Controller-arkitektur | 340 |
| Operativsystem og processer | 343 |
| Personlige data | 347 |
| Sociale medier | 350 |
| Trelags-arkitektur | 352 |
| Elevoplæg | 357 |

Introduktion til underviseren



iStockphoto.com/vladwel

Denne iBog er tiltænkt undervisningen i informatik på C- og B-niveau og dækker læreplanen.

Læreplanen til informatik på C- og B-niveau er bygget op, så B-niveauet er en overbygning på C-niveauet.

Da informatik er et udpræget skaberfag, tager bogen udgangspunkt i projekter, hvor eleverne i de fleste tilfælde skal udvikle et it-system. Den konkrete metode, som iBogen understøtter til udvikling af et it-system, er den [iterative metode](#).

Det anbefales, at man parallelt med et projektforløb inddrager relevante materialeafsnit. Projektbeskrivelser findes i det første afsnit. For hvert projekt er der forslag til valg af materiale, men det er selvfølgelig muligt at fravælge noget og tilvælge andet efter ønske.

Materialeafsnittene er samlet i fire afsnit og et afsnit med andet materiale af mere generel karakter.

- Første afsnit indeholder it-fagets iterative og inkrementielle arbejdsmetode og den konkrete arbejds metode Scrum. Man behøver dog ikke at anvende Scrum i undervisningen.
- Andet afsnit indeholder materiale, der er relevant i planlægningen af et it-system.
- Tredie afsnit er relevant i forhold til udarbejdelsen af it-systemet.
- Fjerde afsnit indeholder materiale til test og evaluering af it-systemet.

Derudover er der i iBogen som nævnt afsnittet *Andet materiale*. Dele af dette er kernestof i faget, og andet har karakter af supplerende stof.

Der er i afsnit 3 peget på fire programmeringssprog. Intentionen er, at man fokuserer på et af dem. Alternativt kan man vælge et femte programmeringssprog og alligevel bruge kapitlet til undervisning i programmering.

Projekter



iStockphoto.com/IconicBestiary

iBogens opbygning

Da faget informatik i høj grad drejer sig om at arbejde med det faglige stof ved at lave projekter, hvor man laver it-systemer, er strukturen i denne bog som følger:

Projekterne kommer først. Derefter følger fire kapitler, der indeholder de arbejdstrin, som kan være relevante for at ende med et færdigt it-system.

Derudover kan man i arbejdsprocessen med at lave et it-system have brug for ekstra materiale. Dette ligger under punktet Andet materiale. Man kan også arbejde med dette materiale uden at være i gang med udviklingen af et it-projekt.

Animation af redoxreaktion



iStockphoto.com/Grafner



Projektbeskrivelse: Animation af redoxreaktion

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en eller flere animationer af redoxreaktioner. Som du ved, foregår der en kemisk reaktion, hvor oxidationstrin ændres, og elektroner bliver overført fra et stof til et andet. Det er denne overførsel, du skal animere i [Scratch](#).

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en animation, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet til it-systeme laves. Du skal bruge en brainstorm til at få en god idé til animationen. Derefter skal du beskrive kravene til animationen. Til sidst skal du definer, hvem der skal have glæde af dit it-system.

2. [Planlægning \(se side 51\)](#) og særligt om [kommunikation \(se side 329\)](#).

Nu er forarbejdet gjort og selve udarbejdelsen af animationen kan gå i gang. Animationen skal programmeres i Scratch.

3. [Udarbejdelse \(se side 65\).](#)

Animationen er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\).](#)

Beregning af LIX



iStockphoto.com/themacx



Projektbeskrivelse: Beregning af LIX

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en hjemmeside, hvor man kan få beregnet LIX i en tekst. Bruger indtaster en tekst, og hjemmesiden beregner LIX.

Formlen for LIX:

$$\text{LIX} = \frac{\text{Antal ord i teksten}}{\text{Antal punktummer i teksten}} + \frac{\text{Antal lange ord (over 6 tegn lange)} \cdot 100}{\text{Antal ord i teksten}}$$

LIX kan placeres i følgende intervaller:

| | |
|-------|--|
| >55 | Meget svær, faglitteratur på akademisk niveau, lovtekster. |
| 45-54 | Svær, fx saglige bøger, populærvidenskabelige værker, akademiske udgivelser. |
| 35-44 | Middel, fx dagblade og tidsskrifter. |
| 25-34 | Let for øvede læsere, fx ugebladslitteratur og skønlitteratur for voksne. |
| <24 | Let tekst for alle læsere, fx børnelitteratur. |

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet laves til it-systemet. Du skal beskrive kravene til it-systemet og definer, hvem der skal have glæde af hjemmesiden.

2. [Planlægning \(se side 51\).](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Hjemmesiden laves. Siderne laves i HTML og CSS.

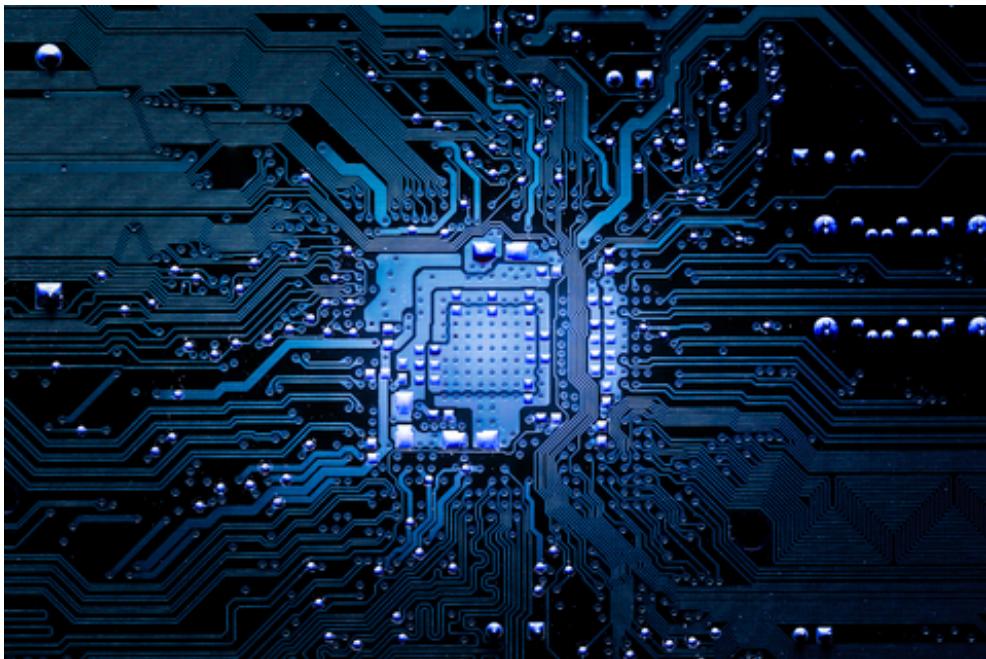
3. [Udarbejdelse \(se side 65\) og særligt om \[HTML og CSS \\(se side 281\\).\]\(#\)](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\).](#)

Der kan laves en udvidelse af it-systemet, så en tekst på forhånd er i systemet, og bruger så kan ændre visse navneord og se, hvordan LIX ændres.

Computerens fysik



iStockphoto.com/tcareob72



Projektbeskrivelse: Computerens fysik

I dette projekt skal du arbejde med computerens fysik. Du skal arbejde med computerens [fysiske enheder](#) og du skal arbejde med de [it-systemer](#), der gør det muligt at afvikle dine it-systemer på en computer.

Drone-simulator



iStockphoto.com/Naypong



Drone-simulator

I dette projekt skal du planlægge, udarbejde og evaluere et it-system, der skal være en drone-simulator. Simulatoren skal give brugeren mulighed for at flyve rundt i et landskab, og dronen skal undgå diverse forhindringer. Du skal have fokus på, at tyngdekraft og fart er troværdig. Du skal også lave en løsning, hvor ikke kun baggrunden bevæger sig.

Bemærk, at droner i vore dage benyttes af en bred vifte af fagpersoner. Desuden er en drone også legetøj, men under alle omstændigheder er droneflyvning underlagt nogle begrænsninger, som du kan læse om her: [Retsinformation](#)

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en drone-simulator, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igen nem flere gange:

1. [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til grundidéen til drone-simulatoren. Derefter skal du (og en evt. bruger?) beskrive kriteriene til it-systemet. Til sidst skal I præcist definere, hvem der skal have glæde af drone-simulatoren.

2. [Planlægning \(se side 51\)](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af simulatoren kan gå i gang. Hvis der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign.

3. [Udarbejdelse \(se side 65\)](#)

Simulatoren er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\)](#)

Dyrekllinik



iStockphoto.com/SbytovaMN



Dyreklinik

I dette projekt skal du planlægge, udarbejde og evaluere et it-system, der skal bestå af en hjemmeside til en dyreklinik og en tilhørende online "patientjournal".

Dyreklíniken ønsker et it-system, hvor kunderne kan blive medlemmer (med tilhørende password), og hvor de dels kan hente alle relevante informationer om dyrekliniken og dens services, og hvor de specifikt kan tilgå eget dyrs journal. Denne journal skal indeholde alle tænkelige informationer om stamdata, stambogsforhold, vaccinationer, medicin og skader. Systemet skal være webbaseret.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

1. [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til grundidéen til hjemmesiden. Derefter skal du (og en evt. bruger?) beskrive kravene til it-systemet. Til sidst skal I præcist definere, hvem der skal have glæde af hjemmesiden.

2. [Planlægning \(se side 51\)](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign. Desuden skal der laves en database. Til sidst skal hjemmesiden laves. Siderne laves i HTML og CSS, og kommunikationen med databasen laves i php.

3. [Udarbejdelse \(se side 65\)](#) og særligt om [html og CSS \(se side 281\)](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\)](#)

Firmahjemmeside



iStockphoto.com/NicoElNino



Projektbeskrivelse: Firmahjemmeside

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en hjemmeside til et firma. Firmaet ønsker at præsentere sig på internettet.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

1. [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til firmatypen og grundidéen til hjemmesiden. Derefter skal du (og firmaet?) beskrive kravene til it-systemet. Til sidst skal I definere, hvem der skal have glæde af hjemmesiden:

2. [Planlægning \(se side 51\)](#)

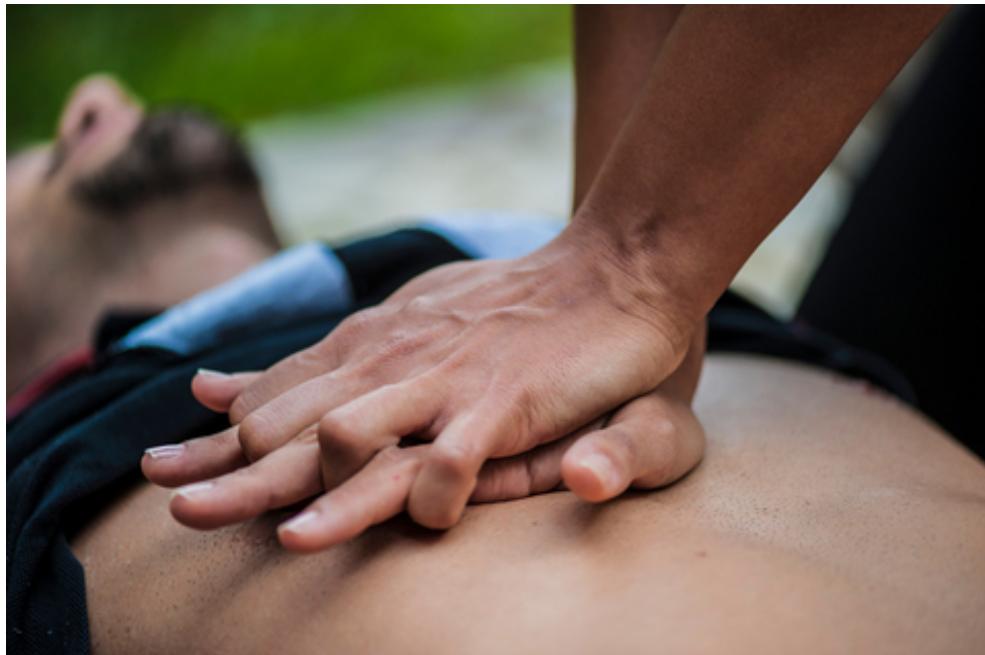
Nu er forarbejdet gjort og selve udarbejdelsen af hjemmesiden kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign. Hjemmesiden laves. Siderne laves i HTML og CSS:

3. [Udarbejdelse \(se side 65\) og særligt om HTML og CSS \(se side 281\).](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den:

4. [Evaluering \(se side 240\)](#)

Førstehjælpsguide



iStockphoto.com/Pixel_away



Projektbeskrivelse: Førstehjælpguide

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en hjemmeside til førstehjælp. Din hjemmeside skal indeholde instruktion i førstehjælp, fx at stoppe blødning. Du skal lave en hjemmeside med små delfilm, der trinvist viser, hvordan man yder den ønskede førstehjælp.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til hjemmesiden. Derefter skal du beskrive kravene til it-systemet. Til sidst skal I definere, hvem der skal have glæde af hjemmesiden.

2. [Planlægning \(se side 51\).](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Da det er et vigtigt emne, er det en god idé at lave et virkelig godt interaktionsdesign. Filmene med delinstruktionerne optages, og hjemmesiden laves. Siderne laves i HTML og CSS.

3. [Udarbejdelse \(se side 65\) og særligt om \[HTML og CSS \\(se side 281\\).\]\(#\)](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\).](#)

Glosetræner



iStockphoto.com/nicoolay



Projektbeskrivelse: Glosetræner

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en glosetræner. Du skal forestille dig, at glosetræneren skal bruges til små børn, der i folkeskolen lige har fået faget engelsk. Hver glose skal følges af et billede eller en illustration, og du skal tænke over, hvordan du rammer denne specielle brugergruppe.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en glosetræner, til du står med det færdige resultat, ses her Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til glosetræneren. Derefter skal du beskrive kravene til it-systemet.

2. [Planlægning \(se side 51\)](#) og særligt om [kommunikation \(se side 329\).](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af glosetræneren kan gå i gang. Da der er børn involveret, er det en god idé at lave et virkelig godt interaktionsdesign.

3. [Udarbejdelse \(se side 65\).](#)

It-systemet er færdigt, og det skal kvalitetssikres. Dette gøres ved at evaluere det.

4. [Evaluering \(se side 240\).](#)

Internethandel



iStockphoto.com/martin-dm



Internethandel

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en hjemmeside til et firma. Firmaet ønsker at tilbyde internethandel. Det skal være muligt for kunderne at bestille varer over nettet, og firmaejeren skal kunne få overblik over kunder og lager vha. dit it-system.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

1. [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til firmatypen og grundidéen til hjemmesiden. Derefter skal du (og firmaet?) beskrive kravene til it-systemet. Til sidst skal I definere, hvem der skal have glæde af hjemmesiden.

2. [Planlægning \(se side 51\)](#)

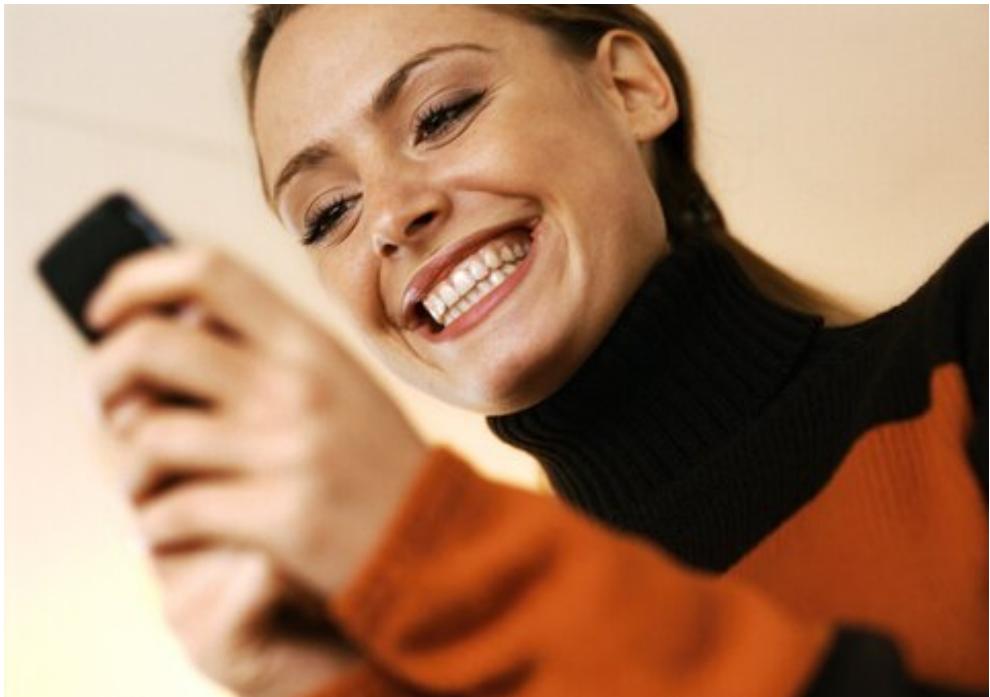
Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign. Desuden skal der laves en database til data over kunderne og varerne. Der er sikkert også andre ting, som skal i databasen. Til sidst skal hjemmesiden laves. Siderne laves i HTML og CSS, og kommunikationen med databasen laves i php.

3. [Udarbejdelse \(se side 65\)](#) og særligt om [html og CSS \(se side 281\)](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\)](#)

It-politik i en virksomhed



Colourbox.com

Må en arbejdsgiver videoovervåge dig i din arbejdstid uden du ved det? Ja og nej. I kortere tid, ja – dog skal der være en begrundelse for overvågningen, før det er lovligt, og data må ikke bare gemmes for evigt.

Må chefen tjekke dine sms'er på firmatelefonen? Ja, men det er ikke altid fyringsgrund.



Opgave: Diskutér artikler om hvad chefen må overvåge.

Find artikler om, hvad der er lovligt (lav evt. en google-søgning om "hvad chefen må overvåge").

- Er det overraskende begrænsninger?
- Er der overvågning i jeres nuværende jobs?
- Er begrænsningerne i orden?
- Vil et job i en overvåget stilling være attraktivt?
- Hvilke rettigheder/muligheder skal virksomheder tilbyde i forbindelse med overvågning?

Hvorfor er det relevant for virksomheder at overvåge deres ansatte?

Mange virksomheder har en personale-/it-politik, der forbyder ansattes brug af visse internetsider, netbanking, sociale medier, private mails og sms'er, samt installering af 'egne' programmer mv. på arbejdsrelateret udstyr.



Opgave: Diskutér artikler om it-vaner på arbejdspladsen

Find artikler, der fortæller om unges it-vaner (søg fx på "virksomheders it-vaner", "brug af it i arbejdstiden", "Facebook koster mia" og "Facebooks dårligdomme").

Du skal se dig i rollen som arbejdsgiver:

- Hvilke rettigheder og muligheder synes du, at de ansatte skal have?
- Hvilke krav skal der stilles til de ansatte?
- Er det i orden, at ansatte lægger programmer på en virksomhedscomputer uden licens?
- Hvilke typer af virksomheder kræver en stram it-politik og hvilke en mere lempelig?
- Hvad vil konsekvensen være, hvis en virksomhed laver en it-politik, der er meget omfattende?



Projekt: It-politik for en virksomhed

Du skal lave en it-politik for en virksomhed.

Indledningsvist skal du identificere og beskrive den valgte virksomhed. Dernæst kan it-politikken laves ud fra Digitaliseringsstyrelsens [skabelon](#). Skabelonen tilpasses, så den passer til dit virksomhedsvalg.

Nudging



iStockphoto.com/gilaxia



Projektbeskrivelse: Nudging

I dette projekt skal du planlægge, udarbejde og evaluere et IT-system i form af et spil. Spillet skal nudge spilleren til at ændre adfærd hen imod en hensigtsmæssig form.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave et spil, til du står med det færdige resultat, ses her. Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til spillet. Derefter skal du beskrive kravene til spillet. Til sidst skal I definere, hvem der skal have glæde af spillet.

2. [Planlægning \(se side 51\)](#) og særligt om [kommunikation \(se side 329\)](#).

Nu er forarbejdet gjort, og selve udarbejdelsen af spillet kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign til spillet. Spillet skal programmeres i Scratch.

3. [Udarbejdelse \(se side 65\).](#)

Spillet er færdigt, og det skal kvalitetssikres. Dette gøres ved at evaluere det.

4. [Evaluering \(se side 240\).](#)

Pandaria Musik



iStockphoto.com/vm



Pandaria Musik

I dette projekt skal du planlægge, udarbejde og evaluere et it-system, der kan administrere et lille koncertsted.

Pandaria Musik er et lille musiksted, som nogle af dine venner har lavet. De spiller selv i forskellige bands og har en del venner, der både spiller og gerne vil til koncert. De er begyndt at tage entré for at dække udgifter som husleje etc. De er kommet til dig for at få udviklet et it-system, der kan administrere billetter, koncerter og bands. De vil gerne have, at der til nogle koncerter kan være faste pladser med pladsnumre. Derfor skal der være en plan over siddepladserne med numre, og det skal være muligt løbende at se, hvilke pladser der allerede er købt.

It-systemet skal være webbaseret.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

- [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til til hjemmesiden. Derefter skal du (og vennerne?) beskrive kravene til it-systemet. Til sidst skal I definere, hvem der skal have glæde af hjemmesiden.

- [Planlægning \(se side 51\)](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign. Desuden skal der laves en database til data over koncerter, pladser, billetter etc. Der er sikkert også andre ting, som skal i databasen. Til sidst skal hjemmesiden laves. Siderne laves i HTML og CSS, og kommunikationen med databasen laves i php.

- [Udarbejdelse \(se side 65\) og særligt om html og CSS \(se side 281\)](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

- [Evaluering \(se side 240\)](#)

Partitest



iStockphoto.com/bizoo_n



Projektbeskrivelse: Partitest

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af en hjemmeside med en test. Brugeren skal ud fra fx 10 spørgsmål, hvor der kan svares enig/ikke enig, have svar på, hvilket parti han eller hun bør stemme på.

Den overordnede struktur på dit arbejde – fra du ved du skal lave en hjemmeside, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet til it-systemet laves. Du skal bruge en brainstorm til at få formuleret de gode spørsgmål til testen. Derefter skal du beskrive kravene til it-systemet. Til sidst skal du definere, hvem der skal have glæde af hjemmesiden.

2. [Planlægning \(se side 51\).](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af hjemmesiden kan gå i gang. Der skal laves en database til data med spørgsmålene. Til sidst skal hjemmesiden laves. Siderne laves i HTML og CSS, og kommunikationen med databasen laves i php.

3. [Udarbejdelse \(se side 65\) og særligt om HTML og CSS \(se side 281\).](#)

Hjemmesiden er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

4. [Evaluering \(se side 240\).](#)

Reklamespil



iStockphoto.com/jacoblund



Projektbeskrivelse: Reklamespil

I dette projekt skal du planlægge, udarbejde og evaluere et it-system i form af et reklamespil til et firma. Spillet skal reklamere for et produkt eller et budskab, og spilaspektet skal være tydeligt.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave et reklamespil, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange.

1. [Fra idé til færdigt system \(se side 34\).](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til spillet. Derefter skal du (og firmaet?) beskrive kravene til spillet. Til sidst skal I definere, hvem der skal have glæde af spillet.

2. [Planlægning \(se side 51\)](#) og særligt om [kommunikation \(se side 329\).](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af spillet kan gå i gang. Da der er kunder involveret, er det en god idé at lave et virkelig godt interaktionsdesign til spillet. Spillet skal programmeres i Scratch.

3. [Udarbejdelse \(se side 65\).](#)

Spillet er færdigt, og det skal kvalitetssikres. Dette gøres ved at evaluere det.

4. [Evaluering \(se side 240\).](#)

Sikkerhed på internettet



iStockphoto.com/imaginima



Projektbeskrivelse: Sikkerhed på internettet

I dette projekt skal du arbejde med sikkerhed på internettet. Du finder materialet under [Andet materiale \(se side 301\)](#).

Du skal lave opgaverne til afsnittet og samle besvarelserne. Igennem hele forløbet skal du se sikkerhedsaspekterne ud fra en privatpersons synsvinkel og overveje problemerne fra et sikkerhedsaspekt.

Træningsprogram



iStockphoto.com/julief514



Træningsprogram

I dette projekt skal du planlægge, udarbejde og evaluere et it-system, der skal indeholde et træningsprogram, som målrettet indeholder træning i forbindelse med adgangsprøven til politiuddannelsen.

Når man søger ind til politiet, skal man bestå en fysisk test. Ikke alle aspiranter kan umiddelbart bestå denne test. Du skal lave en web-app, som giver træningshjælp, der er målrettet adgangsprøven. Det skal være muligt for brugeren løbende at indtaste sine tider i de forskellige discipliner, og din app skal gemme disse. Der skal være en grafisk præsentation af tallene, så det er nemt at se, om man gør fremskridt.

Der skal udvikles en web-app.

Den overordnede struktur på dit arbejde – fra du ved, at du skal lave en app, til du står med det færdige resultat – ses her. Husk, at du måske skal faserne igennem flere gange:

1. [Fra idé til færdigt system \(se side 34\)](#)

Nu skal forarbejdet laves til it-systemet. Du skal bruge en brainstorm til at få en god idé til grundidéen til app'en. Derefter skal du (og en evt. bruger?) beskrive kravene til it-systemet. Find også de konkrete fysiske krav på nettet. Til sidst skal I præcist definere, hvem der skal have glæde af app'en.

2. [Planlægning \(se side 51\)](#)

Nu er forarbejdet gjort, og selve udarbejdelsen af appen kan gå i gang. Det en god idé at lave et virkelig godt interaktionsdesign, blandt andet fordi en mobiltelefon har en mindre skærm end en pc, og derfor skal pladsen optimeres. Desuden skal der laves en database til data over delresultaterne og de enkelte øvelser. Der er sikkert også andre ting, som skal i databasen. Til sidst skal app'en laves. Siderne laves i HTML og CSS, og kommunikationen med databasen laves i php.

3. [Udarbejdelse \(se side 65\)](#) og særligt om [html og CSS \(se side 281\)](#)

App'en er færdig, og den skal kvalitetssikres. Dette gøres ved at evaluere den.

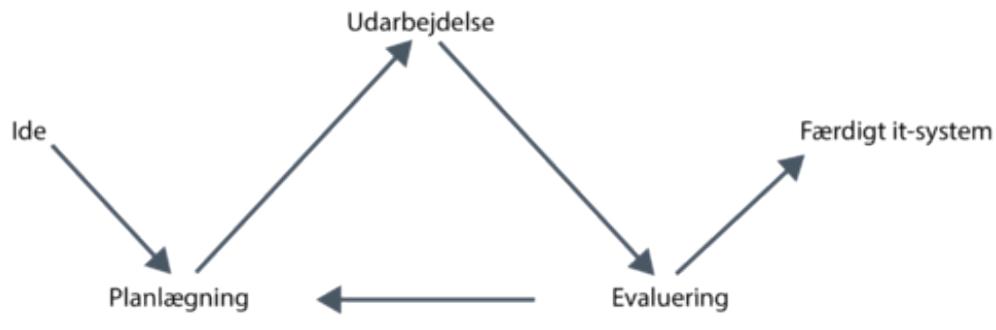
4. [Evaluering \(se side 240\)](#)

1. Fra idé til færdigt it-system

Selv når et simpelt it-system skal laves, ligger der et stykke planlægningsarbejde forud for selve programmeringsarbejdet og resultatet bør evalueres bagefter.

Skal man løse en mere kompliceret opgave, er det ikke nok at planlægge, programmere og evaluere. Så skal man faserne igennem flere gange, fordi evalueringsfasen vil afsløre, at der skal laves ændringer, udvidelser og forbedringer.

En sådan gentagende arbejdsproces kaldes iterativ. Når man laver ændringer, udvidelser og forbedringer, arbejder man inkrementelt.



Det er vigtigt at bruge gode arbejdsmetoder, når man designet et it-system. Gør man ikke det, er der risiko for, at resultatet kan blive af dårlig kvalitet. Det kan være, at man får lavet et it-system, som slet ikke løser den opgave, der faktisk skulle løses. Eller man får lavet et it-system, der er for besværligt at anvende for brugerne.

Eksempel: It-system der ikke løser opgaven

Den kode, der er i en billig lommeregner, regner forkert, når den skal udregne $10 - 3 \cdot 2$

Lommeregneren giver resultatet 14 og ikke det rigtige resultat 4.

Eksempel: It-system der er for besværligt for brugeren at bruge

Det er for svært at anvende en avanceret mobiltelefon for ukyndige it-brugere.

Problemet er løst ved at lave en særlig simpel mobiltelefon til denne brugergruppe.



Opgave: Dårlige it-systemer

Find eksempler på it-sytemer, der enten ikke løser den stillede opgave eller er for besværlige for brugeren at anvende.

En god arbejdsmetode kommer omkring de fleste af følgende trin:

- brainstorm
- kravspecifikation
- målgrupper
- interaktionsdesign
- modellering
- programmering
- test
- brugervejledning
- dokumentation

Som listen viser, er selve programmeringstrinet kun ét af mange trin, der er mindst lige så vigtige.

Det kan være en idé at arbejde med flere trin samtidig. Fx kan man udarbejde dokumentation løbende.

Man kan også give forskellige arbejdsroller til de enkelte medlemmer i en projektarbejdsgruppe. Én kan fx være testansvarlig, mens en anden er dokumentationsansvarlig.



Opgave: Hvilke trin hører til hvilke af de tre faser?

Placér følgende trin:

- test
- modellering
- brainstorm
- kravspecifikation
- brugervejledning
- målgrupper
- dokumentation
- programmering
- interaktionsdesign

i en af de tre faser:

- planlægning
- udarbejdelse
- evaluering

Arbejdsmetoden Scrum

OBS

Dette er kernestof på B-niveau.

Arbejdsmetoden SCRUM er en såkaldt agil arbejdsmetode. Agil betyder bevægelig og adræt.

Når man bruger arbejdsmetoden Scrum, kan man hurtigt lave ændringer og tilpasninger undervejs i arbejdsprocessen, og man kan hoppe rundt i triene og faserne. Dette er især nødvendigt, når man laver IT-systmer, hvor man måske ikke fra start kender de konkrete krav til systemet - eller hvor kravene måske ændrer sig undervejs.

SCRUM

Scrum er en arbejdsmetode til projektledelse af udviklingsprocesser, hvor man arbejder iterativt og inkrementelt.

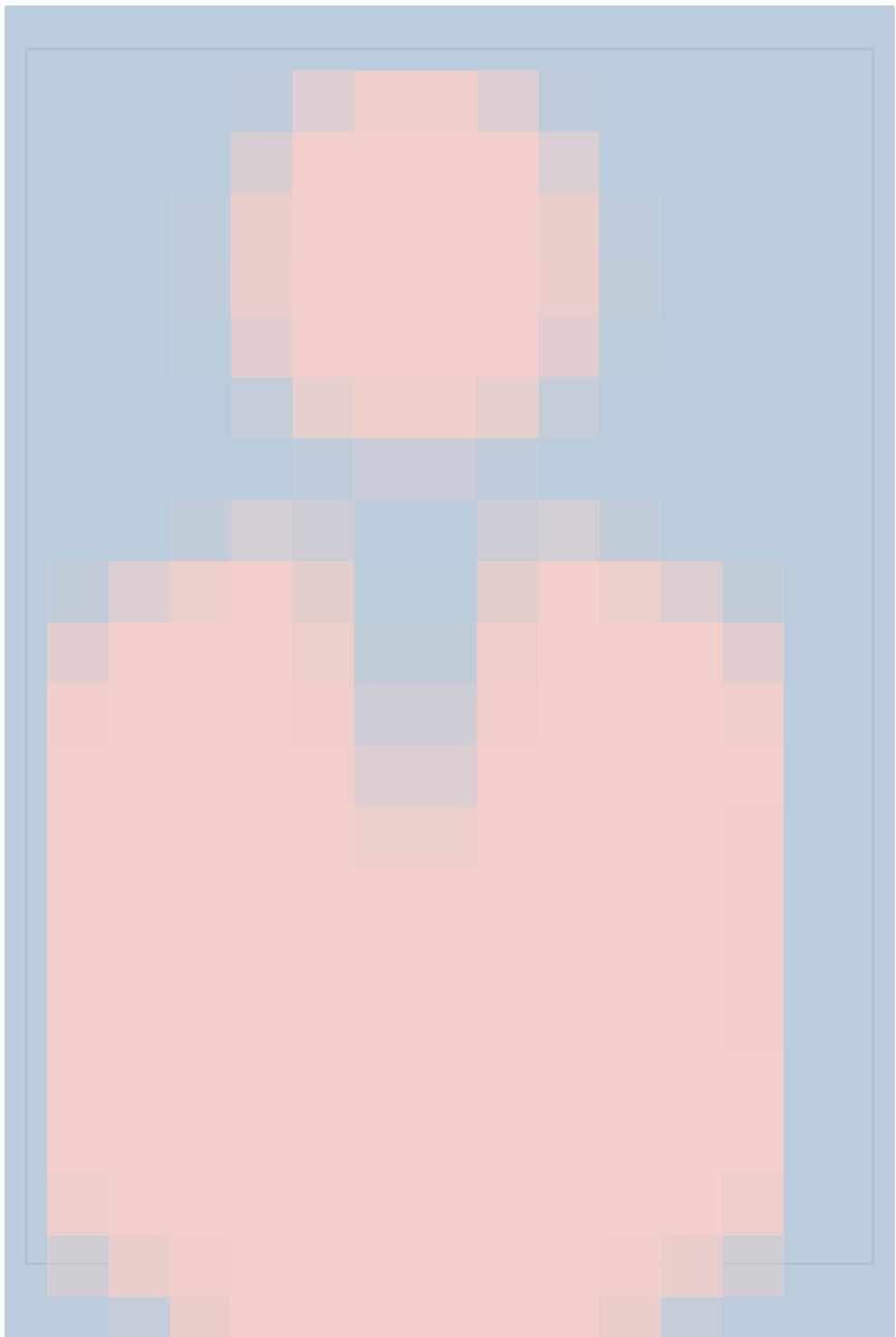
Når man arbejder med arbejdsmetoden Scrum, opdeler man arbejdet med at udvikle et IT-system i mindre dele. Disse mindre dele laves i såkaldte sprint. Et sprint er en iteration.

Scrum bruger tre roller, tre værktøjer og tre mødeformer.

Roller

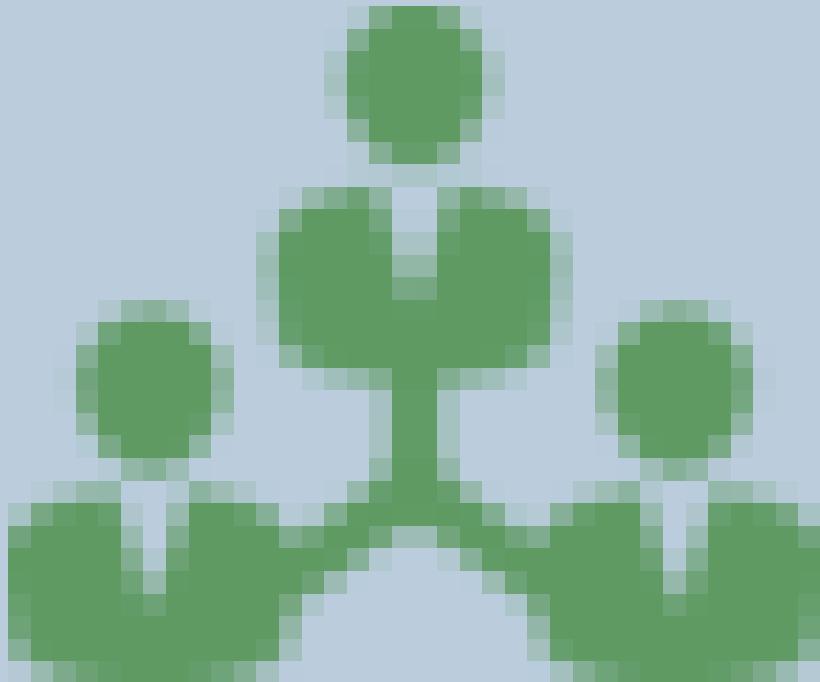
Rollerne er Product Owner, Team og Scrum Master, og rollerne tildeles de personer, der skal udvikle it-systemet.

Product Owner



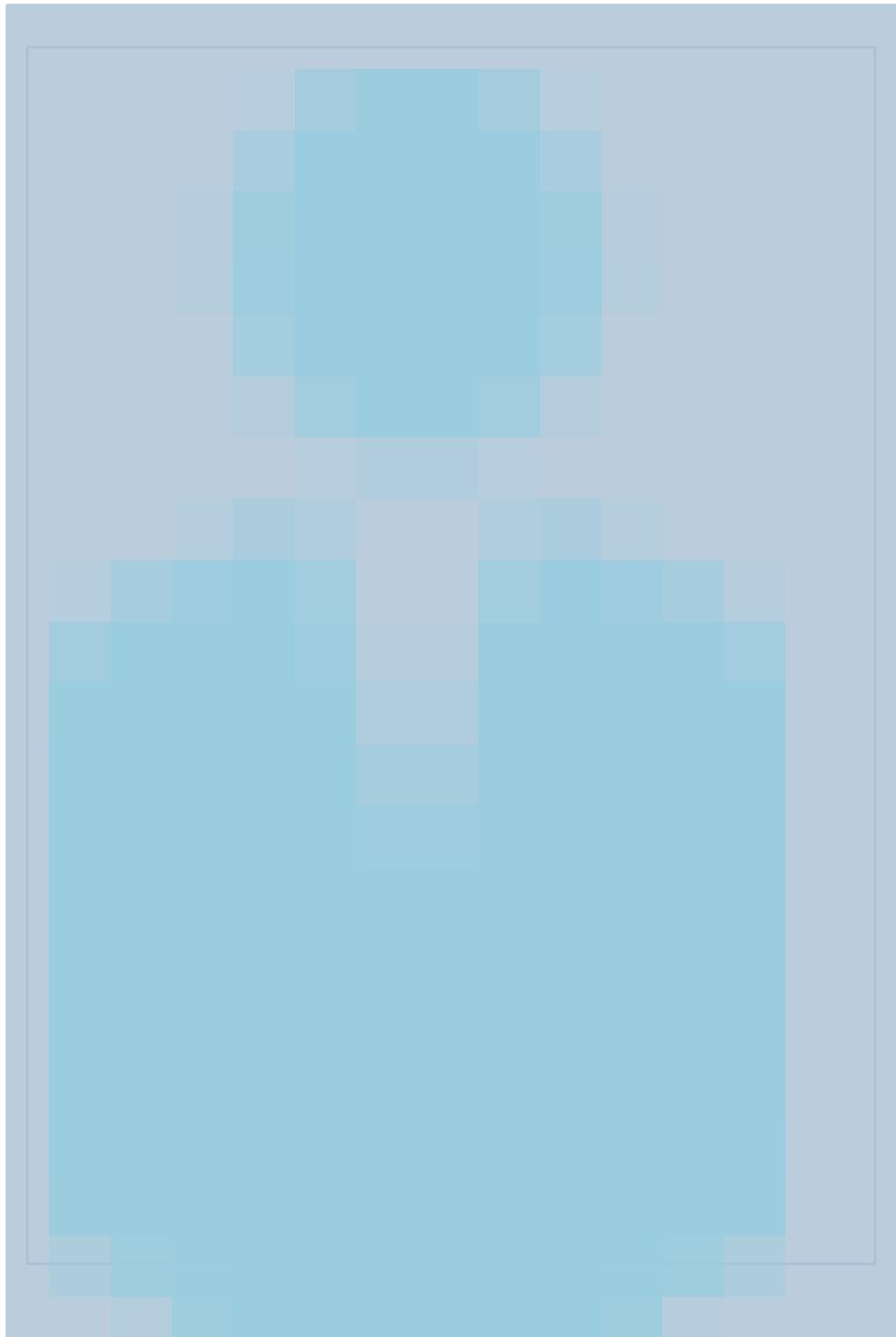
Product Owner har det overordnede ansvar for, at it-systemet bliver lavet og har ansvaret for Product Backlog (se nedenfor).

Team



Team-medlemmerne er den arbejdsgruppe der udfører de delopgaver, der skal laves, for at it-systemet samlet set bliver færdigt.

Scrum Master



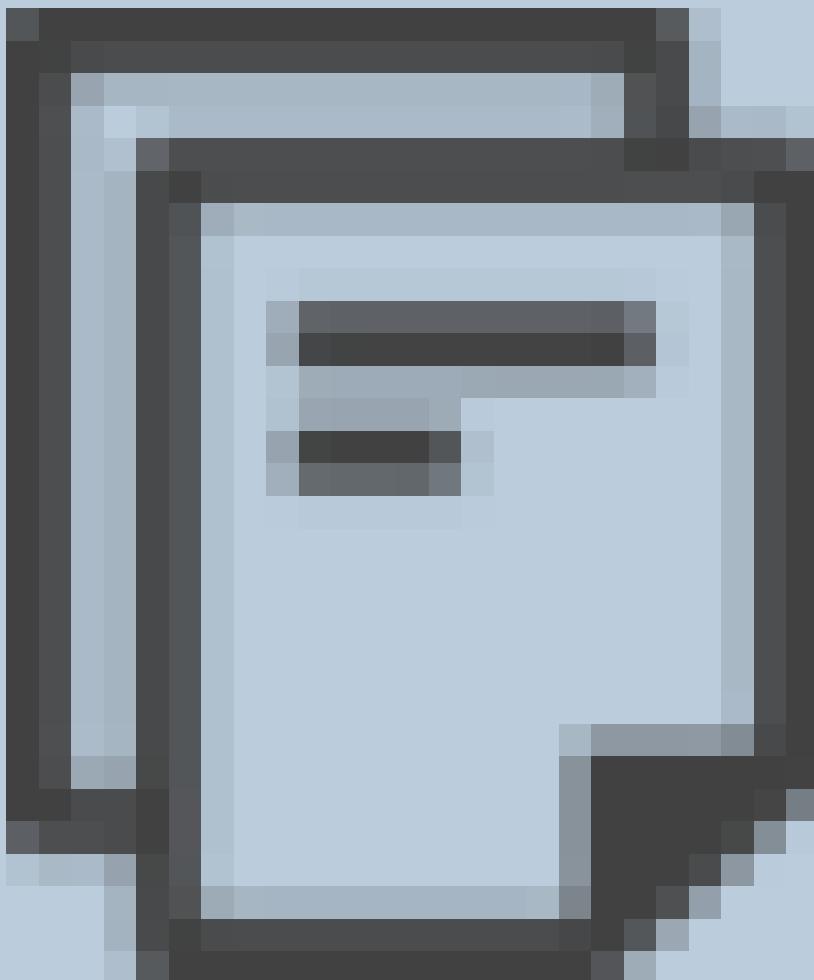
Scrum Master har ansvaret for, at arbejdsmetoden Scrum bliver anvendt korrekt.

Både ProductOwner og ScrumMaster kan også deltagte i Team, så har de bare to roller.

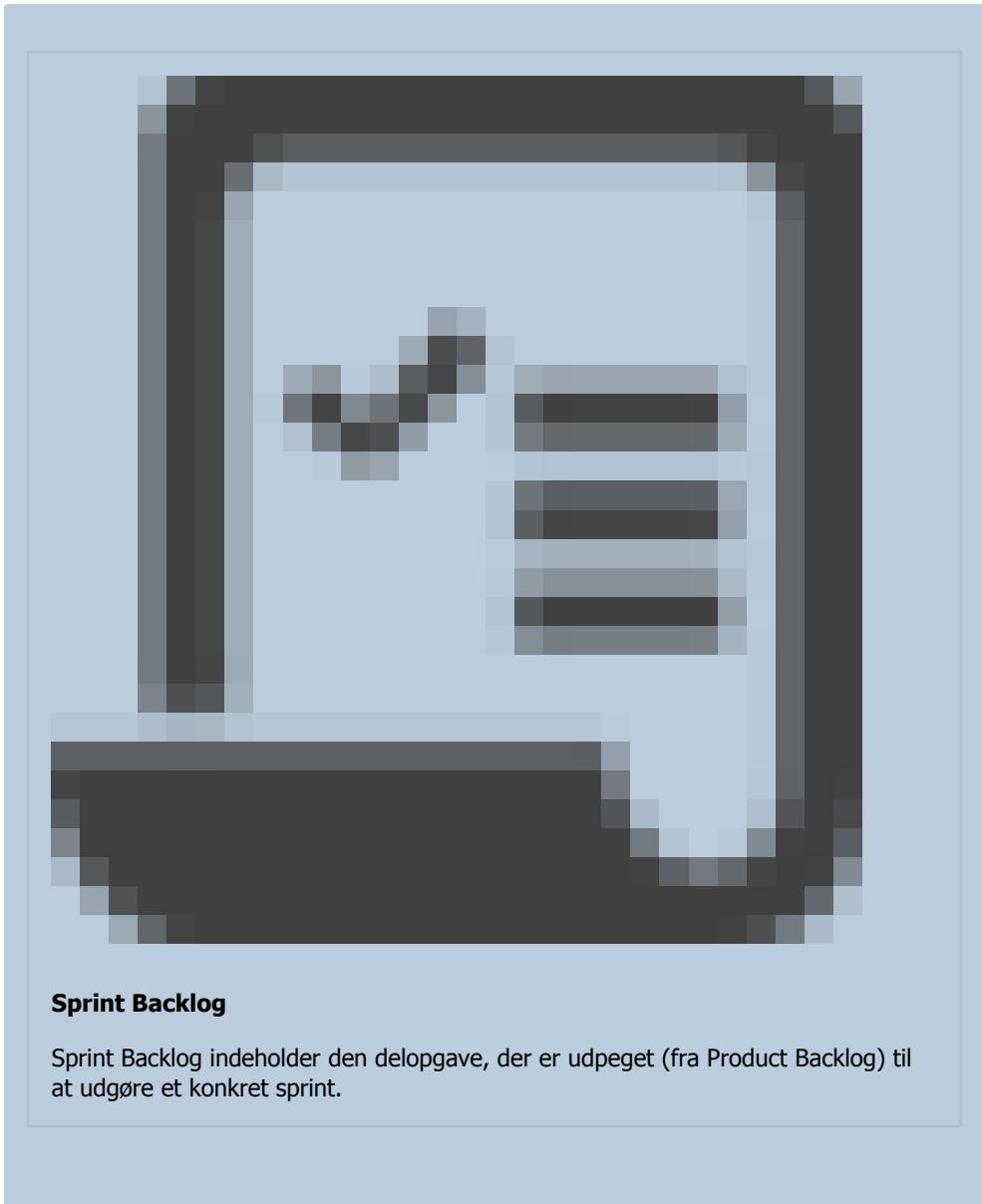
Værktøjer

Værktøjerne er Product Backlog, Sprint Backlog og Burndown Chart. Disse er dokumenter, der bruges til at styre arbejdsprocessen.

Product Backlog



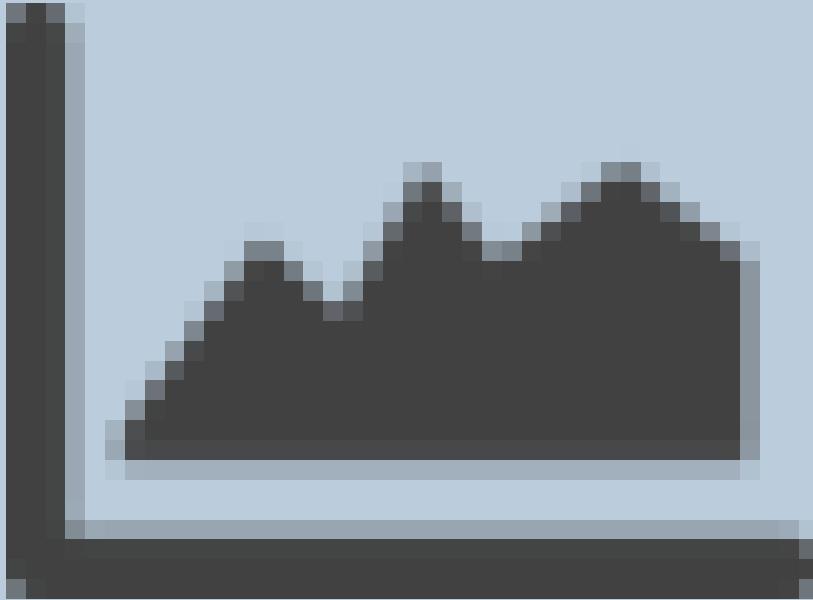
Product Backlog består af en liste med alt, der er relevant for opgaven. Den indeholder krav, beskrivelser, rækkefølger og estimerater.



Sprint Backlog

Sprint Backlog indeholder den delopgave, der er udpeget (fra Product Backlog) til at udgøre et konkret sprint.

Burndown Chart



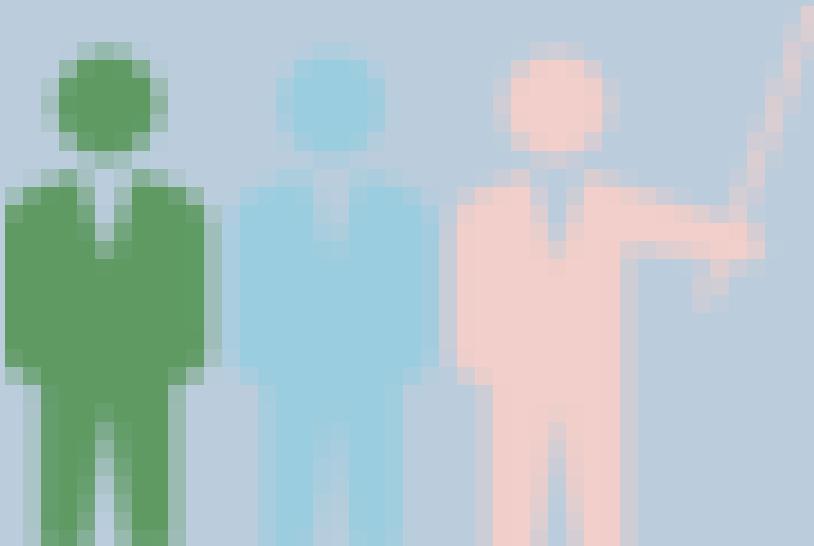
Burndown Chart indeholder en oversigt over, hvor meget tid der fremadrettet er nødvendig for at løse delopgaverne i et sprint.

Product Owner har ansvaret for Product Backlog. Sammen har de tre roller ansvaret for Sprint Backlog og Burndown Chart.

Mødeformer

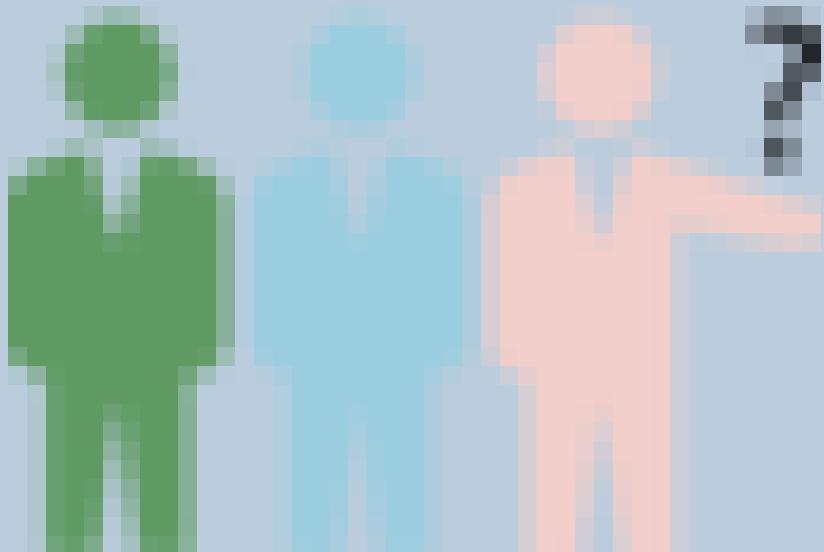
Mødeformerne er Sprint Planning, Daily Scrum og Sprint Review. Møderne bruges til at planlægge arbejdet med at udvikle it-systemet.

Sprint Planning



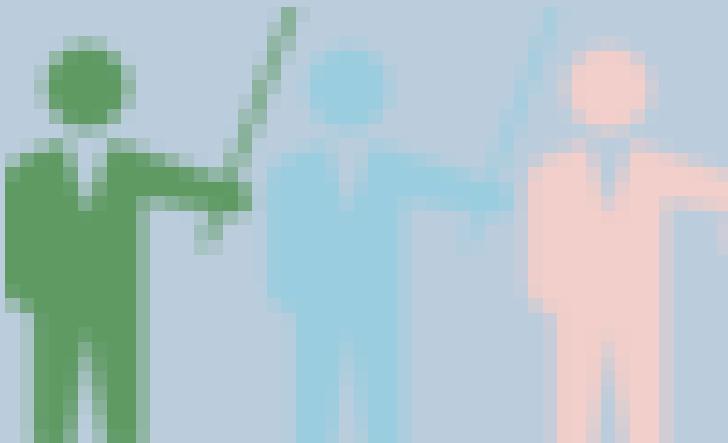
Sprint Planning-mødet bruges til at definere, hvad der skal laves i næste sprint, og hvordan det skal gøres.

Daily Scrum



Daily Scrum-mødet er det daglige møde, der afholdes under et sprint. Her fortælles, hvad der er lavet siden sidst, og hvad der skal laves inden næste Daily Scrum-møde.

Sprint Review



Sprint Review-mødet afholdes efter et sprint. Her undersøges, om man i sprintet nåede det ønskede.

Alle tre roller – Product Owner, Team og Scrum Master – deltager i ovenstående møder.

Arbejdet med at udvikle et it-system starter med, at Product Backlog bliver udfyldt med en liste af krav, beskrivelser, rækkefølger og estimerer. Herefter følger en række sprints, og til sidst er it-systemet færdigt.

SPRINT

Hvert sprint starter med, at de tre roller er Product Owner, Team og Scrum Master mødes til et Sprint Planning-møde. På baggrund af Product Backlog tilrettelægger de et sprint, der beskrives i SprintBacklog.

Efter et sprint afholdes et Sprint Review-møde, hvor der gøres status over sprintet. Ud over de tre roller kan der være gæster udefra, der er med til at se, om sprintet giver anledning til, at der foretages ændringer i krav eller beskrivelse af det ønskede it-system. Dette skrives i Product Backlog.

I løbet af et sprint afholdes der dagligt Daily Scrum-møder. Her fortæller teammedlemmerne, hvad de har nået, og hvad de vil lave færdigt til næste Daily Scrum-møde.

Estimatet over, hvor meget tid der er behov for for at lave arbejdet i sprintet færdigt, skrives i Burndown Chart. Dermed kan man hele tiden holde øje med, om tidsplanen holder, eller der skal justeres.

2. Planlægning af et it-system



iStockphoto.com/IconicBestiary

Når man planlægger et it-system, går man igennem de fleste af nedenstående trin:

- brainstorm
- kravsspecifikation
- målgrupper

Trinene bør løbende dokumenteres. Ellers vil man i løbet af kort tid ikke kunne huske detaljer i arbejdet, og dermed bliver det svært at lave ændringer senere. Andre vil heller ikke kunne lave ændringer i systemet, hvis der ikke følger en dokumentation med.

Brainstorm



iStockphoto.com/switchpipipi

Ved arbejde med en idé til et IT-system er det godt at begynde med en brainstorm. Dette gælder både, hvis idéen er ens egen, eller hvis idéen kommer fra brugeren eller kunden.

Projektarbejdsgruppen (dem der skal løse opgaven) mødes for at blive inspireret til at løse projektopgaven. Alle ideer, input, meninger og drømme om slutproduktets udseende og funktion er velkomne. Der skrives ned, tegnes, snakkes, synges ...

Brainstorm-fasen resulterer i en usorteret liste af ideer. Listen er typisk kreativ og uden tekniske detaljer.

Fordelen ved at anvende brainstorm er, at man måske får vilde ideer, man ellers ikke ville været kommet på.

På dette trin kan man også anvende innovative arbejdsprocesser til idéudvikling.



Opgave: Brainstorm til en hjemmeside

Lav en brainstorm til en hjemmeside. Hjemmesiden skal være en reklame for budskabet "Red elefanterne, der bliver vanrøgt i Thailand".

Skriv 50 ord op, der falder dig ind, når i tænker på emnet.

Analyse

I planlægningen af et it-system kan man lave en grundig analyse. Der kan tages udgangspunkt i en [brainstorm](#). Ved denne analyse diskuterer man en række spørgsmål. F.eks.:

- Hvad beskriver den virksomhed eller det sted, hvor it-systemet skal bruges? - her kan inddrages en [målgruppeanalyse](#).
- Hvilke problemer opleves der?
- Hvilke funktioner ønsker man? Hvad er målet?
- Hvilke data er i spil?
- Hvad vil vi opnå? Hvad er kravene til it-systemet?
- Hvor mange penge har vi?
- Hvor lang tid har vi?
- Andre begrænsninger?
- Hvordan tilrettelægger vi arbejdet?
- Skitseforslag til løsninger

Svarene samles i en rapport, der danner baggrund for det videre arbejde. En egentlig beskrivelse af [kravsspecifikation](#) er en del af rapporten.

Eksempel: Lagkage-app

Vi ønsker at udvikle en app, hvor man kan bestille lagkager hos et bestemt konditori. Appen skal bruges af kunder, og vi antager at disse er så it-kyndige, at de kan (lære at) betjene en app.

På nuværende tidspunkt oplever kunderne, at der er lang kø i butikken, fordi andre kunder er længe om at beslutte sig for, hvad de skal have. Desuden er der kunder, som går forgæves, fordi den konkrete lagkage, de gerne vil købe, er udsolgt eller viser sig at indeholde en ingrediens, som de ikke kan tåle.

Der ønskes en app, hvor man kan bestille en lagkage til afhentning på et bestemt tidspunkt, og det skal være muligt at fravælge ingredienser i kagen såsom nødder eller ferskner. Ødelægges grundelementet i kagen, skal ingrediensen ikke kunne fravælges (f.eks. chokolade i en chokoladekage).

Der vil bl.a. være følgende data i systemet: brugernavn/password til kunden, kagerne, ingredienserne.

De mere konkrete krav til systemet er en log-on-funktion, en funktion hvor man kan vælge den konkrete lagkage, en funktion hvor man kan vælge afhentningstidspunkt o.l.

Vi har 10.000,- til systemet.

Vi har to måneder til at udvikle app-en.

App-en skal være en web-app.

Arbejdet tilrettelægges som eneste arbejdsopgave for den, der skal lave appen.

Der skal laves skitser til løsningen.



Øvelse: Museumsapp, som viser info om ting på museet "Fregatten Jylland"



iStockphoto.com/perreten

Lav en analyse af en mulig app til Fregatten Jylland. Appen skal bruges til at give information under et museumsbesøg.

Kravspecifikation

Når man laver kravspecifikation analyseres, hvilke behov det kommende it-system skal dække. Det gøres ved at kortlægge, hvad de kommende brugere forventer af it-systemet - f.eks. via interviews eller spørgeskemaer. Undersøgelsen kan gøre brug af brainstorming. F.eks. kan man spørge brugere om deres favorit blandt flere forskellige løsningsforslag.

Ud over brugerkrav til produktet kan der være krav fra kunden til f.eks. driftsomkostninger.

Behovene udmønter sig i en detaljeret række af krav. Listen med krav kaldes en kravspecifikation. Hvert krav i en kravspecifikation skal

- *være præcist formuleret*, så der ikke er uklarhed om, hvad kravet går ud på.
- *være testbart*, så det kan undersøges, om et slutprodukt faktisk opfylder kravet.
- *give mening* i forhold til produktet.

Hvis et krav ikke giver mening, bør det kasseres eller genovervejes. Tivlsomme krav kan udsættes for en "hvorfor?" -prøve: Er der ingen god grund til at have kravet, kasseres det.

Kravspecifikation fastlægger altså ikke alene, hvad projektet skal kunne, men *afgrænser* også projektet.

Udfordringerne er, at kravspecifikationerne kan ændre sig, mens it-systemet udvikles, både fordi man bliver klogere, men også fordi den virkelighed it-systemet skal bruges i, ændrer sig. Det kan også for brugere og kunder være svært at beskrive kravene til et helt nyt it-system.



Opgave: Kravspecifikation

Lav en kravspecifikation til opgaven "Hold min 18 års fødselsdag".

Vær præcis med hensyn til, at kravene skal være præcist formuleret, nemt at afgøre om de blev opfyldt og de skal give mening i forhold til en 18 års-fødselsdag.

Hvem er kravspecifikationen henvendt til?



Opgave: Udfordringer til kravspecifikation

Find et eksempel på et it-system, hvor det har taget så lang tid at udvikle det, at kravspecifikationen har ændret sig undevejs.

Find et eksempel på et it-system der, da det kom, var så nyskabende, at der kan have været problemer med at lave en klar kravspecifikation.

I forbindelse med udarbejdelsen af en kravspecifikation, er det en god ide, i samarbejde med brugeren, at lave en beskrivelse af arbejdsgange når it-systemet bruges. En sådan beskrivelse kaldes et brugsmønster.

Brugsmønster

Et brugsmønster er en beskrivelse af et scenarie, hvor en bruger interagerer med it-systemet.

Der bør både laves brugsmønstre over de forventede scenarier, men også over alternative gennemløb hvor der sker fejl.

Eksempel: Bilkørsel

Et brugsmønster for en køretur i bil kunne være:

1. tag bilnøglen op af lommen
2. stik nøglen i låsen og lås op
3. åbn døren
4. stig ind
5. stik nøglen i tændingen
6. kobl ud
7. drej nøglen
8. ...

Her interagerer køreren med en bil.

Et alternativt scenarie kunne være en beskrivelse af, hvordan man starter bilen, hvis man ingen nøgle har.



Opgave: Kakaoautomat

Beskriv brugsmønstret på det at trække en kop kakao i en automat.

Beskriv også et alternativt brugsmønster og et brugsmønster, hvor der sker en fejl.

Hjem interagerer med hvem i dette eksempel?

Bemærk, at man også kan beskrive brugsmønstre mellem to it-systemer.

Målgrupper



iStockphoto.com/Savaryn

Når man udvikler et it-system, er det vigtigt at kende sin målgruppe. Når dette er på plads, kan man målrette designet at it-systemet netop til denne målgruppe.

Der er flere typer af værktøjer til at inddæle befolkningen i målgrupper:

- klassiske *livsstilsundersøgelser*, der opdeler befolkningen i et mindre antal *segmenter*
- såkaldte *demografiske* værktøjer, der ser på, hvordan lokalområder og befolkningen i de forskellige postdistrikter er sammensat.
- endelig er der forskellige værktøjer, der kan anvendes til analyse af, hvordan forbrugerne rent faktisk handler.

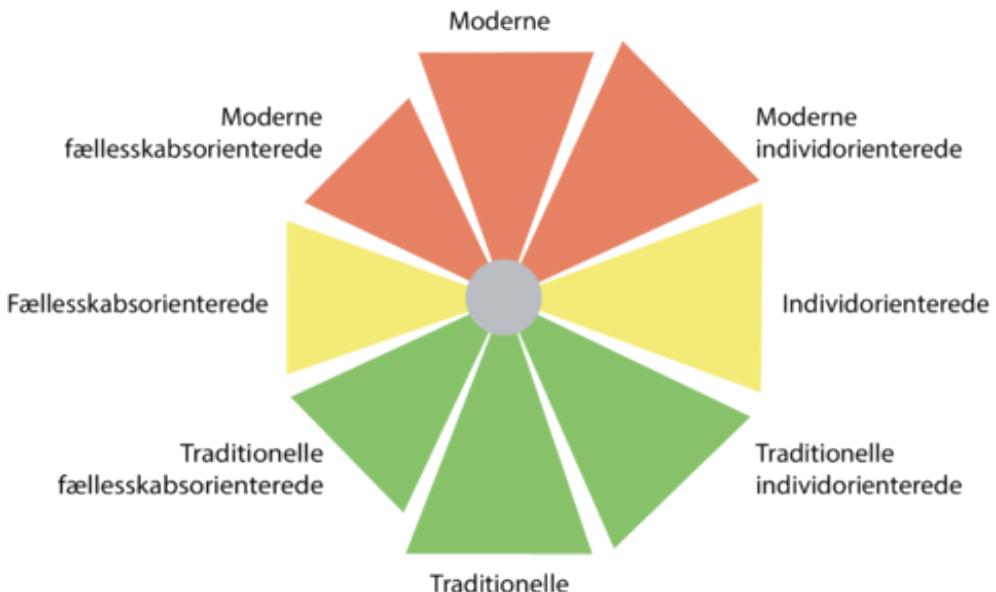
Gallup Kompas og Minerva-modellen

Gallup Kompas

Gallup Kompas er et omfattende "Media- og Marketings-informationssystem", der på baggrund af livsstilsundersøgelser inddeler befolkningen i otte segmenter:

- de *moderne*
- de *moderne-individorienterede*
- de *individorienterede*
- de *traditionelle-individorienterede*
- de *traditionelle*
- de *traditionelle-fællesskabsorienterede*
- de *fællesskabsorienterede*
- de *moderne-fællesskabsorienterede*.

Der er desuden et niende segment, *centergruppen*, som omfatter dem, der ikke kan placeres i de otte egentlige segmenter.



I denne udgave af Gallup Kompas vises også størrelsen af de enkelte segmenter (udgaven er fra 2006). Man kan fx se, at der er flere mennesker med traditionelle holdninger end med moderne.



Opgave: Gallup Kompas i dag

Gå på nettet, og find den aktuelle størrelse af de forskellige segmenter.

1. Er de moderne flere end de traditionelle i dag?
2. Hvordan har de fællesskabsorienterede udviklet sig?
3. Hvordan fordeler de to køn sig?
4. Hvordan stemmer segmenterne?

Gallup Kompas om alder

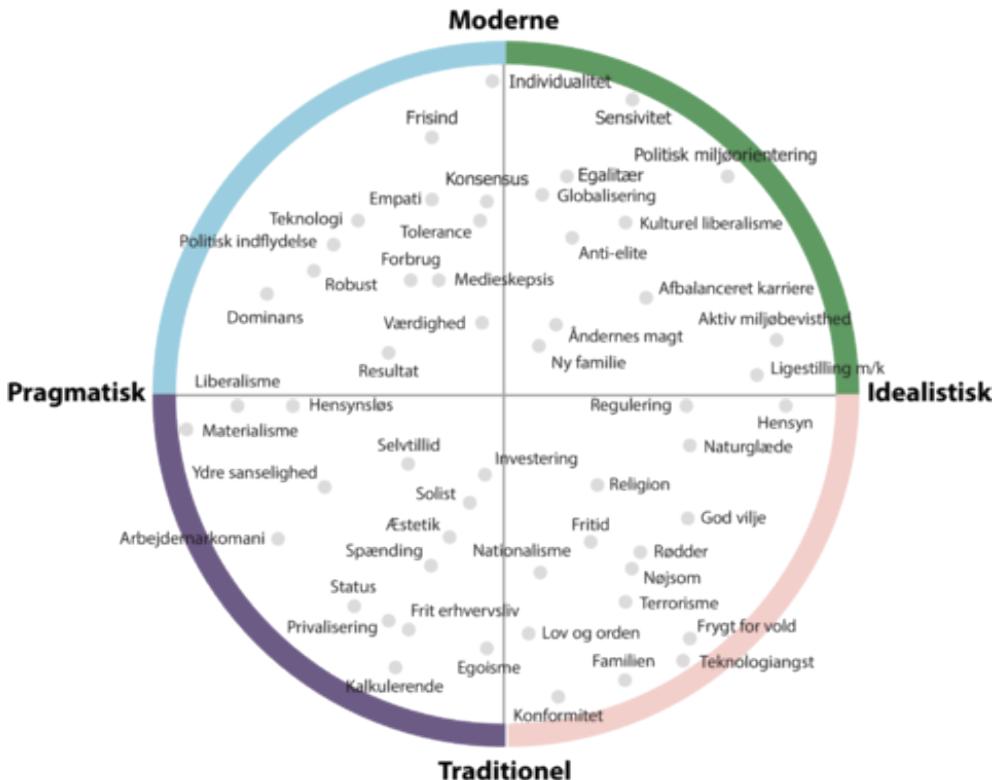
- De 13-19-årige fordeler sig ret spredt (de afspejler i høj grad forældrenes holdning), men generelt er de moderne. Og en stor del ligger i den diffuse centergruppe.
- De 20-29-årige er mere udpræget moderne, både moderne individorienterede og moderne fællesskabsorienterede.
- De 30-39-årige er ret moderne, primært kombineret med det individorienterede, men også med det fællesskabsorienterede.
- De 40-49-årige er i udpræget grad fællesskabsorienterede og moderne fællesskabsorienterede. En del er også moderne og/eller individorienterede samt traditionelt fællesskabsorienterede.
- De 50-59-årige er ret jævnt fordelt over segmenterne, men med en overvægt af de traditionelt orienterede segmenter.
- De 60-69-årige er udpræget traditionelle, både kombineret med det individorienterede og de fællesskabsorienterede.
- De over 70-årige er helt overvejende traditionelle, især kombineret med det individorienterede.

Minerva-modellen

En anden model er Minerva-modellen. Den bygger også på en cirkel, dog med kun fire hovedsegmenter plus den diffuse midtergruppe. Minervas fire hovedsegmenter kaldes for hhv. det *blå*, *grønne*, *rosa* og *violette* segment:

- det blå segment defineres af begreberne 'selvtillid og forbrug'
- det grønne defineres af begreberne 'engageret aktivitet'
- det rosa defineres af begreberne 'tradition, familie og det nære miljø'
- det violette defineres af begreberne 'stabilitet, tradition og gør-det selv'.

Modellen minder meget om Gallup Kompas. Gallups 'moderne' svarer til området, hvor Minervas blå og grønne mødes, og Gallups 'traditionelle' svarer til der, hvor det rosa og violette mødes. De 'individorienterede' svarer til dér, hvor det violette og det blå mødes, og de 'fællesskabsorienterede' svarer til dér, hvor det grønne og det rosa mødes.



Minerva-modellen.

Kilde: Henrik Dahl/ AIM/Nielsen Denmark

Opgave: Placér dig selv i Minerva-modellen

Find de ti værdier, der er vigtigst for dig.

Du vil sikkert opdage, at selvom de fleste værdier nok ligger nogenlunde tæt sammen, så er der nogle, der ligger mere spredt. Vi er ikke nemme at sætte i bås.



Opgave: Gallup Kompas og Minerva-modellen på nettet

Gå på nettet, og find en online målgruppetest, der placerer dig i Gallup Kompass eller i Minerva-modellen.

Er du placeret, hvor du mener, du skal være?



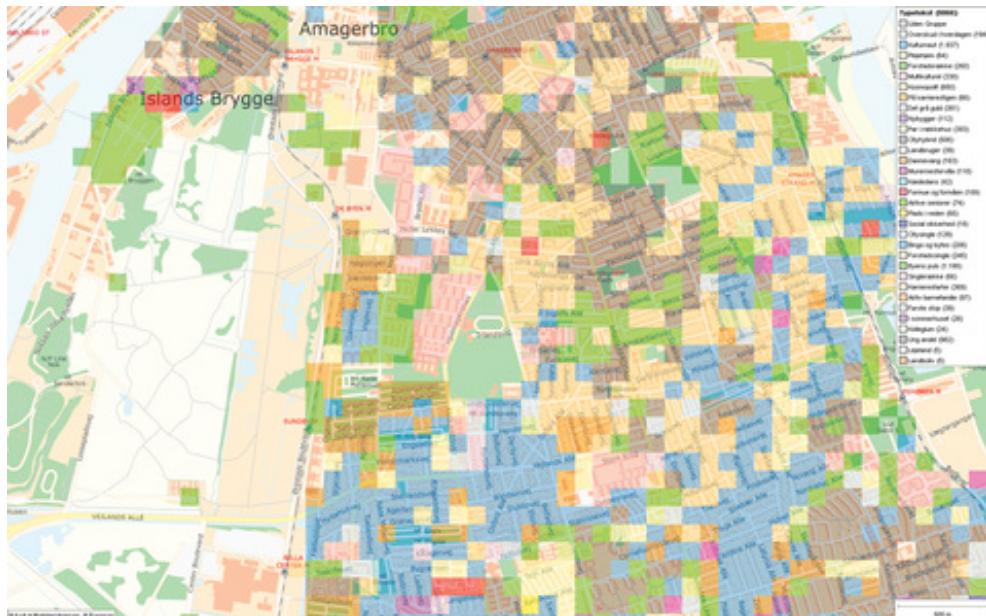
Opgave: Målgrupper med Gallup Kompas og Minerva-modellen

1. Du skal være med til at lave en ny tv-station, der især skal henvende sig til folk med moderne holdninger. Hvilken slags udsendelser og hvilke emner skal denne tv-station profilere sig på?
2. Du skal være med til at lave et nyt ugeblad, der især skal henvende sig til folk med traditionelle holdninger. Hvilken slags stof og hvilke emner skal dette ugeblad profilere sig på?

conzoom®

conzoom® er et såkaldt geodemografisk segmenteringssystem, dvs. et system, som ser på, hvilken slags mennesker der bor i forskellige områder. Modellen bygger på en sammenkobling af Gallup Kompas og oplysninger fra Danmarks Statistik. Med modellen lægger man et net med 430.000 firkantede celler ned over de danske geografiske bopælsområder; det går således helt ned på karréniveau. Cellerne bliver først delt op i otte forskellige typer, fx *Boligejere med overskud, Seniorer og Unge på vej*. Disse otte forskellige typer bliver yderligere opdelt i i alt 32 forskellige typer, fx *Aktiv børnefamilie, Kosmopolit og Karrierestarter*.

Modellen er altså både en geografisk model og en demografisk model.



Her kan man se, hvilke grupper der dominerer på det nordlige Amager. Bl.a. er det segmentet "Ung Andel", der spiller en hovedrolle her.

conzoom® om segmentet 'Første stop'

- Typen omfatter 1,92 % af befolkningen og 2,58 % af husstandene.
- 54 % spiser kylling mindst én gang om ugen – og generelt køber gruppen gerne økologiske dagligvarer.
- 32 % går i biografen mindst én om måneden – og de er generelt meget aktive mht. sport og fritid.
- 22 % læser M! – men generelt læser gruppen mange – og mange forskellige – tidsskrifter. Deres foretrukne avis er Information, men de læser også andre landsdækkende aviser.
- Deres mest populære tv-kanaler er TV2 Zulu, TV3 og TV3+; de er flittige internetbrugere, men derimod hører de ikke meget radio.
- De stemmer især på Det radikale Venstre, Socialdemokraterne, SF og Enhedslisten. Omvendt er de ikke særlig tilbøjelige til at stemme på Kristendemokraterne, Venstre eller Dansk Folkeparti.
- De bor mest i de større byer, foruden København er det især Albertslund, Slagelse, Roskilde, Odense, Svendborg, Ålborg, Aarhus, Holstebro, Herning, Esbjerg og Kolding. På Bornholm finder man dem især i Rønne og Nexø.

Data mining



iStockphoto.com/Jirsak

Ved data mining finder man en række forskellige data om befolkningens indkøbsvaner. Fx kan man analysere boner fra kasseapperaterne. På denne måde kan man afdække, hvad der købes sammen, fx øl og færdegretter – og vin og slik. Og derefter kan man forudsige salget af de varer, der bliver købt sammen.

Sådanne undersøgelser kan fx bruges til at beslutte temaer for nye kampagner eller placering af varerne i butikkerne.

Kundekort

I nogle supermarkedskæder, bl.a. COOP, kan man få kundekort. Det er umiddelbart en stor fordel, for kortet giver både adgang til særlige rabatter og til optjening af bonuspoint, der siden kan veksles til varer.

Det kan oven i købet laves så smart, at kunderne får særlige tilbud. Køber man mange økologiske varer, vil man få specielle tilbud på disse varer.

Kunder, der køber meget, får ekstra gode tilbud.

Facebook

Når du er på fx Facebook, kommer der reklamer på skærmen. Det er ikke tilfældige reklamer – de er skræddersyet din personlige profil. Den har du udfyldt helt frivilligt og sikkert uden at tænke over, hvad Facebook kan bruge dine data til.

Det er ikke nødvendigvis problematisk. Dine venner – og deres venner – må naturligvis godt vide, hvilket gymnasium du går på, hvem du evt. er i forhold med, om du er til mænd eller kvinder, hvilken religion du har, hvilken politisk holdning du har, hvilke film du synes bedst om osv. Det er ikke umiddelbart kontroversielle oplysninger, men faktisk hører både seksuel orientering og religion til den slags private oplysninger, der betragtes som personfølsomme.

Der er mange steder i verden, hvor en bestemt seksuel orientering eller religion kan være farlig. Det samme gælder også mange steder mht. politisk holdning. Det bør du naturligvis tænke på, når du udfylder din Facebook-profil.

Hvis din profil er helt offentlig, kan alle og enhver uden videre følge med – fx også din arbejdsgiver eller potentielle arbejdsgiver.

PET

PET og andre efterretningstjenester registrerer personlige data som en del af deres arbejde. Målet er at fange terrorister og andre, der er – eller kan tænkes at være – farlige for statens sikkerhed.



Opgave: For og imod data mining

Diskutér fordele og ulemper ved dataopsamling ved kundekort, Facebook og PET.

3. Udarbejdelse af et it-system



iStockphoto.com/TCmake_photo

Når man via f.eks. brainstorm, kravspecifikation og målgruppeanalyse har planlagt sit it-system, skal man i gang med at udarbejde det. Typisk vil man gennemgå nogle af følgende trin – som minimum programmeringsdelen:

- interaktionsdesign
- modellering
- programmering

Det er en god idé, sideløbende med planlægningen af et større it-system, at lave en brugervejledning. For mindre it-systemer kan man lave brugervejledningen i evalueringsfasen.

Man bør også dokumentere trinene løbende. Så bliver evalueringsprocessen nemmere og kortere.

Interaktionsdesign



iStockphoto.com/molotovcoktail og Systime

Ud over at sikre sig at det it-produkt man har designet løser den givne opgave, så er det vigtigt, at brugerne rent faktisk kan anvende det uden irritation eller følelsen af utilstrækkelighed. Man skal kunne gennemskue, hvad man skal gøre for at få systemet til at virke. Derfor bør man designe brugergrænsefladen og interaktionsdesignet, så brugeren bliver tilfreds med it-systemet. Dette arbejde tager bl.a. udgangspunkt i beskrivelsen af brugsmønstre.

Brugergrænseflade

Brugergrænsefladen er bindeleddet mellem en bruger og et it-system. Brugergrænsefladen modtager input fra brugeren og leverer output.

- Input kan modtages på mange måder. Det kan være via tastatur, knapper eller skærmtouch.
- Output kan være på skærm, som lyd eller som robotbevægelser.



Opgave: Forskellige it-systemers brugergrænseflade

På en computer kan man give input via fx tastatur og skærmtouch. Output kan være grafik på skærmen.

1. Hvad er input og output på en vaskemaskine?
2. Giv flere eksempler på it-systemer og de tilhørende input og output.

Interaktionsdesign

Interaktionsdesign er designet af de arbejdsgange, som en bruger skal benytte, hvis han eller hun skal anvende et it-system.



Opgave: Interaktionsdesign til en bil

Hvis vi betragter en bil som et it-system, hvilke arbejdsgange indgår så, når man skal bruge bilen?

Brugervenlighed

Et brugervenligt it-system opfylder følgende krav:

- Let at lære
- Let at huske
- Effektivt at bruge
- Forståeligt
- Tilfredsstillende at bruge



Opgave: Brugervenlighed

Find it-systemer, der overholde alle fem ovenstående krav til brugervenlighed.

Find it-systemer, der fejler på en eller flere af ovenstående krav.

Godt design af grænsefladen...

Dårligt design af grænsefladen...

| | |
|---|--|
| ... er opbygget med udgangspunkt i brugerens viden | ... er opbygget med udgangspunkt i specialviden, man ikke kan forvente af brugeren |
| ... er opbygget med udgangspunkt i genkendelse | ... er opbygget med udgangspunkt i <i>genkaldelse</i> |
| ... giver brugerne en fornemmelse af at vide, hvad de laver | ... forvirrer brugerne |

| Godt design af grænsefladen... | Dårligt design af grænsefladen... |
|--|--|
| ... lader brugeren anvende de typer input, brugeren foretrækker | ... skal bruges på én bestemt måde |
| ... er organiseret efter brugernes brugsmønstre: De mest almindelige funktioner er de lettest tilgængelige | ... er organiseret efter en dybere, men for brugere skjult logik: De funktioner, en bruger hyppigst ønsker at bruge, er ikke nødvendigvis nemme at komme til |

Grundelementer i brugercentrering.

Metoder til design af brugerflader

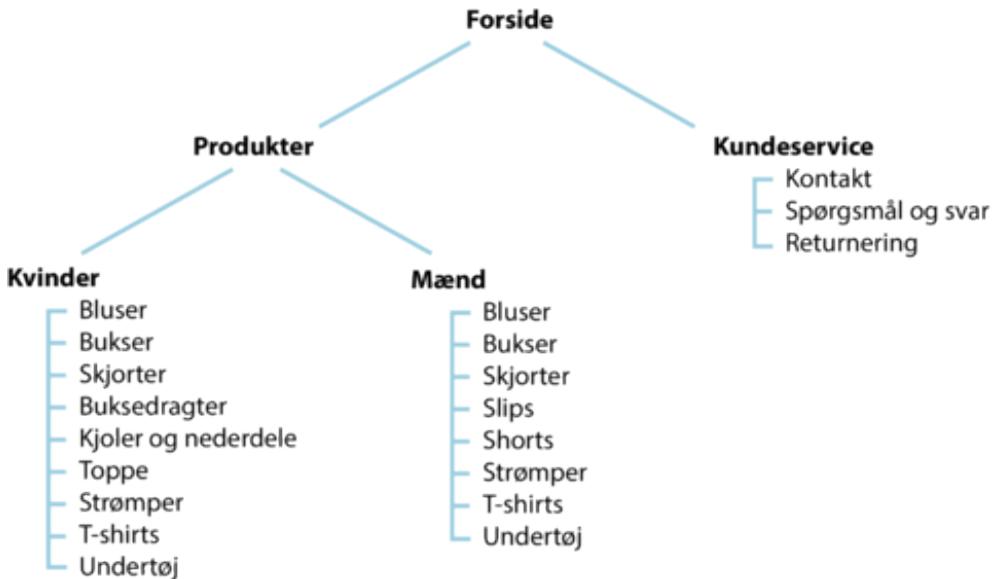
Der findes en lang række metoder, som kan hjælpe interaktionsdesigneren med at designe brugerflader. Her ser vi nærmere på diagrammer, skitser, wireframes og prototyper. Alle slags prototyper kan bruges til test af brugervenlighed. Om de er lavet af papir eller på computer, og om de ligner det færdige produkt eller ej, har ikke den store betydning for, hvilke fejl man kan finde i designet af brugerfladen.

Diagrammer

Diagrammerne er tegninger, der giver os mulighed for at se tingene i fugleperspektiv. De giver overblik og kan hjælpe os med at finde ud af, hvordan designet skal være. Derfor er diagrammer rare at arbejde med især i starten af et projekt, hvor det hele kan være lidt uoverskueligt.

Interaktionsdesignere bruger forskellige former for diagrammer. De vigtigste er strukturdia-grammer og flowdiagrammer.

Strukturdigrammer



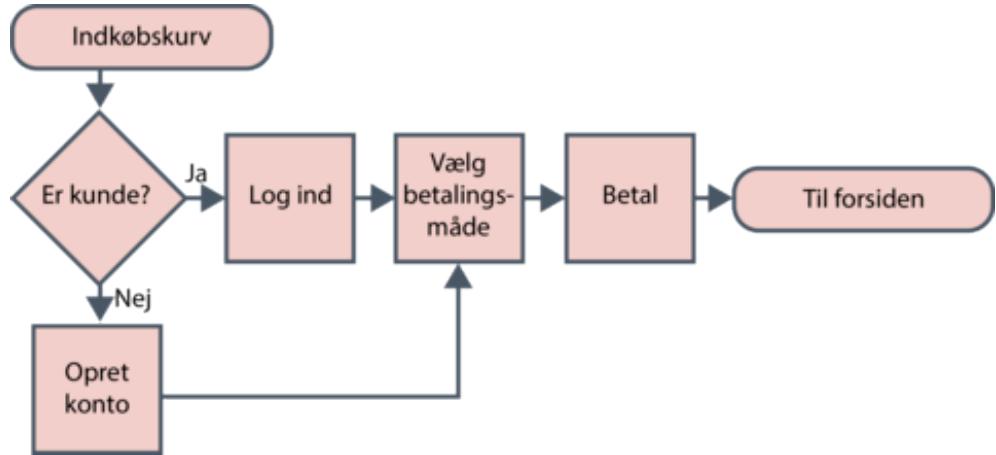
Strukturdiagrammer bruges mest til at planlægge, hvordan indhold på hjemmesider skal struktureres. En hjemmeside består ofte af en masse sider, som er hierarkisk struktureret i kategorier og underkategorier. I en online-tøjbutik, vil man fx kunne finde "jeans" under kategorien "bukser". Her kan strukturdiagrammet hjælpe os med at med at skabe overblik og få kategoriseret indholdet på en logisk måde, så folk kan finde det, de leder efter.



Opgave: DSB

Lav strukturdiagrammet for hjemmesiden dsb.dk.

Flowdiagrammer



Ikke alt er struktureret hierarkisk. Heller ikke på et website. I en online-butik vil der fx altid være en betalingsproces, hvor tingene sker i en bestemt rækkefølge. Fx kan man først få lov at betale, når man har skrevet sin adresse. Til at visualisere en sådan trinvis proces bruger man et flowdiagram.

Symboler i flowdiagrammer



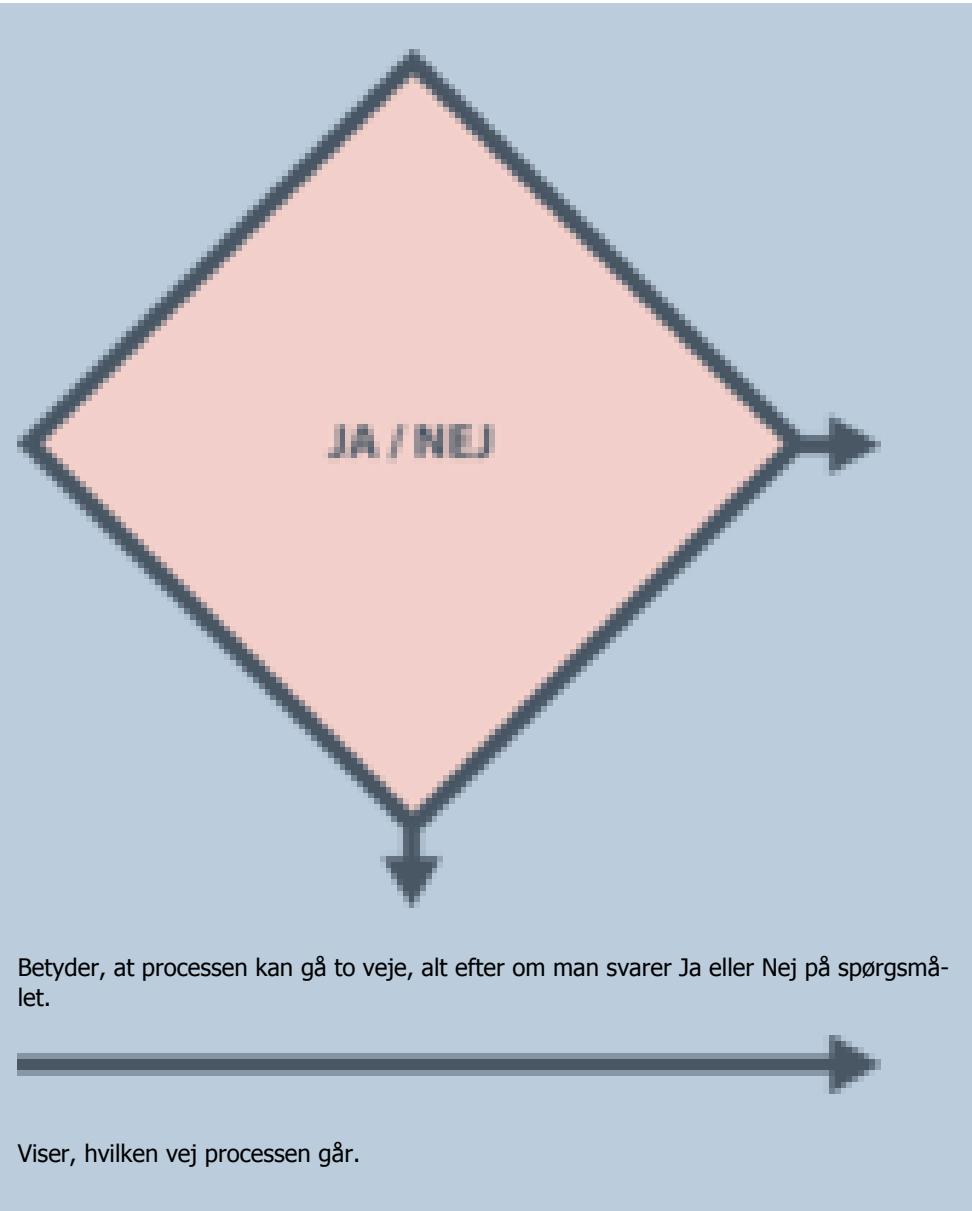
START / SLUT

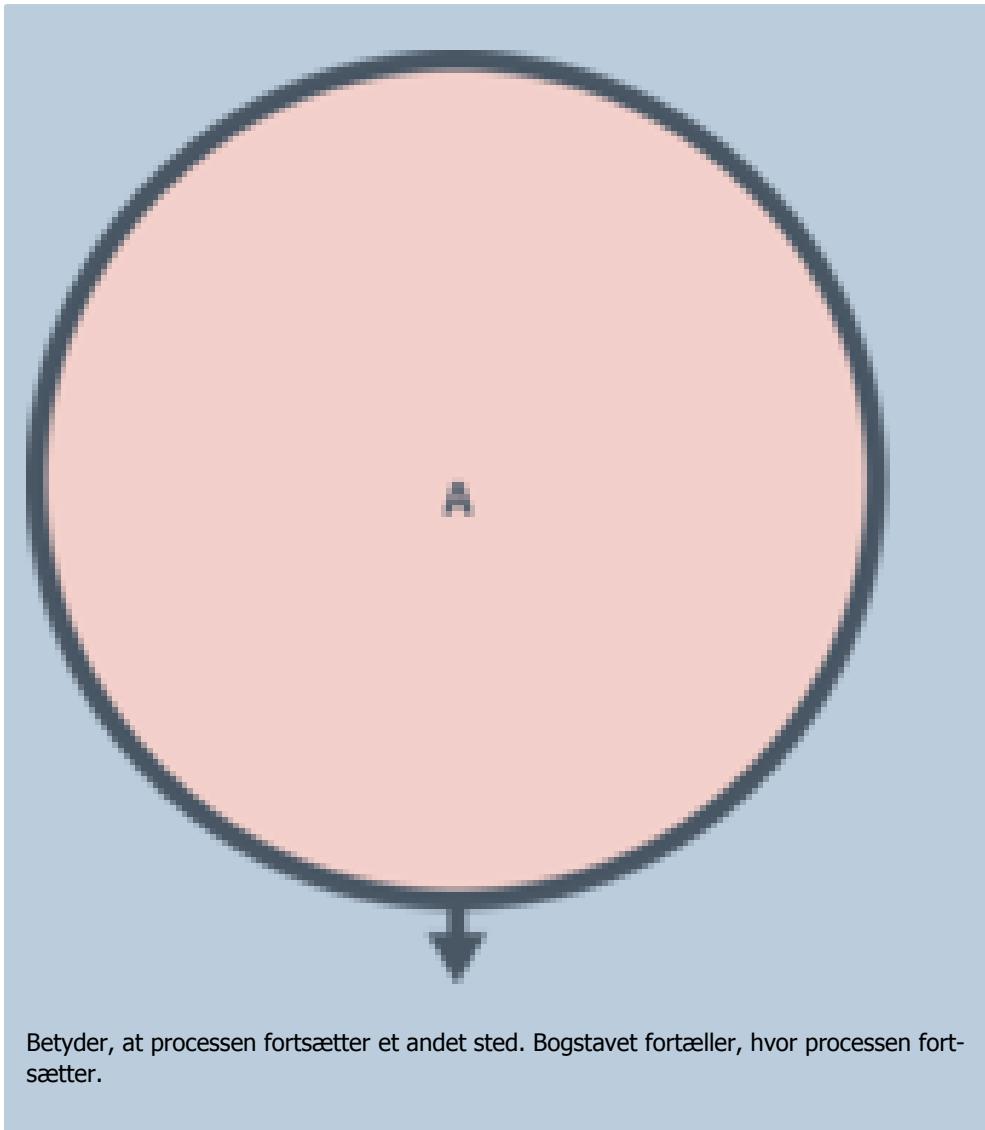
Bruges til at vise starten eller slutningen på en proces.



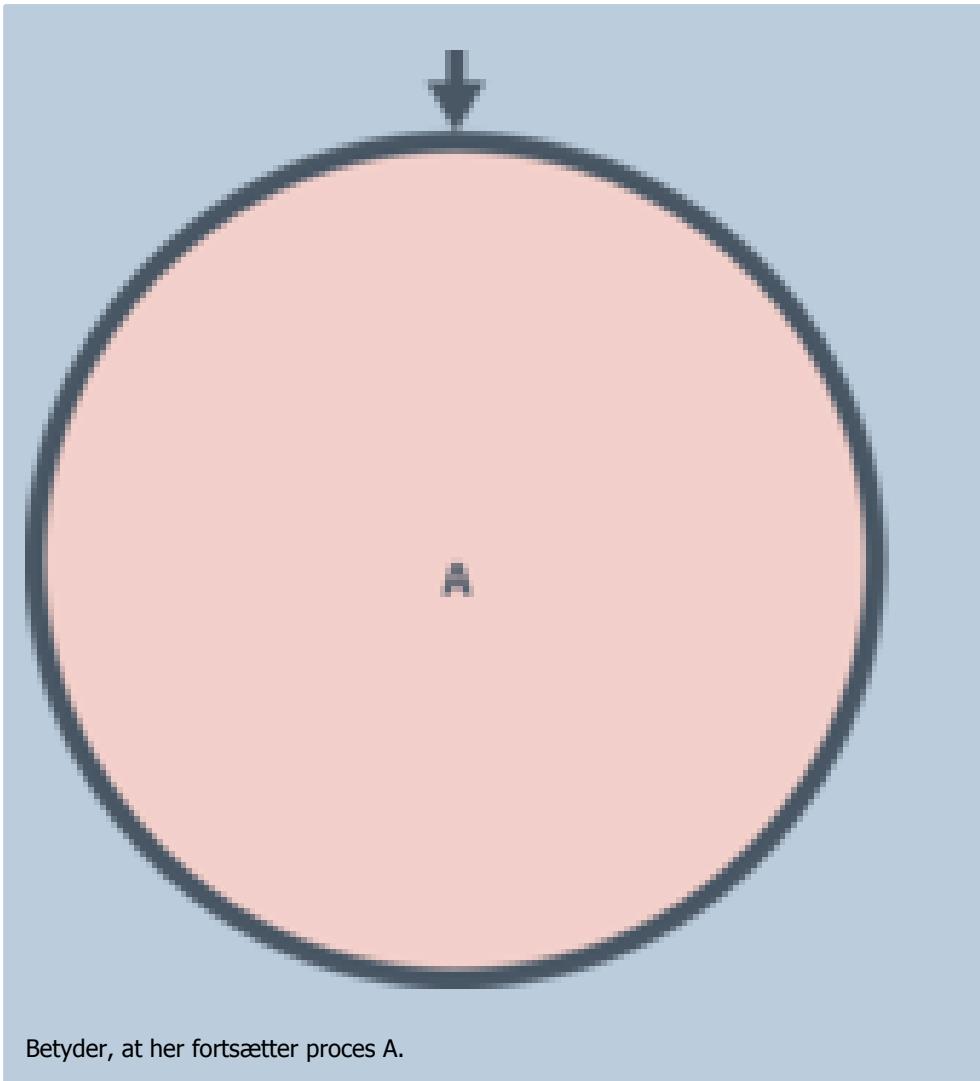
TRIN

Viser et trin i processen.





Betyder, at processen fortsætter et andet sted. Bogstavet fortæller, hvor processen fortsætter.



Betyder, at her fortsætter proces A.



Opgave: Morgenrutiner

Lav et flowdiagram, der viser, hvad du laver, fra du bliver vækket om morgenen, til du er kommet i skole.

Skitser

Det at lave skitser, før man går i gang med noget større, gør man inden for mange faggrupper. Håndværkere gør det, designere gør det, kunstnere gør det og interaktionsdesignere gør det.



Skitse.

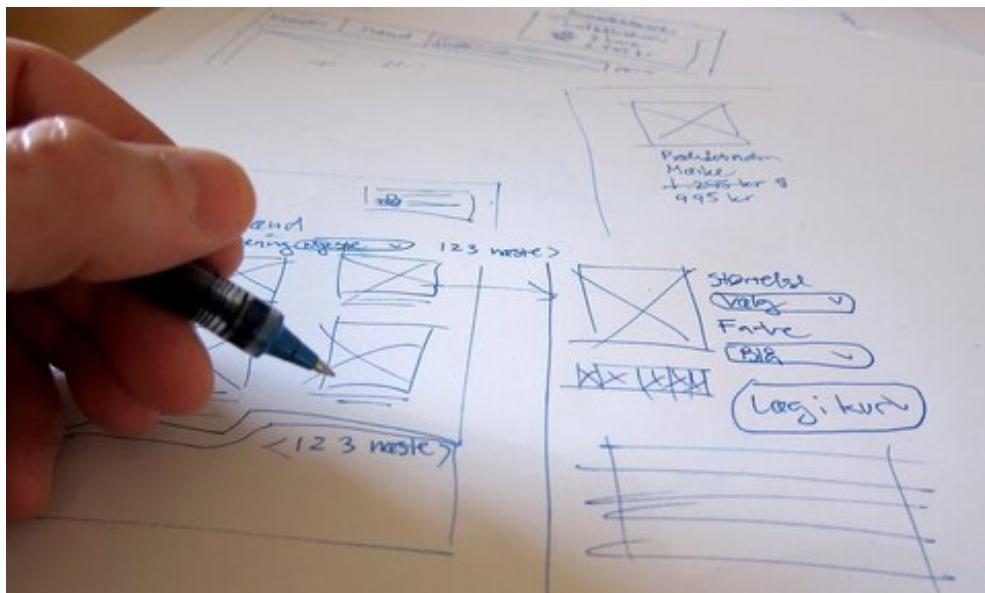
Raphael Santi: Forstudie til Saint Catherine of Alexandria, 1507. akg-images/Scanspixakg-images/Scanspix



Det færdige værk.

Raphael Santi: Saint Catherine of Alexandria, 1507. National Gallery, London, UK/Bridgeman Art Library/Scanpix

Interaktionsdesigneren bruger skitser til at eksperimentere med, hvordan en brugerflade skal opbygges, og hvordan interaktionen med produktet skal fungere.



Håndtegnede skitser er en hurtig måde at prøve ideer af på. At få ideerne ned på papir gør det meget nemmere at se tingene for sig og afgøre, om en idé er værd at arbejde videre med.

Skitser er også gode, når man arbejder sammen med andre. På et stykke papir eller en tavle kan man hurtige vise, hvad man mener.

Skitser bruges tit i starten af designprocessen til at får styr på ideerne, før man går videre med dem. Men de bruges også undervejs, når man hurtigt skal prøve noget af eller forklare en idé til andre.



Opgave: Skitse til spil

Lav på papir en skitse, der beskriver et computerspil.

Wireframes

Wireframes er skitser, som viser, hvordan et produkts brugerflade skal se ud.

Kundeservice

Indkøbskurv
2 varer
1.512,50 kr.

Forside Kvinder Mænd Indtast segeord... Søg

Bluser
Bukser
Skjorter
Shorts
Slips
Strømper
T-shirts
Undertøj

Jack & Jones Jeans 'Stan Osaka'
499,75 DKK

DND Jeans 'Buddha'
799,75 DKK 399,75 DKK

DND Jeans 'Buddha'
799,75 DKK 499,75 DKK

Tiger of Sweden 'Pisseleng'
Jeans
1.199,75 DKK

Indicode Jeans
499,75 DKK 369,75 DKK

Diesel 'Thaval' Jeans
1.199,75 DKK

Brunn & Stengade
'Copenhagen' jeans
499,75 DKK 399,75 DKK

Solid jeans
499,75 DKK 299,75 DKK

G-Star 'Arc 3D slim' jeans
1.049,75 DKK

Wireframes er tit lavet på computer. Men de kan også sagtens være tegnet i hånden.

Man kan kende wireframes på, at de ligner rigtige brugerflader – bare uden farver, billeder og andre grafiske finesser.

Tit er wireframes forsynet med noter, som forklarer brugerfladens opbygning, og hvordan de interaktive dele skal fungere.

Wireframes bruges især til at forklare andre, hvordan det færdige produkt skal fungere. Fx kan grafikere bruge wireframes til at se, hvordan skærbilledernes layout skal opbygges. Og programmører kan bruge dem til at sætte sig ind i, hvordan tingene skal virke.

Wireframes er en meget populær måde at visualisere designet af en brugerflade på. Der er bare det lille problem, at wireframes ikke er interaktive. Det kan derfor være svært at forstå, hvordan de interaktive dele af designet vil fungere. Det er prototyper til gengæld gode til.

Prototyper

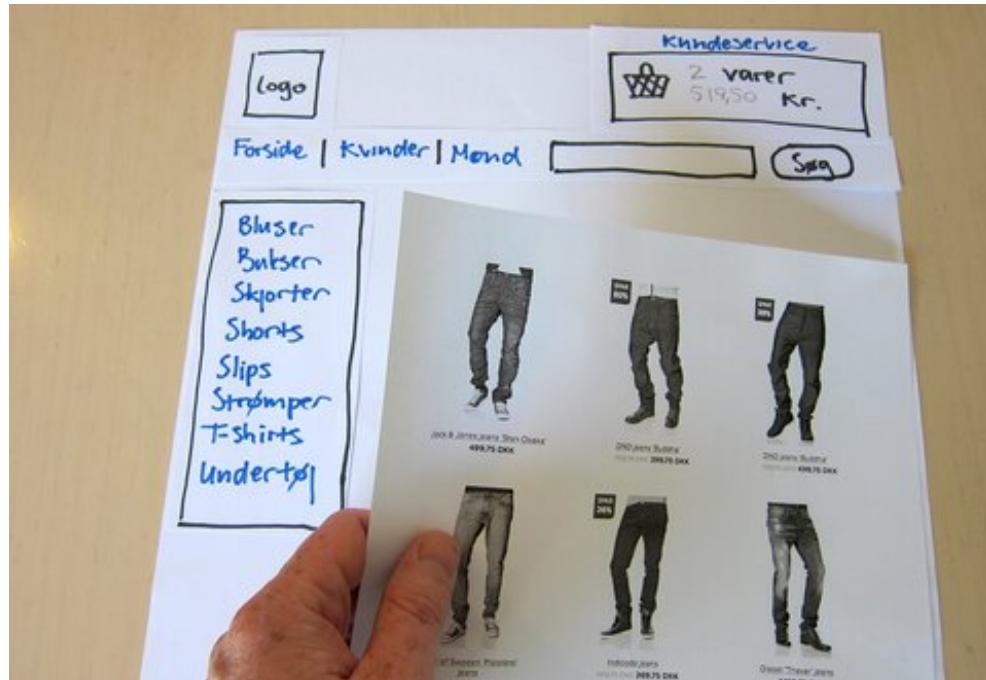
En prototype er en model, man bygger for at eksperimentere med designet af et fremtidigt produkt.

Til forskel fra skitser og wireframes er en prototype ikke bare en tegning på et stykke papir, men noget som rent faktisk fungerer. En prototype er interaktiv og giver os mulighed for at opleve, hvordan en brugerflade vil fungere i virkeligheden. Dermed kan den også bruges til at teste designet af brugerfladen med rigtige brugere.

Prototyper kan være meget forskellige. Nogle er tegnet i hånden. Andre er lavet på computer.

Nogle kan en hel masse. Andre kan kun en lille smule af det, det færdige produkt skal kunne.

Prototyper af papir er de letteste at lave. En papirprototype er tegnet i hånden. Interaktivitet simulerer vi med håndkraft. Hvis vi fx forestiller os, at en bruger klikker på et link, udskifter vi bare en tegning af den nuværende side med en tegning af den side, brugeren gerne vil se.



Papirprototype af online-tøjbutik, hvor brugeren har valgt "bukser" i menuen.

Fidelity

Ordet "fidelity" er engelsk og betyder "nøjagtighed". Man bruger ordet fidelity til at beskrive, hvor meget en prototype ligner det færdige produkt, og i hvor høj grad prototypen kan simulere de ting, som det færdige produkt skal kunne.

- En high-fidelity-prototype er en prototype med stor nøjagtighed. Den vil typisk være lavet på computeren og vil føles som et færdigt produkt.
- En low-fidelity-prototype er en prototype med lav nøjagtighed. Den vil typisk være tegnet i hånden og være lavet på papir.

Medium-fidelity-prototyper er alle de prototyper, som hverken er low-fi eller hi-fi, men et sted midt imellem.

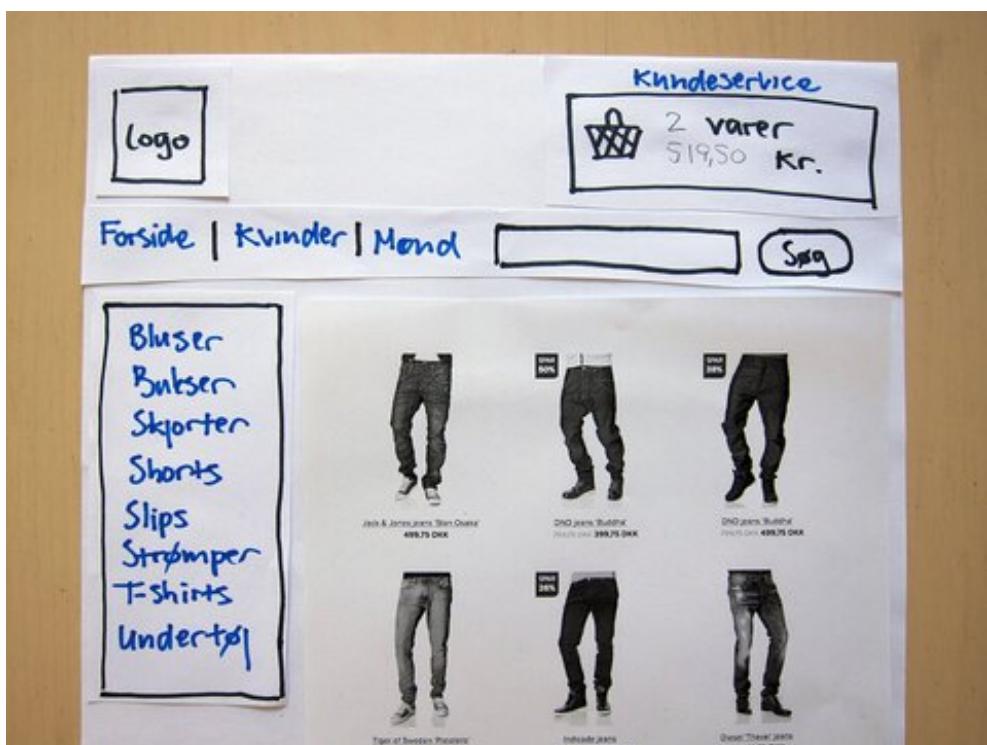
Sådan laver du papirprototyper

Før du går i gang, skal du skaffe følgende ting:

- Et stykke karton, der fungerer som vores skærm. Har du ikke noget karton, kan et stykke papir også bruges. Det er en fordel at arbejde i en lidt større målestok end i virkeligheden. En "skærm" i A3 eller større er praktisk, når det er et computerprogram eller et website, vi skal designe. A4 er passende til en app til mobiltelefon.

- Sort tusch eller kuglepen til at tegne brugerfladen med. Det kan være en fordel at have både en tynd og en tyk.
- Blå tusch eller kuglepen til at tegne links og andre ting, man kan klikke på.
- Almindeligt papir i rigelige mængder. Det bruger vi til at tegne brugerfladens elementer på.
- Saks til at klippe brugerfladens elementerne ud med.
- Blyant, som bruges, når der skal "indtastes" oplysninger i formularfelter.
- Viskelæder til "fejlindtastninger", og når vi skal begynde forfra.
- Post-it-sedler (eller "gule sedler", som de også kaldes). Kan bruges til mange ting, fx fejlmeldelser og drop down-menuer.
- Limstift med samme slags lim som på post-it-sedler. De er meget praktiske, når vi skal placere et stykke papir på skærmen, som hurtigt skal kunne fjernes igen. Men de specielle limstifter er desværre svære at opdrive. Kan du ikke finde dem, kan du sagtens klare dig uden.

På det stykke karton eller papir, som vi bruger som skærm, opbygger vi selve brugerfladen. Det gør vi ved at tegne brugerfladens elementer på papir, klippe dem ud og placere dem på skærmen.



Papirprototype af af online-tøjbutik.

Er det et website, kan brugerfladen fx bestå af en menu i toppen og et indholdsområde nedenfor. Disse elementer klipper vi ud hver for sig og placerer på skærmen.

Fordelen ved at opbygge brugerfladen på denne måde er, at når en bruger klikker på et link i menuen, behøver vi kun at ændre indholdet. Menuen bliver bare siddende, hvor den er.

Når en papirprototype bruges til test af brugervenlighed, er der en person, som leger computer. Denne person sørger for, at prototypen reagerer på brugerens handlinger. Normalt er der desuden én, som er testleder og én, som tager noter. Det er testlederens opgave at guide testdeltageren gennem opgaverne. Personen, som tager noter, sørger for at skrive ned, når testdeltageren har problemer.



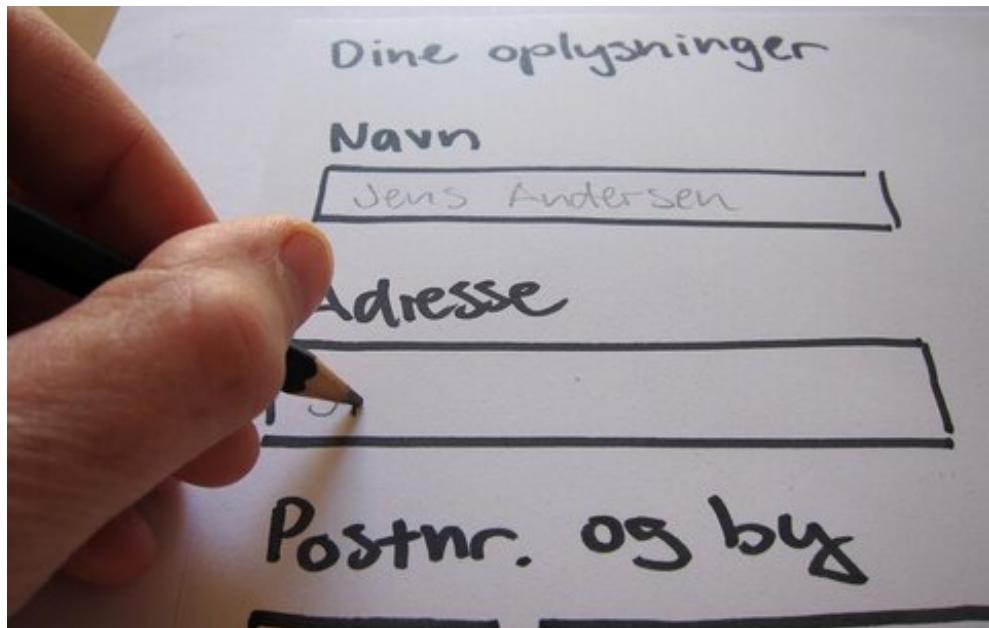
En papirprototype bliver testet. Manden til venstre er "computeren".



Papirprototype i brug under workshop.

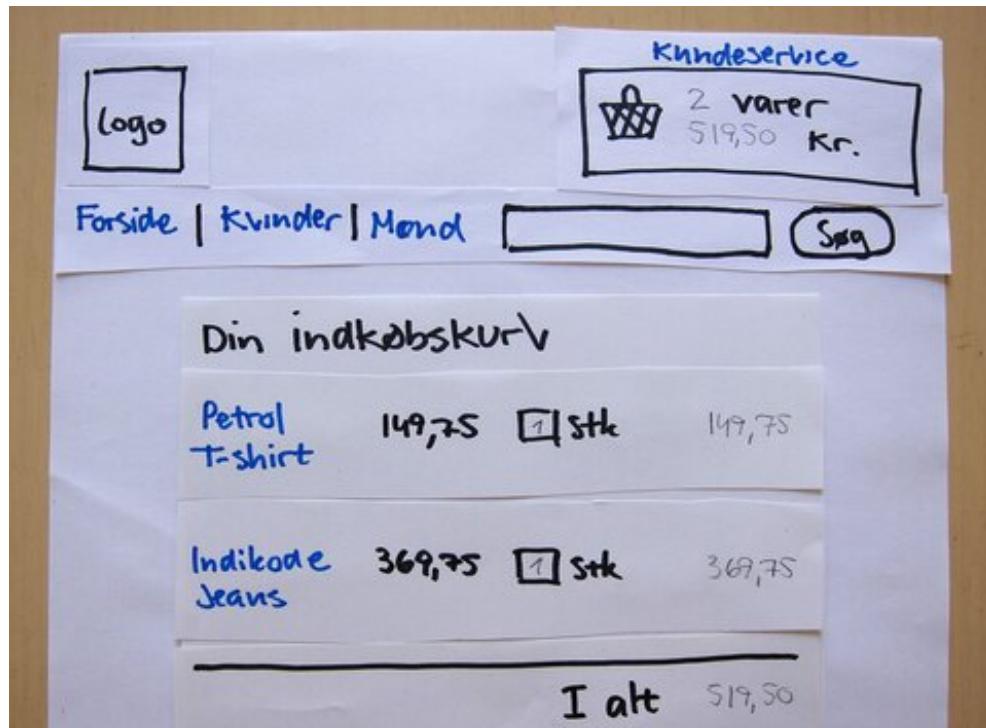
Selvom de virker lidt primitive, kan vi faktisk simulere meget avancerede funktioner med papirprototyper – noget, der nogle gange kan være svært og tidskrævende med prototyper lavet på computeren.

Fx kan vi nemt simulere indtastning af data. Skal brugerne fx oplyse deres adresse i en online-butik, kan de "indtaste" oplysningerne med blyant. Skriver de forkert, kan det rettes med et viskelæder. Og når en ny bruger skal teste løsningen, kan vi viske det hele ud og begynde forfra.



Indtastning af oplysninger kan nemt simuleres med en papirprototype.

Vi kan også nemt simulere en indkøbskurv i en online-butik. Hvis en bruger vælger at putte et produkt i indkøbskurven, tilføjer "computeren" bare produktet til indkøbskurven med blyant. Er der flere produkter i indkøbskurven, regner "computeren" den samlede pris ud, visker den gamle pris ud og skriver den nye.



Indkøbskurv simuleret ved hjælp af en papirprototype

Metoder til vurdering af brugerflader

Ligesom der findes metoder til at designe brugerflader, findes der også en række metoder til at vurdere deres brugervenlighed på. Her ser vi på metoderne inspektion og brugervenlighedstest.

Inspektion



iStockphoto.com/vladwel

Inspektion går ud på, at et antal eksperter i brugervenligt design gennemgår et produkts brugerflade for at vurdere, om den lever op til en række principper for godt design. Det kan fx være et par enkle principper som disse:

- Giv brugerne nem adgang til de funktioner og informationer, de har brug for.
- Tal et sprog, som målgruppen forstår.
- Gør det indlysende, hvad produktet kan.
- Gør det indlysende, hvordan produktet skal bruges.
- Lad ikke brugerne være i tvivl om, hvad der sker (fx hvis noget tager lang tid).
- Gør fejlmeddelelser forståelige, og fortæl folk, hvad de skal gøre for at rette fejlen.
- Giv altid brugerne mulighed for at fortryde det, de har gjort.

Under inspektionen vurderer hver ekspert produktets brugerflade på egen hånd. De gennemgår for det meste produktet to gange. Én gang, hvor de ser på hvordan produktet hænger sammen som helhed. Og én gang, hvor de kigger nærmere på de mindre detaljer – fx ordlyden i en fejlmeddeelse.

Under inspektionen laver eksperterne hver en liste over mulige problemer. Bagefter sammenligner de deres lister og bliver enige om, hvad der bør rettes.

Inspektioner kan gennemføres med prototyper eller færdige produkter. Men der er store fordele ved at gøre det så tidligt som muligt, da det er meget nemmere og dermed også billigere at rette en fejl i en prototype end i et færdigt produkt.

Inspektion er ikke den sikre vej til et brugervenligt design. Når brugerfladen bliver vurderet af eksperter og ikke af rigtige brugere, kan der ske fejl. Eksperterne kan fx vurdere, at noget er et problem, selvom det ikke er et problem for de rigtige brugere. Eller eksperterne kan helt overse ting, som de rigtige brugere har svært ved. Derfor er inspektion bedst som et supplement til brugervenlighedstest med rigtige brugere.

Brugervenlighedstest

I en brugervenlighedstest tester vi it-systemet med potentielle brugere for at se, om folk kan finde ud af at bruge det.

Under testen får hver deltager en række opgaver, som de skal løse. Det kan fx være at købe et par bukser i en online-butik.

Deltagerne skal løse opgaverne på egen hånd. Lederne af testen skal bare lytte, observere og skrive ned, når deltagerne har problemer. Deltagerne får kun hjælp, hvis de sidder helt fast og ikke kan komme videre.

Mens deltagerne løser opgaverne, skal de tænke højt. Det vil sige, at de skal fortælle,

- hvad de ser,
- hvad de vil gøre,
- og hvad de forventer, der vil ske.

På den måde kan man bedre forstå, hvad der går galt, når der er noget, deltagerne ikke kan finde ud af.

Test af brugervenlighed kan gennemføres med både prototyper og færdige produkter. Ligesom ved inspektion kan det være en fordel at gennemføre tests så tidligt i udviklingsforløbet som muligt. Det er nemlig meget nemmere at rette en prototype end at rette det færdige produkt.

Sådan laver man opgaver til brugervenlighedstest

Her er en række tips til hvordan man laver gode testopgaver:

- **Opgaverne skal være korte og lette at forstå.** Deltagerne skal helst forstå opgaven med det samme og ikke være nødt til at læse den igen for at huske, hvad den gik ud på.
- **Første opgave skal helst være nem.** Folk, der deltager i en test, kan nogle gange være lidt nervøse for, om de kan finde ud af det. Hvis de nemt kan løse den første opgave, kan det få dem til at slappe af.
- **Opgaverne må ikke indeholde skjult hjælp.** Deltagerne skal selv finde ud af, hvordan de skal løse opgaverne. Tester man fx et firmas hjemmeside, skal man ikke bede deltagerne om at klikke på linket "Karrieremuligheder". I stedet skal man bede deltagerne om at se, om de kan finde ledige job på hjemmesiden. Deltagerne kan jo netop have svært ved at regne ud, at de kan finde ledige job under "Karrieremuligheder". Bruger man ordet "Karrieremuligheder" i opgaven, har man afsløret det hele.
- **Opgaverne skal være realistiske.** De skal handle om noget, folk kan finde på at gøre i virkeligheden. Man skal ikke bede deltagerne om at klikke rundt på en hjemmeside, som de har lyst. Der er ingen, der går ind på en tilfældig hjemmeside uden at ville et eller andet.
- **Opgaverne skal være relevante.** De skal handle om noget, som deltagerne selv kunne finde på at gøre. Tester man en online-tøjbutik, er det bedre at bede deltagerne finde et par bukser, de selv kunne finde på at købe, end at bede dem finde et bestemt par bukser. Når folk selv skal vælge, er de nemlig mere kritiske, end hvis de skal gøre noget på kommando.

Sådan forløber en brugervenlighedstest

1. **Velkomst.** Vi starter med at byde deltagerne velkommen til testen og takker dem for, at de vil deltage. Derefter fortæller vi lidt om, hvad testen går ud på - men uden at fortælle noget, der kan afsløre, hvordan opgaverne skal løses. Det er vigtigt at fortælle deltagerne, at det ikke er dem, vi tester, men produktet. Vi tester ikke for at se, hvor gode de er til at bruge en computer eller en smartphone. Vi tester for at se, om folk kan finde ud af at bruge det produkt, vi tester.
2. **Indledende interview.** Det er meget normalt at stille deltagerne et par spørgsmål, inden vi går i gang med opgaverne. Spørgsmålene går typisk ud på at finde ud af, om deltagerne kender det produkt, vi tester, eller om de kender lignende produkter. På den måde kan vi få en fornemmelse af, hvor "øvede" deltagerne er.
3. **Opgaver.** Så er vi klar til at gå i gang med selve opgaverne. Deltagerne får én opgave ad gangen. Normalt læser vi først opgaven op for deltagerne. Derefter får deltagerne opgaven på et stykke papir. Så kan de selv læse den, hvis de får brug for det.
4. **Efterfølgende interview.** Normalt sluttet der af med et kort interview om, hvad deltagerne synes om produktet. Her giver vi dem også mulighed for at komme med kommentarer og stille spørgsmål.
5. **Afslutning.** Til sidst takker vi deltagerne for, at de ville være med i testen. Normalt får deltagerne til en lille gave som tak.

Opgave til interaktionsdesign



Colourbox.com

Alle opgaver i dette afsnit går ud på at designe og evaluere en app til mobilen, hvor man kan bestille pizza.

Appen skal kunne følgende:

1. Man skal kunne vælge, hvilket pizzeria man vil bestille fra
2. Man skal kunne bestille én eller flere pizzaer
3. Man skal oplyse, hvor man vil have pizzaerne bragt hen
4. Man skal betale for pizzaerne

Formålet med opgaverne er at prøve nogle design- og evalueringsmetoder af.

Begræns også udvalget at pizzerier og pizzaer for at holde det simpelt. Hav fx fem pizzerier, og lad hvert pizzeria sælge de samme fem pizzaer.



Opgave: Lav et diagram over app'en

Lav et diagram over, hvordan app'en skal hænge sammen. Du får brug for både

- et strukturdiagram til valg af pizzeria og pizza
- og et procesdiagram til betalingsprocessen.



Opgave: Skitsér din app

Lav skitser af, hvordan appens brugerfladen skal se ud, og hvordan brugerne skal kunne vælge pizzeria og pizza, oplyse leveringsadresse og betale. Tegn skitserne på papir eller på en tavle.



Opgave: Byg din egen prototype

Lav en prototype af pizza-app'en. Det kan være smart at lave prototypen lidt større end i virkeligheden – fx så den fylder en A4-side.



Opgave: Lav en inspektion

Har du lavet en prototype af pizza-app'en, kan du udsætte den for en inspektion. Husk, at der skal være en, som skal være "computer".

Har I ikke lavet en prototype, kan I udsætte et website, et computerprogram eller en app for en inspektion.



Opgave: Lav en brugervenlighedstest

Hvis I har lavet en prototype af pizza-app'en, skal I lave en brugervenlighedstest.

Har I ikke lavet en prototype, kan I gennemføre en brugervenlighedstest af et website, et computerprogram eller en app.

Gestaltlovene

OBS

Dette er kernestof på B-niveau.

Gestaltlovene er en række principper, der beskriver hvordan vi opfatter det vi ser. Her er beskrevet fem af principperne(lovene). Lovene beskriver hvordan vi opfatter elementer på

en flade. Det kan for eksempel være knapper, dropdown-menuer eller billeder på en hjemmeside.

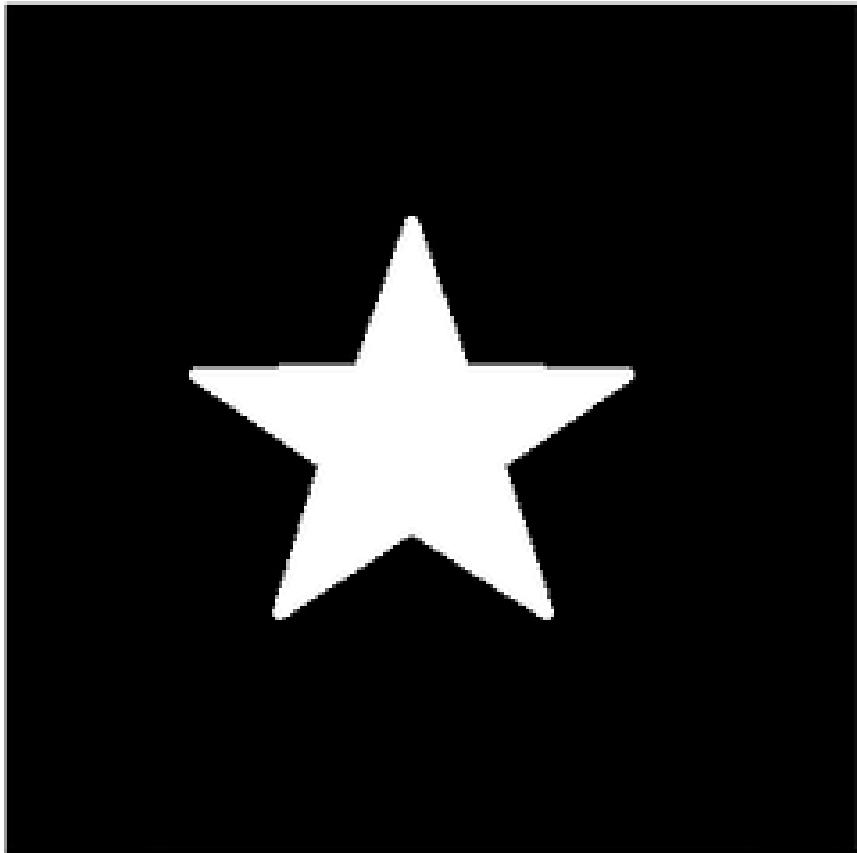
Gestaltlovene kan bruges, når vi laver brugergrænseflader. Hvis gestaltlovene er overholdt, opfatter brugeren nemmere de ideer it-udvikleren har med en hjemmeside. Det bliver også nemmere for brugeren at forstå hvordan udvikleren har forestillet sig at it-systemet skal bruges.

Loven om figur og baggrund

Den mindste figur på en flade opfattes som vigtigst og vil blive opfattet først. Den største flade opfattes som baggrund.

For at overholde denne lov, er det selvfølgelig vigtigt, at det tydeligt kan ses, at der er en forgrund og en baggrund. Dette kan sikres med en tydelig kontrast.

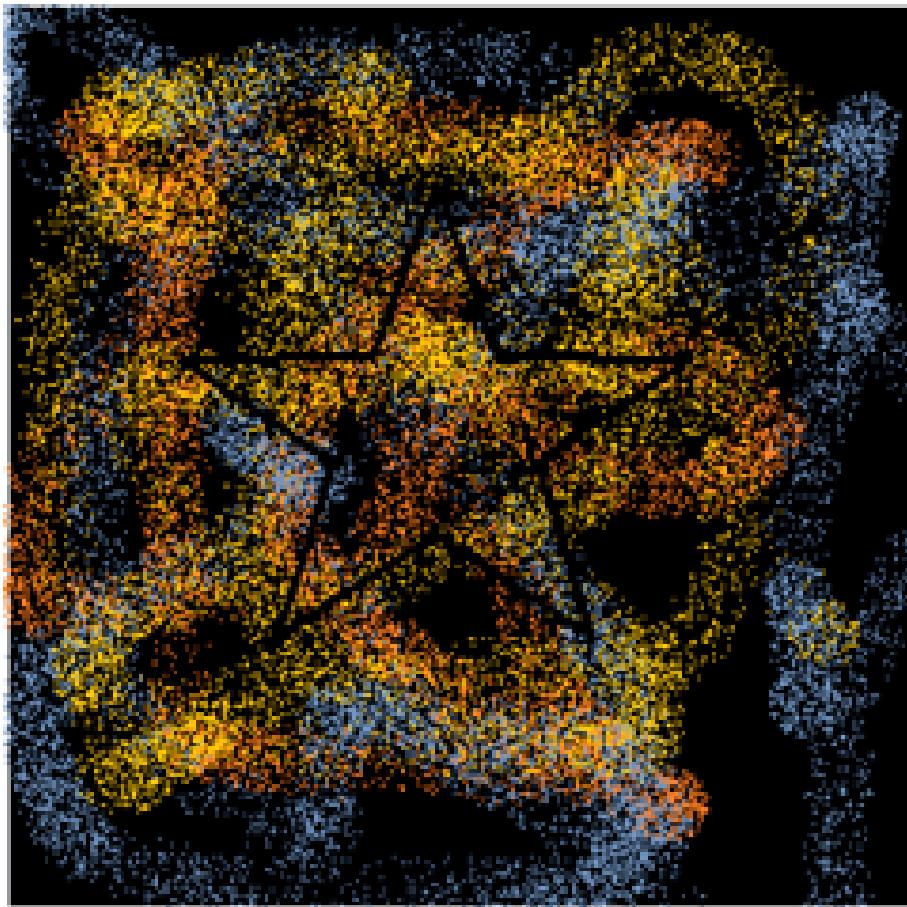
Eksempel: Loven om figur og baggrund



Her ses et eksempel hvor loven er overholdt. Det er tydeligt at den hvide stjerne er mindst og den træder frem.



Her er et eksempel hvor man kommer i tvivl, fordi de to figurer fylder næsten lige meget (Rubins vase – efter den danske psykolog Edgar Rubin). Er det et billede af en vase eller et billede af to silhuetter? Læg mærke til at vi ikke kan se vasen og ansigterne samtidig.



Her er et eksempel hvor baggrunden er så urolig at det er svært at se forgrunden:



Opgave: Loven om figur og baggrund

Gå på nettet og find en hjemmeside hvor loven om figur og baggrund er overholdt. Det kan for eksempel være en knap, der er tydelig forgrund og dermed nem at finde.

Find en hjemmeside hvor baggrunden er så urolig, at det er svært at se den vigtige forgrund. Forgrunden kan for eksempel være en knap der er svær at se fordi den falder i med baggrunden.

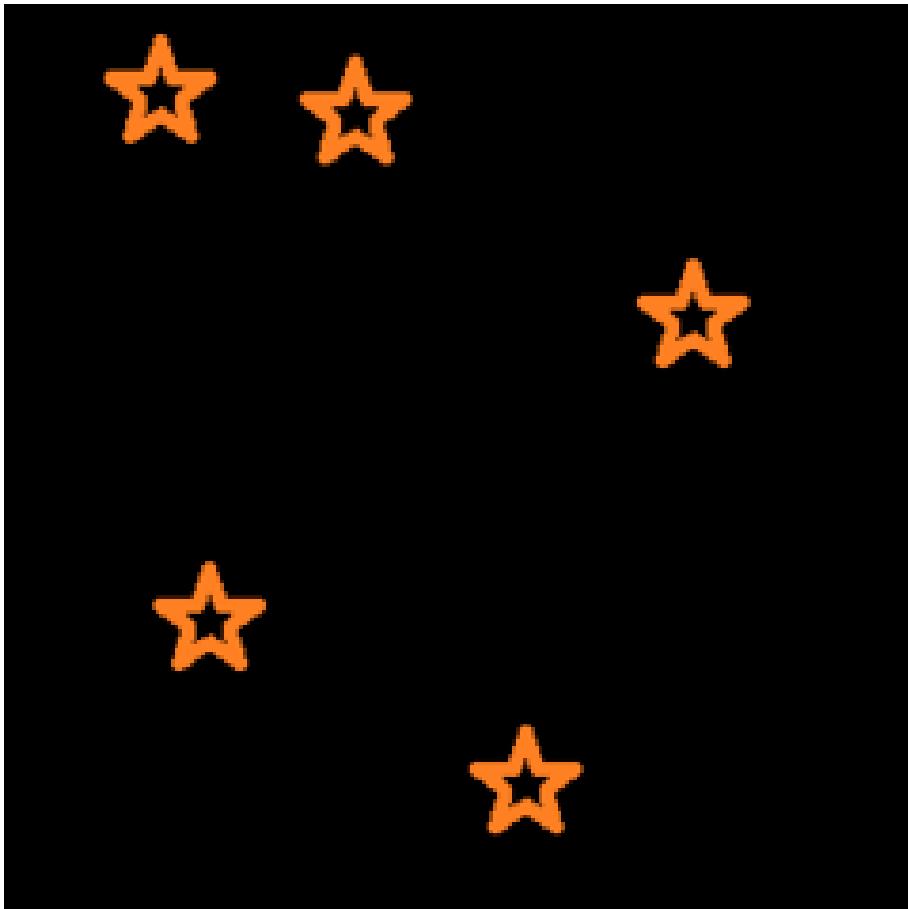
Loven om nærhed

Elementer, der er placeret tæt på hinanden, opfattes som hørende sammen.

Eksempel: Loven om nærhed



Her ses det at loven om nærhed er overholdt. De tre stjerne øverst til venstre hører sammen og de to stjerner nederst til højre hører sammen.



Her er et eksempel hvor man kommer i tvivl: Der er tre stjerner, der hører sammen. Er det de tre stjerner længst til venstre der hører sammen eller er det de øverste stjerner der hører sammen? Eller hører de to stjerner øverst til venstre sammen og så er det de sidste tre stjerner der hører sammen?



Opgave: Loven om nærhed

Gå på nettet og find en hjemmeside hvor loven om nærhed er overholdt. Det kan for eksempel være billede og billedtekst, der står tæt sammen.

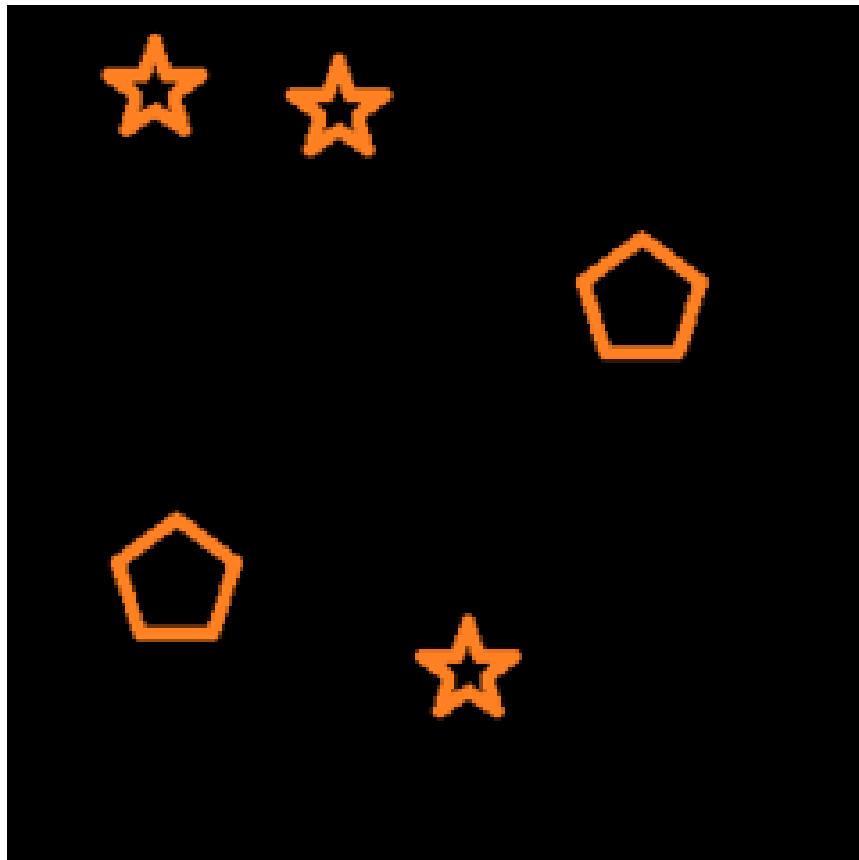
Find en hjemmeside, hvor loven om nærhed ikke er overholdt.

Loven om lighed

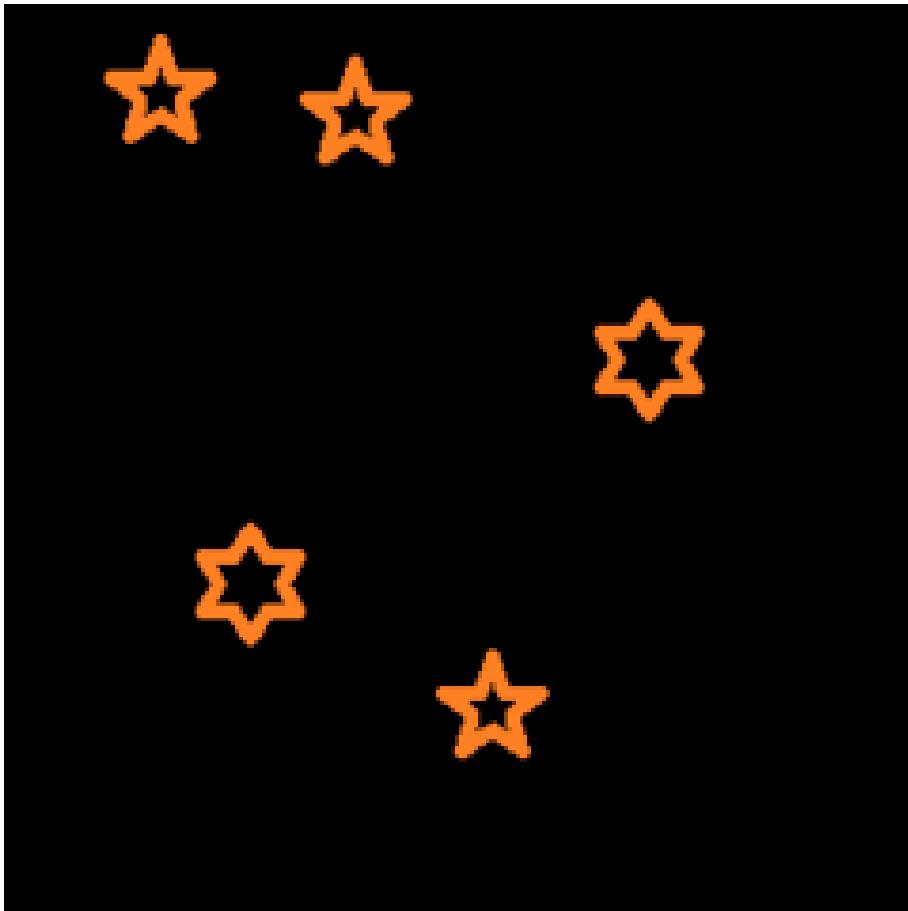
Elementer der ligner hinanden hører sammen.

Derfor er det vigtigt at der er afstand til andre elementer.

Eksempel: Loven om lighed



Her ses et eksempel på loven om lighed. De tre stjerne hører sammen og de to femkanter hører sammen.



Her er et eksempel hvor figurerne ligner hinanden meget og derfor er det svært at se hvilke elementer der hører sammen.



Opgave: Loven om lighed

Gå på nettet og find en hjemmeside hvor loven om lighed er overholdt. Det kan for eksempel være menupunkter med samme skriftype og størrelse.

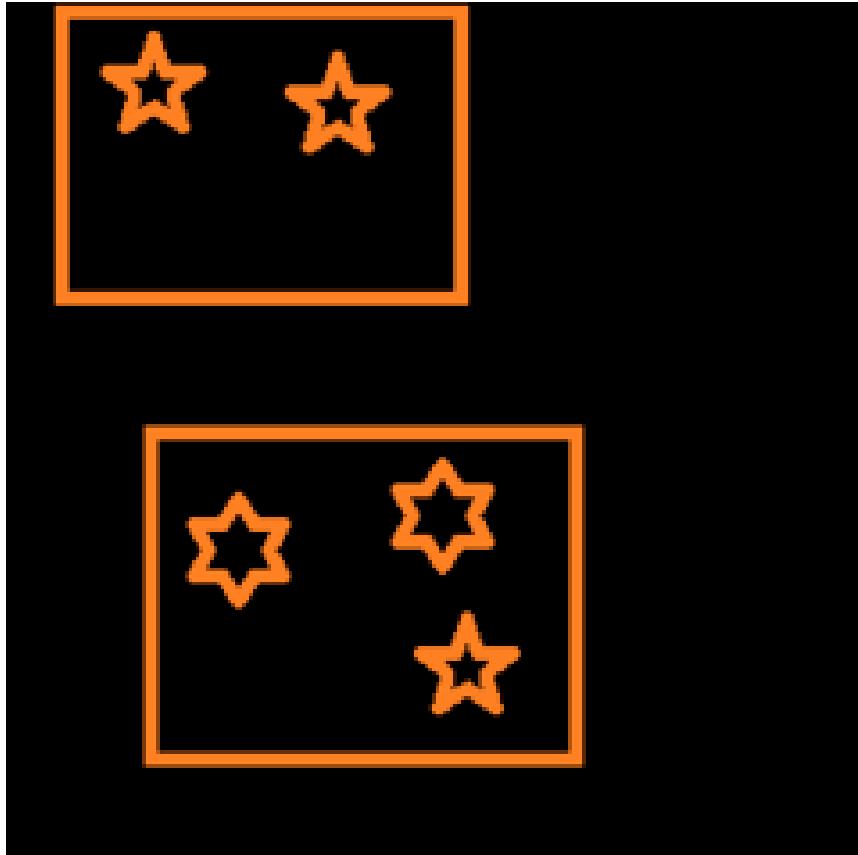
Find også en hjemmeside hvor loven om lighed ikke er overholdt.

Loven om lukkethed

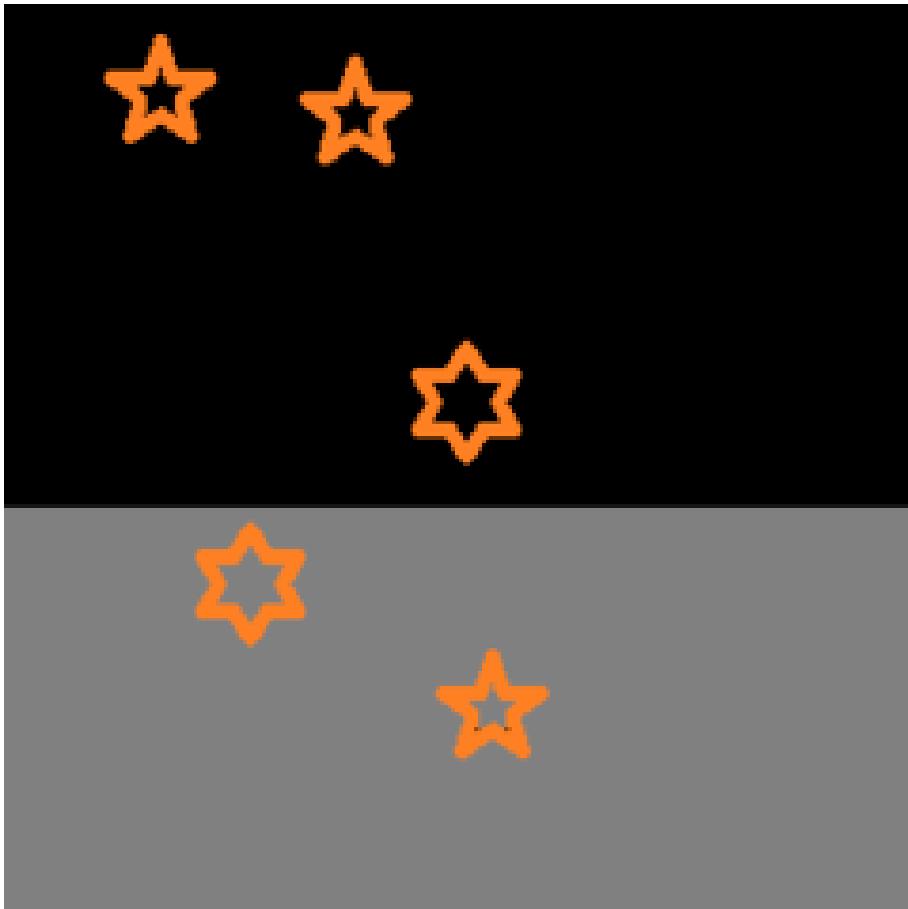
Elementer der er placeret inde i samme ramme hører sammen.

Loven om lukkethed gælder også hvis elementerne er placeret på samme baggrundsfarve.

Eksempel: Loven om lukkethed



Her ses et eksempel på loven om lukkethed. Rammerne sikrer at selvom elementerne er forskellige, så er det tydeligt hvilke der hører sammen.



Her er et eksempel hvor det er baggrundsfarven der bestemmer hvilke elementer der hører sammen.



Opgave: Loven om lukkethed

Gå på nettet og find en hjemmeside hvor loven om lukkethed er overholdt.

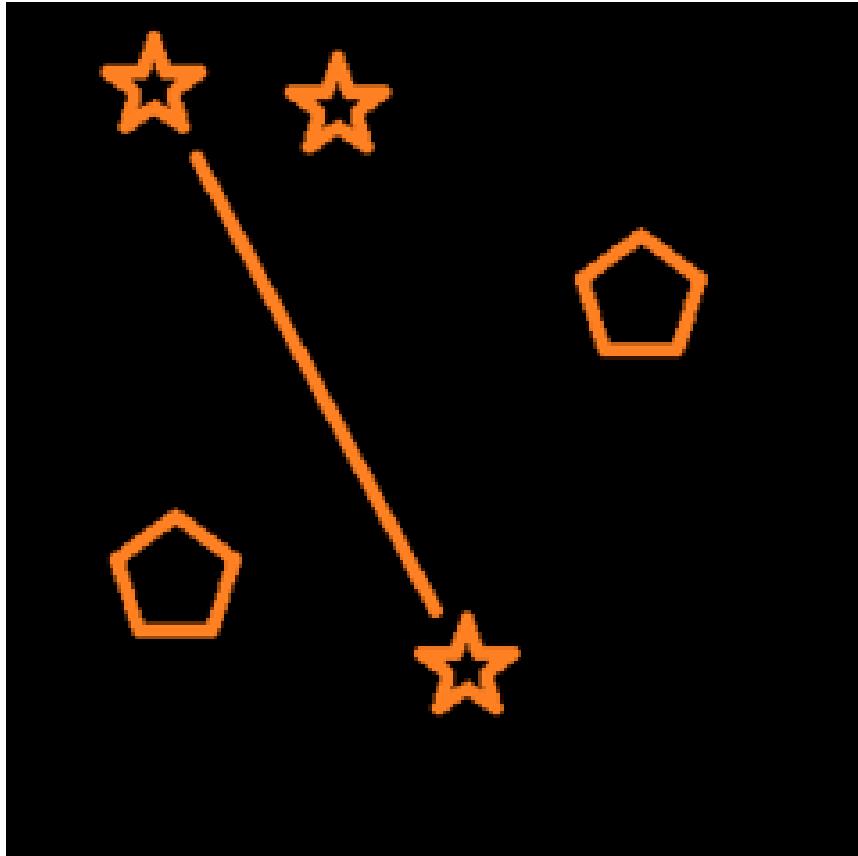
Find også en hjemmeside hvor loven ikke er overholdt.

Loven om forbundethed

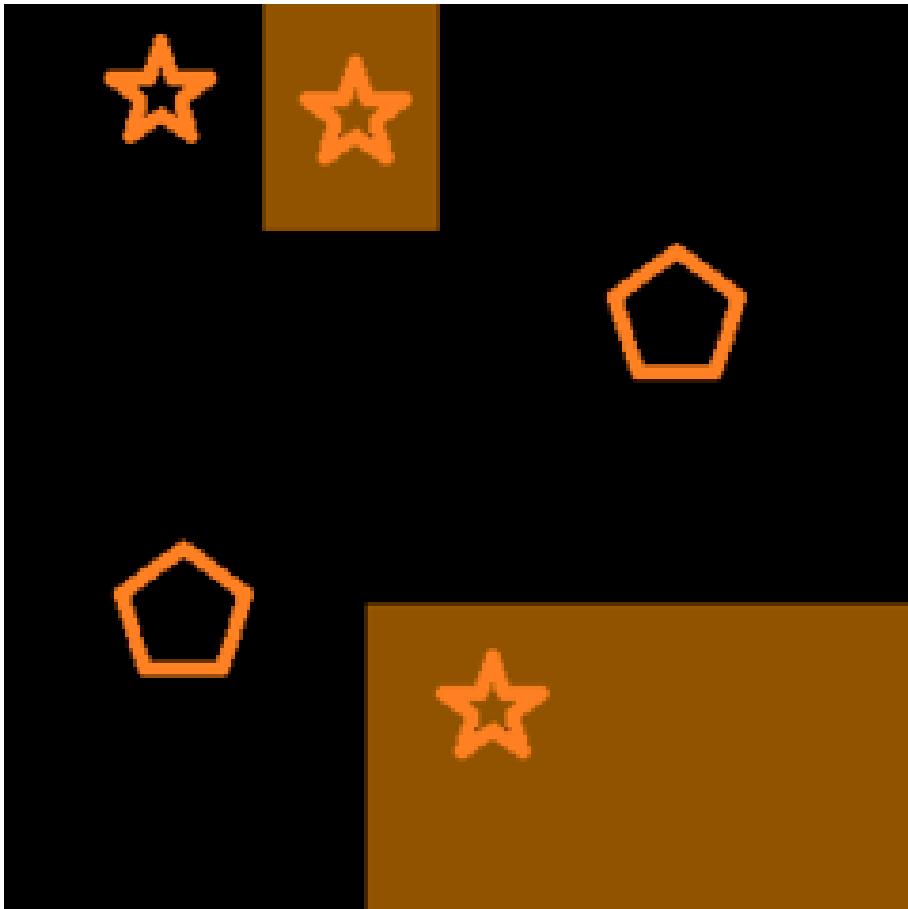
Elementer der er forbundet hører sammen. Elementerne kan være forbundet med stregen eller farver.

Loven bruges typisk hvis elementer hører sammen selvom de ikke er placeret tæt sammen.

Eksempel: Loven om forbundethed



Her er et eksempel hvor loven om forbundethed er overholdt. De to stjerner er forbundet med en streg.



Her er et eksempel hvor loven om forbundet hed er overholdt. De to stjerne på orange baggrund hører sammen.



Opgave: Loven om forbundethed

Gå på nettet og find et eksempel hvor loven om forbundethed er overholdt.

Find også et eksempel hvor to elementer langt fra hinanden hører sammen, men hvor man ikke umiddelbart kan se det.

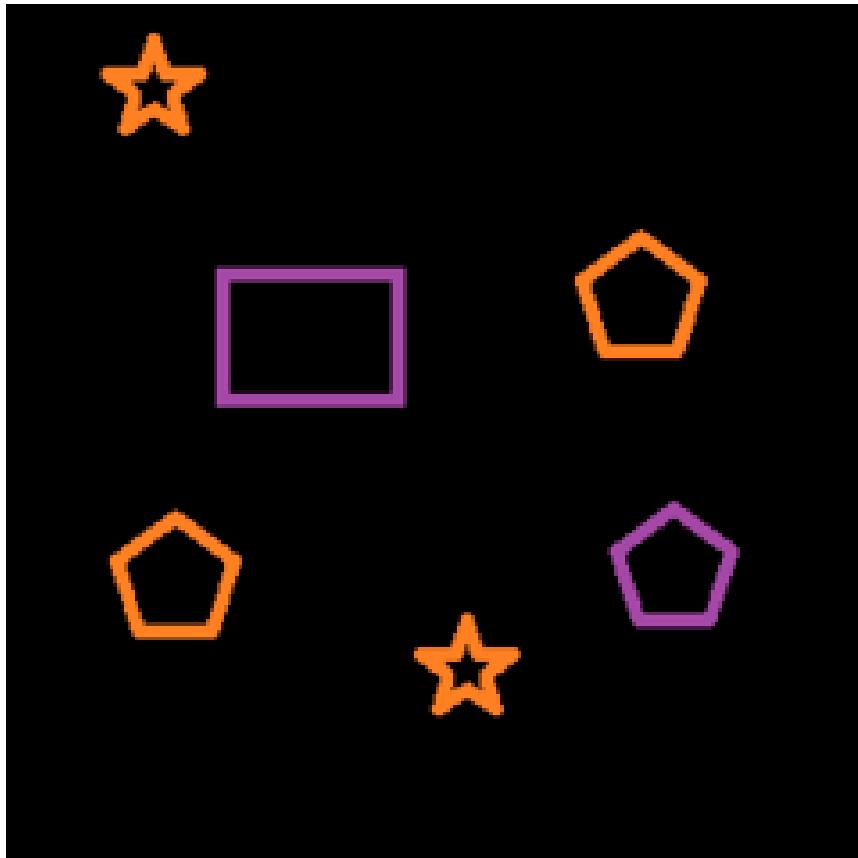


Opgave: Flere gestaltlove på en gang

Gå på nettet og find eksempler hvor flere gestaltlove er overholdt samtidig. Det er nemmest at finde eksempler, hvor elementerne er knapper eller menupunkter.



Opgave: Forskellige figurer



Lav tre forskellige figurer i tre forskellige farver og klip dem ud. Placér dem på forskellige måder på en baggrund, så de illustrerer de fire sidste gestaltlove.

Her ses et eksempel, hvor ingen gestaltlove er overholdt:



Opgave: Andre eksempler på loven om figur og baggrund

Gå på nettet og find et eksempel hvor loven om figur og baggrund ikke er opfyldt.

Modellering



iStockphoto.com/alessiamalatini og Systime

I modelleringsfasen fastlægges den logiske struktur af det it-system, man vil lave.

- Er der tale om en relationel database, fastlægges tabeller (gerne ud fra E/R-overvejelser) og attributter, og tabellerne normaliseres.
- Er der tale om et objekt-orienteret design, fastlægges klasserne, deres klassehierarkier, attributter og indbyrdes afhængigheder.
- Er der tale om et it-system, der f.eks. på et tidspunkt skal sortere elementer, findes der algoritmer, som beskriver, hvordan dette kan gøres.

Ligesom en arkitekt beskriver en planlagt bygning med en samling tekniske og grafiske dokumenter, beskrives et it-system vha. tekster, diagrammer og andet. Det er en kunst at lave en god modellering af et it-system.

Algoritmer



iStockphoto.com/grinvalds

Hvis du skal bage et brød, har du brug for en opskrift. Den skal indeholde konkrete instruktioner som "Vask hænder" og "Mål alle ingredienser af".

Når du skal løse en bestemt type af problem med programmering, beskrives fremgangsmåden med en *algoritme*, som svarer til opskriften på et brød.

Abstraktion

Ved abstraktion ser man bort fra visse aspekter af et emne og fokuserer på ét aspekt. I en algoritme, der er en abstraktion over en programmeringskode, fokuserer man på metoden og ser bort fra sprogspecifikke aspekter som datatyper, erklæringer, syntaks osv. Der skal vælges et passende abstraktionsniveau. Niveauet er passende, når det kan bruges som udgangspunkt for at skrive et program i et vilkårligt programmeringssprog.



Opgave: Største fælles divisor

Største fælles divisor for to tal er det tal, der går op i begge de givne tal, og som er det største, der har denne egenskab. For eksempel er 14 den største fælles divisor af 70 og 252, fordi tallet går op i begge tal (divisor) og er det største af alle fælles divisorer (1, 2, 7, 14).

1. Beregn uden brug af computer største fælles divisor for 35 og 63.
2. Beskriv så præcist som muligt din metode til at beregne største fælles divisor.
Tag eventuelt udgangspunkt i din beregning ovenfor.

Algoritme

En algoritme er en metode med instrukser til at løse et bestemt problem, så et output beregnes ud fra et input.

En algoritme skal altså beregne et output ud fra et input, ligesom en matematisk funktion. Forskellen er, at en algoritme ydermere skal beskrive metoden til at beregne dets output.

Det er altså ikke nok at beskrive, at en algoritme ud fra to tal skal beregne største fælles divisor. Algoritmen skal også med præcise instrukser (i et forståeligt sprog) forklare fremgangsmåden, så et menneske i principippet kan udføre disse instrukser med pen og papir.

Herunder ses et eksempel på Euklids algoritme, som beregner den største fælles divisor af to tal. Euclids algoritme er formodentlig en anden metode til at beregne største fælles divisor end den, du anvendte i opgaven ovenfor.

Euklids algoritme

```

heltal euklid (positivt heltal A, positivt heltal B) {
    så længe (B != 0) {
        hvis (A > B) så A ← A - B;
        ellers B ← B - A;
    }
    returner A;
}
```

Algoritmen ovenfor er skrevet i såkaldt pseudokode, som er en uformel måde at beskrive en algoritmes eller et programs instruktioner.

Den første forekomst af ordet *heltal* indikerer, at algoritmens output er et heltal. Dernæst er det angivet, at de to input A og B også er heltal.

Eksempel: Beregning af største fælles divisor

Dette eksempel viser, hvordan det med Euklids algoritme beregnes, at 14 er største fælles divisor af 70 og 252.

Første kolonne angiver antal gennemløb af løkken, mens A og B angiver værdien af variablerne efter den i'te løkke.

| i | A | B | B!=0 | A>B |
|---|----|-----|-------|-------|
| 0 | 70 | 252 | sandt | falsk |
| 1 | 70 | 182 | sandt | falsk |
| 2 | 70 | 112 | sandt | falsk |
| 3 | 70 | 42 | sandt | sandt |
| 4 | 28 | 42 | sandt | falsk |
| 5 | 28 | 14 | sandt | sandt |
| 6 | 14 | 14 | sandt | falsk |
| 7 | 14 | 0 | falsk | |



Opgave: Brug Euklids algoritme

Beregn største fælles divisor af 114 og 183 ved udregning på papir med Euklids algoritme.



Opgave: Algoritme til at finde personnavn i liste

Beskriv en algoritme, hvis input er en liste af personnavne og et personnavn. Output skal være sandt, hvis listen indeholder personnavnet, og ellers falsk.

Databaser

OBS

Dette er kernestof på B-niveau.

Databasesystemer bruges til at opbevare data, så det er *let* og *hurtigt* at finde, gemme og ændre data i systemet.

Eksempel: Databaser

Der hæves penge, bestilles flybilletter og købes varer over internettet – det er transaktioner, der involverer databaser. Virksomheder lagrer informationer om medarbejdere, kunder og økonomi i databaser. Efterretningstjenester lagrer forbryderprofiler i databaser.



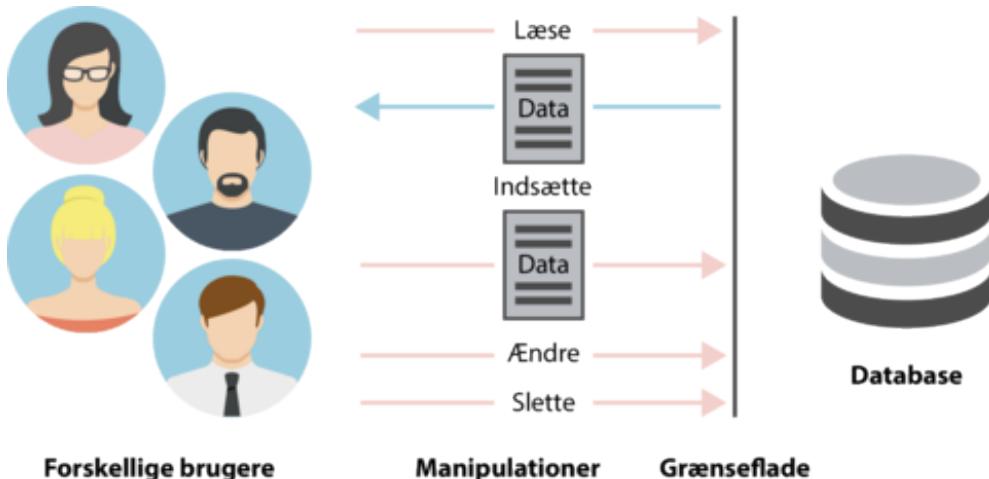
Opgave: Databaser

Find eksempler på it-systemer, som du anvender, hvor der bruges en database.

Databasesystem

Et databasesystem består af to dele. En database, der er en struktureret datamængde, og en grænseflade hvorigennem databasebrugere kan tilgå databasen, dvs. kan:

- læse data i databasen
- indsætte data i databasen
- ændre i data i databasen
- slette data i databasen



iStockphoto.com/artvea/FingerMedium / Systime

Eksempel: Grænseflader, udtræk og manipulation

Når du hæver penge i en automat, er automaten din grænseflade til den database, der indeholder bankens kundekonti. Du kan manipulere med "din" del af databasens indhold ved at hæve penge eller udskrive en kontooversigt. Din hjemlige netbank er en internet-grænseflade, der også kan lave udtræk og manipulere. Bankens ansatte bruger særlige bank-programmer, der kan manipulere enhver kundekonto – og fx hæve gebyrer fra den, ændre dens rentesats eller lukke den.

Et godt databasesystem har en række systemkvaliteter:

- *pladseffektiv lagring*: databasen er billig at lægge diskplads til,
- *fleksibel design*: databasen er let at bruge, vedligeholde og udvide,
- *data er lagret sikert*: ondsindede personer kan ikke uretmæssigt ændre databasens indhold,
- *data er korrekt*: databasens indhold afspejler den virkelighed, databasen modellerer.

En beskrivelse af, hvordan man kan tilgå databaser via en hjemmeside, kan læses under "Andet materiale" i afsnittet [Interaktion med databaser](#).

Analyse

Fra brainstorm til færdigbeskrevet databasesystem

Når man modellerer en database, gennemgår man disse arbejdsprocesser:

- Analyse
- E/R-diagram
- Nøgler
- Tabelskitser
- Normalformer

Analyse

Vi vil tage udgangspunkt i en konkret case og vil modellere databasen, så de tabeller med data, der indgår i databasen, er relateret til hinanden.

Spilleklubben "7 Wonders"

Klubformanden i den lokale spilklub "7 Wonders" har indtil nu forsøgt at have overblik over medlemmer, spil, spilaftener, turneringer, dommere i turneringer og meget andet i et regnark.

| Medlem | Adresse | Har et spil? | Spilnummer | Version | Afgang | Med i turnering? | Turnering | Dato |
|-------------------|---------|--------------|------------|---------|--------|------------------|--------------|------------|
| Matias Hevyej 4 | ja | | 4 | Duel | 2010 | nej | | |
| Matias Sevej 5 | ja | | 5 | Babel | 2015 | | | |
| Marie Baevje 1 | [nej] | | | | ja | | Odense Cup | 13-05-2017 |
| Alma Baevje 1 | ja | | 2 | Babel | 2015 | ja | Roskilde Cup | 19-11-2019 |
| William Skovvej 8 | [nej] | | | | | nej | | |
| Emma Roevej 3 | [nej] | | | | ja | | Odense Cup | 13-05-2017 |

Klubbens regnark

Som det ses, er situationen ved at være helt uoverskuelig, og klubformanden tager kontakt til en systemudvikler. Han ønsker et it-system, der kan opbevare de relevante data, og som kan bruges til at administrere de opgaver, der løbende kommer.

For at analysere problemet og for at systemudvikleren kan blive sat grundigt ind i opgaven med at udvikle en it-løsning, laver klubformanden og systemudvikleren en [brainstorm \(se side 51\)](#) sammen.

De finder alle mulige ord, der på en eller anden måde relaterer sig til klubformandens opgave. Derefter skriver de dem ind i en tabel. Ordene skal ind under den rigtige overskrift, og overskrifterne dækker over følgende:

- En entitet er en enhed, der ofte er en enhed i fysisk forstand. Fx en bil, en person eller et spil.
- En relation er en sammenkobling af to entiteter. Fx medlem ejer spil - her er ejer relationen.
- En attribut er en egenskab ved en entitet.

| Ord | Entitet | Relation | Attribut | Andet |
|--------|---------|----------|----------|---------|
| Medlem | X | | | |
| Bil | | | | Udeladt |

| Ord | Entitet | Relation | Attribut | Andet |
|---------------|---------|----------|----------|-------|
| Navn | | | X | |
| Medlemsnummer | | | X | |
| Turnering | X | | | |
| Ejer | | X | | |
| Spil | X | | | |
| Dommere | | | X | |
| ... | | | | |

Som det ses, er bil udeladt. Senere kan det være en mulighed at udvide it-systemet, så det også kan håndtere, hvem der ejer en bil. Så vil dette kunne indgå, fx når der skal køres til turneringer.

Som det også ses, er Dommere attribut til Turnering. Man kunne også have valgt at lade Dommere være en entitet.



Opgave: Brainstorm til "7 Wonders"

Udfyld ovenstående tabel med flere ord og flere kryds'er.

Sørg for, at der er mindst tre entiteter, og at de hænger sammen via mindst to relationer. Sørg også for, at der er mindst to attributter til hver entitet.

Udover brainstorm med formanden nedskrives også de krav, som formanden har til det færdige it-system. Det kan fx være mulighed for at

- indskrive nye medlemmer
- slå et spil op og se, hvem der ejer det
- slette et medlem, der melder sig ud af klubben

Efter udarbejdelsen af databasen skal den evalueres med henblik på at sikre, at formandens krav er opfyldt.



Opgave: Krav til "7 Wonders"

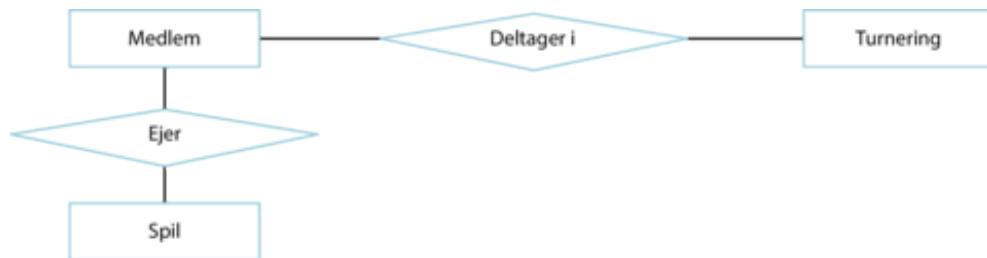
Opskriv mindst fem krav til det færdige it-system ud over de tre, der står ovenfor.

E/R-diagram

I sidste afsnit fik vi analyseret vores database med formålet at finde de entiteter, relationer og attributter, der skal med i vores it-system til klubformanden. Nu vil vi lave et E/R-diagram ud fra entiteterne og relationerne fra vores brainstorm. Det gør vi ved at følge følgende punkter:

1. For hver entitet skrives entitetsnavnet i en kasse
2. For hver relation skrives relationsnavnet i en diamant
3. Den enkelte relation placeres mellem to entiteter, og der tegnes streger mellem dem.

Det vil i vores it-system se sådan ud:



Man vælger navnene på relationerne, så de nogenlunde giver mening, når man læser dem.
Hvis vi ser på relationen "deltager i", læses den fra den ene side:

- medlem **deltager i** turnering

og fra den anden side:

- turnering **har deltagende** medlem

Som det ses, er det ikke sprogligt helt godt begge veje. Alligevel vælger man ikke at have to navne til en relation.



Opgave: Navne på relationer

Opskriv de to sætninger, der hører til relationen Ejer.

Relationsgrad

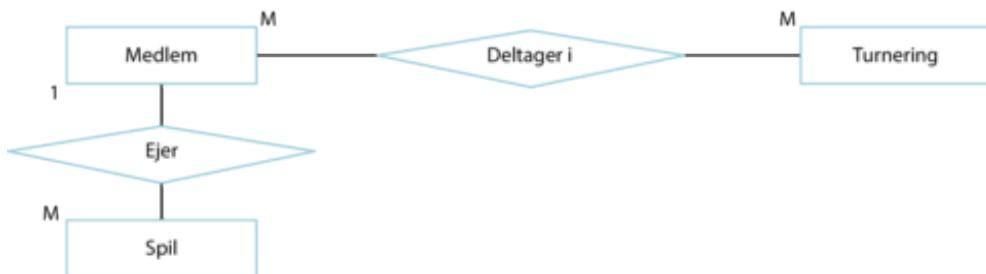
Når vi har lavet vores E/R-diagram, skal relationsgraden bestemmes.

Relationsgrad

Relationsgraden beskriver antalsforholdet mellem relationer. Der er tre typer relationsgrader:

- 1-M (en-til-mange)
- M-M (mange-til-mange)
- 1-1 (en-til-en)

Symbolerne 1 og M placeres på E/R-diagrammet ved entiteterne og ved den linje, der hører til den relevante relation. I vores it-system skal de placeres sådan:



Man læser relationsgraderne som følger. Bemærk, at der er ét symbol pr. sætning, og at man læser symbolet så sent som muligt i sætningen.

1-M-relationen på ovenstående figur læses:

- medlem ejer **mange** spil
- spil ejes af **et** medlem

og M-M-relationen læses:

- medlem deltager i **mange** turnering(er)
- turnering har **mange** medlem(mer)



Opgave: Relationsgraden 1-1

Lav et E/R-diagram med entiteterne patient og journal. Lav relationen har. Her er tale om relationsgraden 1-1.

Lav et nyt eksempel på et E/R-diagram med relationsgraden 1-1.

Er der for hver relation altid kun én mulighed for valg af relationsgrad?

Medlemstype

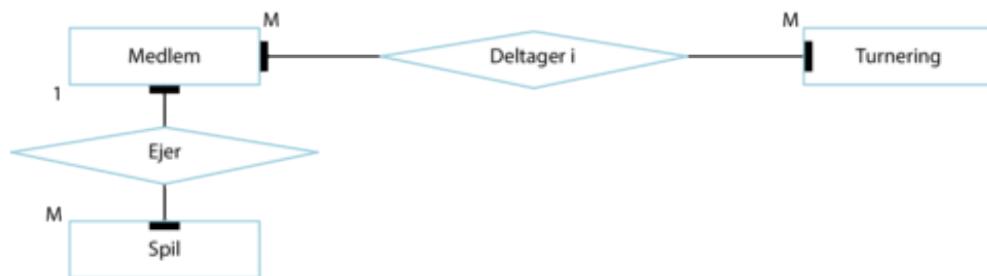
Nu mangler vi at bestemme medlemstypem – så er E/R-diagrammet færdigt.

Medlemstype

Medlemstypen beskriver, om en entitet kan eller skal deltage i en relation.

Symbolet for medlemstypen *skal* er en sort blok placeret inde i en entitet. Symbolet for medlemstypen *kan* er en sort blok placeret uden for entiteten.

I vores it-system kommer det til at se sådan ud:



Man læser medlemstyperne som følger. Bemærk, at man læser én type pr. sætning, og man læser typen så tidligt som muligt i sætningen:

- medlem **kan** eje spil
- spil **skal** ejes af medlem
- medlem **kan** deltage i turnering
- turnering **skal** have medlem



Opgave: Medlemstype

Lav et E/R-diagram med entiteterne dyrlæge og hund. Lav relationen **har besøgt**.

Indtegn medlemstyperne.

Er der her kun én mulighed for valg af medlemstyper?

Nøgler

I analysefasen blev der valgt en række attributter. Vi kan ikke i tabellen over brainstorm se, hvilke attributter der hører til hvilke entiteter. Vi vil starte med at få dette på plads. Vi opskriver entiteternes navne efterfulgt af de tilhørende attributter i en parentes.

I vores it-system kunne det se sådan ud:

Spil(Version, Årgang)

Medlem(Fornavn, Efternavn, Alder)

Turnering(Turneringsdato, Turneringssted, Turneringsnavn, Dommere)

Som det ses, er der særligt for entiteten Spil et problem. Der kan i klubben være adskillige spil med samme version og fra samme årgang. Her er med andre ord et problem med entydighed. Klubformanden har allerede opdaget dette problem, så han har bedt spillenes ejere om, at spillene bliver nummereret. Her klarer vi problemet ved at tilføje attributten Spilnummer til entiteten Spil. Attributten Spilnummer er en såkaldt nøgle, fordi den entydigt definerer det enkelte spil. Her vil vi skrive nøgler med grøn skrift.

Spil(Version, Årgang, **Spilnummer**)

Ved entiteten Medlem er der også et problem. Hverken attributten Fornavn, Efternavn eller Alder sikrer entydighed. Problemet løses ved at tildele et medlemsnummer i form af nøglen Medlemsnummer:

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Entiteten Turnering har også problemet med, at entydighed ikke er sikret. Her løser vi ikke problemet ved at tildele en ekstra attribut. I stedet slår vi to attributter sammen og laver den koblede nøgle Turneringsdato, Turneringssted:

Turnering(**Turneringsdato**, **Turneringssted**, Turneringsnavn, Dommere)

Nøgle

En nøgle er en eller flere attributter, der entydigt definerer et medlem af en entitet.



Opgave: Nøgler



iStockphoto.com/ inka_s

Entiteten Person kan have nøglen CPR-nummer.

Er denne nøgle entydig?

Er der tilfælde, hvor man ændrer en persons CPR-nummer?

Som opgaven ovenfor viser, er det ikke smart at vælge nøgler, der er informationsbærende.

Tabelskitser

Nu vil vi lave tabelskitser. Disse skitser bruges til at lave de endelige tabeller til databasen.

Eksempel: Fra tabelskitse til Tabel

Tabelskitsen til entiteten Elev er

Elev(Navn, Adresse, Fag, Klasse)

og den vil blive til følgende tabel. Bemærk, at der er skrevet data i tabellen i form af eksempler på elever og deres data:

Elev

| Navn | Adresse | Fag | Klasse |
|----------|----------|------------------------------|--------|
| Mathias | Høvej 22 | Matematik, Dansk, Informatik | 3c |
| Sophia | Elvej 11 | Samfunds fag, Dansk, Engelsk | 2b |
| Mohammed | Søvej 2 | Fysik, Informatik, Engelsk | 3x |
| Emma | Høvej 5 | Matematik, Dansk, Engelsk | 2b |

Tabelskitse.

En tabelskitse er et entitetsnavn efterfulgt af de relevante attributter i en parentes.

Når man laver tabelskitser til et helt E/R-diagram, gennemgår man følgende punkter:

- For hver entitet laves en tabelskitse med en nøgle.
- For hver M-M-relation laves en tabelskitse med nøglerne fra de to relationer som nøgle.
- For hver 1-M-relation tages en kopi af nøglen i entiteten ved 1, og denne skrives i entiteten ved M.
Nøglen kaldes da en fremmednøgle i entiteten med M.
- For hver 1-1-relation slås de to tabelskitser sammen til en tabelskitse med alle attributterne.

I vores eksempel med it-systemet til klubben 7 Wonders er punkt 1 færdigt, og foreløbig ser tabelskitserne sådan ud:

Spil(Version, Årgang, **Spilnummer**)

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Turnering(**Turneringsdato**, **Turneringssted**, Turneringens navn, Dommere)

Nu udføres punkt 2, og tabelskitserne ser sådan ud:

Spil(Version, Årgang, **Spilnummer**)

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Turnering(**Turneringsdato**, **Turneringssted**, Turneringsnavn, Dommere)

Deltager I(**Medlemsnummer**,**Turneringsdato**, **TurneringsSted**)

Efter punkt 3 ser tabelskitserne sådan ud:

Spil(Version, Årgang, **Spilnummer**, Medlemsnummer)

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Turnering(**Turneringsdato**, **Turneringssted**, Turneringsnavn, Dommere)

Deltager I(**Medlemsnummer**,**Turneringsdato**, **TurneringsSted**)

Punkt 4 er unødvendigt, da vi ikke har 1-1-relationer.



Opgave: Tabelskitser til spilklubben

Lav tabeller med indhold til de fire tabelskitser til spilklubben.



Opgave: Tabelskitse for 1-1-relation

Se på 1-1-relationen patient har journal.

Find nogle attributter til entiteterne patient og entiteten journal.

Tegn 1-1-relationen som E/R-diagram.

Lav den tilhørende tabelskitse.

Normalformer

Det har vist sig, at der kan opstå en række problemer med en database. For at løse disse problemer vil vi sikre os, at vores tabeller i databasen overholder tre regler. Disse tre regler kaldes *normalformer* (Boyce–Codd normalform). Reglerne bygger ovenpå hinanden, så man starter med at sikre 1. normalform, derefter 2. normalform og til sidst 3. normalform.

1. normalform (1NF)

Alle attributter skal dække over enkle værdier.



Opgave: Hvorfor 1. normalform

Hvorfor er det godt at tabellerne er på 1. normalform?

Hjælp: Hvordan søger man på data i en tabel?

Lad os tjekke om vores tabeller overholder 1. normalform. Tabelskitserne så sådan ud:

Spil(Version, Årgang, **Spilnummer**, Medlemsnummer)

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Turnering(**Turneringsdato**, **Turneringssted**, Turnerningsnavn, Dommere)

Deltager I(**Medlemsnummer**,**Turneringsdato**, **Turneringssted**)

I tabelskitsen for tabellen Spil overholder 1. normalform, da alle fire attributter dækker over enkelte værdier. De ses nemmest ved at lægge data i en mulig tabel:

Spil

| Version | Årgang | Spilnummer | Medlemsnummer |
|---------|--------|------------|---------------|
| Duel | 2007 | 1 | 1004 |
| Leaders | 2015 | 2 | 1662 |
| Babel | 2012 | 3 | 1109 |

På samme måde tjekker vi de tre andre tabeller:

Medlem

| Fornavn | Efternavn | Alder | Medlemsnummer |
|---------|-----------|-------|---------------|
| Emma | Ravn | 19 | 1109 |
| Anton | Hansen | 20 | 1662 |
| Niels | Rasmussen | 19 | 1004 |

Som det ses, er 1. normalform også overholdt for denne tabel.

Nu tjekkes tabelskitsen Turnering:

Turnering

| Turneringsdato | Turneringssted | Turnerningsnavn | Dommere |
|----------------|----------------|-----------------|----------------------------|
| 01 - 08 - 2017 | Lystrup | Lystrup Cup | Brixen, Frandsen, Sønderby |

| Turneringsdato | Turneringssted | Turneringsnavn | Dommere |
|----------------|----------------|----------------|------------------|
| 22 - 04 - 2017 | Odense | Odense Cup | Frandsen, Hansen |

Som det ses, er der et problem med attributten Dommere, og dermed er tabelskitsen ikke på 1. normalform. Problemet løses ved at opsplitte tabelskitsen i to tabelskitser. Den ene indeholder de attributter, der ikke er en del af problemet. Den ser sådan ud:

Turnering(Turneringsdato, Turneringssted,Turneringsnavn)

Den anden indeholder attributten med problemet og nøglen:

Dommer(Turneringsdato, Turneringssted, Turnersdommer)

Med eksempler på data i de tilhørende tabeller kunne det se sådan ud:

Turnering

| Turneringsdato | Turneringssted | Turneringsnavn |
|----------------|----------------|----------------|
| 01 - 08 - 2017 | Lystrup | Lystrup Cup |
| 22 - 04 - 2017 | Odense | Odense Cup |

og

Dommer

| Turneringsdato | Turneringssted | Turnersdommer |
|----------------|----------------|---------------|
| 01 - 08 - 2017 | Lystrup | Brixen |
| 01 - 08 - 2017 | Lystrup | Frandsen |
| 01 - 08 - 2017 | Lystrup | Sønderby |
| 22 - 04 - 2017 | Odense | Frandsen |
| 22 - 04 - 2017 | Odense | Hansen |

Nu tjekkes den sidste tabelskitse DeltagerI:

DeltagerI:

| Medlemsnummer | Turneringsdato | Turneringssted |
|---------------|----------------|----------------|
| 1004 | 01 - 08 - 2017 | Lystrup |
| 1662 | 01 - 08 - 2017 | Lystrup |
| 1004 | 22 - 04 - 2017 | Odense |

| Medlemsnummer | Turneringsdato | Turneringssted |
|---------------|----------------|----------------|
| 1004 | 22 - 04 - 2017 | Odense |

Her er der ingen problemer med 1. normalform.

Alt i alt er alle tabelskitser på 1. normalform, og de ser nu sådan ud:

Spil(Version, Årgang, **Spilnummer**, Medlemsnummer)
 Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)
 Turnerer(**Turneringsdato**, **Turneringssted**, TurneringsNavn)
 Dommer(**Turneringsdato**, **Turneringssted**, Turneringsdommer)
 Deltager I(**Medlemsnummer**, **Turneringsdato**, **Turneringssted**)



Opgave: 1. normalform

Få følgende tabelskitse på 1. normalform:

| Ejer | Hund |
|-------------------------|--------|
| Anton, Emma, Jens | Diesel |
| Anton, Cecilie, Mathias | Benn |

2. normalform (2NF)

Tabelskitsen skal være på 1. normalform og

hvis der er en attribut, der er afhængig af nøglen, så skal den være afhængig af hele nøglen.

Her kan opstå et problem med 2. normalform, hvis der i tabelskitsen indgår en koblet nøgle.

Vi tjekker de fem tabelskitser. Det ses, at der ingen problemer er med tabellerne Spil og Medlem, for her indgår ikke koblede nøgler:

Spil(Version, Årgang, **SpilNummer**, MedlemsNummer)
 Medlem(Efteravn, Alder, **Medlemsnummer**)

Nu ser vi på tabelskitsen DeltagerI. Her er der ingen problemer, da alle attributterne indgår i nøglen:

DeltagerI(**Medlemsnummer**, **Turneringsdato**, **Turneringssted**)

Så ser vi på tabelkitsen Dommer. Her ses det, at attributten Turneringsdommer ikke er afhængig af nogen af attributterne i den koblede nøgle:

Dommer(**TurneringsData**, **Turneringssted**, TurneringsDommer)

Til sidst ser vi på tabelkitsen Turnering:

Turnering(**TurneringsData**, **Turneringssted**, TurneringsNavn)

Klubformanden har fortalt, at alle turneringsnavne skal bestå af bynavnet og ordet 'Cup'. Eksempler er Odense Cup, Thisted Cup, Glostrup Cup etc.

Her opstår et problem med 2. normalform. Attributten Turneringsnavn er afhængig af stedet – men ikke af dato'en.

Problemet løses ved at opdele tabelkitsen i to. Den ene indeholder den afhængige attribut og den del af nøglen, som den er afhængig af:

Turneringssted(**Turneringssted**, TurneringsNavn)

Den anden indeholder alle attributter bortset fra den, der var delvis afhængig:

Turnering(**TurneringsData**, **Turneringssted**)

Med eksempler på data i de tilhørende tabeller kunne det se sådan ud:

Turneringssted

| Turneringssted | Turneringsnavn |
|----------------|----------------|
| Lystrup | Lystrup Cup |
| Odense | Odense Cup |

og

Turnering

| Turneringsdato | Turneringssted |
|----------------|----------------|
| 01 - 08 - 2017 | Lystrup |
| 22 - 04 - 2017 | Odense |

Alt i alt ser tabellerne på 2. normalform sådan ud:

Spil(Version, Årgang, **SpilNummer**, MedlemsNummer)

Medlem(Fornavn, Efternavn, Alder, **Medlemsnummer**)

Turneringssted(**TurneringsSted**, TurneringsNavn)

Turnering(**TurneringsData**, **TurneringsSted**)

Dommer(**Turneringsdato**, **Turneringssted**, Turneringsdommer)
Deltager I(**Medlemsnummer**,**Turneringsdato**, **Turneringssted**)



Opgave: Afhængighed

Hvem er afhængig af hvem?

| Telefonnummer | Navn |
|---------------|------|
| | |
| | |

| Navn | Adresse |
|------|---------|
| | |
| | |

| Bil | Nummerplade |
|-----|-------------|
| | |
| | |

| Bil | Værksted |
|-----|----------|
| | |
| | |

Find selv på 3 til:



Opgave: Hvorfor 2. normalform

Hvorfor er det godt at tabellerne er på 2. normalform?



Opgave: 2. normalform

Giv SP et nyt fag.
Giv EH et nyt telefonnummer.

| Lærer B | Fag B | Telefonnummer | Klasse |
|---------|-----------|---------------|----------|
| EH | Matematik | 56347865 | 2.x |
| EH | Datalogi | 56347865 | 2sm3smDI |
| EH | Matematik | 56347865 | 2sMA |
| SP | Tysk | 23876311 | 1.y |
| SP | Tysk | 23876311 | 3.a |

Få tabellen på 2. normalform.

Giv derefter igen SP et nyt fag og EH et nyt telefonnummer.

3. normalform (3NF)

Tabelskitsen skal være på 2. normalform og

ingen attributter må være indirekte afhængige af nøglen.

Vi tjekker de seks tabelskitser og ser, at der ingen problemer er.

Vi vil se på et eksempel, hvor problemet opstår:

Eksempel: 3. normalform

Vi ser på følgende tabel over måner. Tabellen indeholder også den planet, som den enkelte måne tilhører, og planetens afstand til Solen:

Måne

| Måne | Planet | Afstand planet/Solen |
|--------|---------|----------------------|
| Io | Jupiter | 671000 |
| Europa | Jupiter | 671000 |
| Dione | Saturn | 377400 |

Hvis der opdages en ny måne til en planet, skal der ved indskrivning både skrives planet OG dennes afstand Planet/sol – med fare for fejlindskrivning. Måne er nøgle. Planet er afhængig af Måne. Afstanden er afhængig af Planet. Det vil sige, at Afstand Planet/solen er indirekte afhængig af Måne.

Problemet løses ved at splitte tabelskitsen op i to. I den første er nøglen og den afhængige attribut:

| Måne | Planet |
|--------|---------|
| Io | Jupiter |
| Europa | Jupiter |
| Dione | Saturn |

Her er Måne igen nøgle.

| Planet | Afstand til planet/Solen |
|---------|--------------------------|
| Jupiter | 671000 |
| Saturn | 377400 |

Her er Planet nøgle.

Nu er databasen på tredje normalform.



Opgave: Hvorfor 3. normalform

Hvorfor er det godt, at tabellerne er på 3. normalform?



Opgave: 3. normalform



Statue af Frederik 5.

iStockphoto.com/harryfn

Indskriv skulpturen Knælende Dreng, der står i Holstebro.

| SkulpturNummer | SkulpturNavn | Postnummer | By |
|----------------|--------------------------|------------|-----------|
| 101 | Kvinde på kærre | 7500 | Holstebro |
| 332 | Krukken | 8900 | Randers |
| 511 | Statue af Frederik d. 5. | 1257 | København |

Få tabellen på 3. normalform.

Indsæt nu skulpturen Hjertefugl, der står i Randers.

Nu er tabelskitserne til klubben klar til at blive realiseret som tabeller i en database.

Programmering



iStockphoto.com/mrPliskin

Programmeringssprog kan på mange måder sammenlignes med naturlige sprog, som vi taler og skriver i det daglige. Disse varierer både i anvendte tegn og ord samt konstruktion af sætninger.

Dette afsnit indeholder en række eksempler skrevet i fire udvalgte programmeringssprog. Herunder kan du se en oversigt over de fire sprog.

| Sprog | Form | Anvendes til... | Anvendes af... |
|------------|--------------|---------------------------------------|----------------------------------|
| Scratch | Visuelt | Spil og animationer | Elever og studerende |
| JavaScript | Tekstbaseret | Kommunikation med brugeren i browsere | De fleste websiders programmører |

| Sprog | Form | Anvendes til... | Anvendes af... |
|------------|--------------|---------------------------------------|------------------------------------|
| PHP | Tekstbaseret | Serverprogrammer til webapplikationer | Blandt andet Facebook og Wikipedia |
| Processing | Tekstbaseret | Digital kunst og grafik | Elever og studerende |

Vejledning til udviklingsmiljøer

For at kunne lave programmer i et af de fire programmeringssprog, er det nødvendigt at få adgang til et udviklingsmiljø. Herunder er en oversigt over yderligere information for de fire sprog.

| Sprog | Fortolker | Editor | Dokumentation |
|------------|-----------------------------------|------------------------------|--------------------------------------|
| Scratch | Scratch webside | Se fortolker. | Scratch blokke |
| JavaScript | En browser | F. eks. Atom | JavaScript reference |
| PHP | Indeholdt i XAMPP | F.eks. Atom | PHP reference |
| Processing | Processing | Se fortolker. | Processing reference |

Bogens eksempler i filer

Undervejs vil du støde på eksempler på programmer, som du skal køre og/eller udvide. Herunder er der for hvert sprog links til zip-komprimerede mapper, som indeholder filerne.

| Sprog | Vejledning |
|--|--|
| Scratch (<i>Filen kan downloades fra ibogen se https://informatik.systime.dk/index.php?id=1075&L=0</i>) | Importeres i Scratch: Fil > Upload fra din computer. |
| JavaScript (<i>Filen kan downloades fra ibogen se https://informatik.systime.dk/index.php?id=1075&L=0</i>) | HTML-filer med JavaScript: Dobbeltklik på filen. |
| PHP (<i>Filen kan downloades fra ibogen se https://informatik.systime.dk/index.php?id=1075&L=0</i>) | Start webserveren (XAMPP) og åbn linket i browseren (localhost/...). |
| Processing (<i>Filen kan downloades fra ibogen se https://informatik.systime.dk/index.php?id=1075&L=0</i>) | Dobbeltklik på filen for at åbne i Processing, og tryk på Run. |



Opgave: Programmer og udviklingsmiljøer

1. Undersøg programmer lavet med et givet programmeringssprog ved at søge på nettet efter "programs made with <sprog>". Eksempelvis "programs made with processing". Hvad bruges sproget til?
2. Hent de nødvendige programmer eller opret dig som bruger for dit programmeringssprogs udviklingsmiljø via ovenstående tabel.

Nu er du klar til at lave programmer i dit programmeringssprog!

Syntaks og semantik

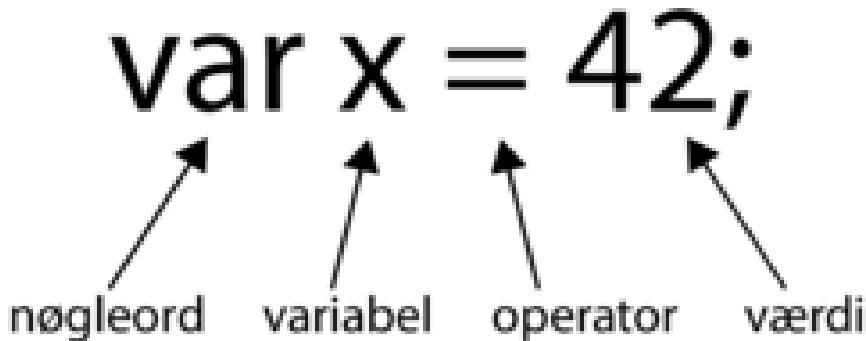
Naturligt sprog består af sætninger. Et eksempel er sætningen "Jeg vil tage sko på nu".

Et program skrevet i et programmeringssprog består også af sætninger.

Sætning

En sætning er et programmeringssprogs grundelement, der udtrykker en handling.

Eksempel: Sætninger



I Scratch genkendes sætninger som blokke, der kan stables oven på hinanden.

I JavaScript, PHP og Processing er sætninger adskilt på forskellig vis. Som udgangspunkt er de adskilt med semikolon.

```
var x = 42; var y = 10; var z = x + y;
```

Andre programmeringssprog adskiller sætninger med linjeskift, så én linje svarer til én sætning.

En sætning kan blandt andet være opbygget af variabelnavne, værdier, operatorer, udtryk, nøgleord og kommentarer. Herover består første sætning af et nøgleord, et variabelnavn, en operator og en værdi.

Et særligt element, der kan indgå i sætninger er udtryk:

Udtryk

Et udtryk er et programmeringssprogs grundelement, der kan evalueres til en værdi.

Eksempel: Udtryk

I Scratch genkendes udtryk som blokke, der ikke kan stables. De har enten afrundede eller kantede ender.

I tekstbaserede sprog som JavaScript, PHP og Processing genkendes udtryk som en kombination af værdier, variable, operatorer og funktioner, som i sidste ende evalueres til en værdi.

Et eksempel er:

$\text{sqrt}(x)+10$



Colourbox.com

Sætninger i naturlige sprog kan være mere eller mindre korrekte. Se på følgende to sætninger:

- "Hygjik alpif gutre"
- "Vandrer hest det os dem"

Den første sætning består slet ikke af ord fra et naturligt sprog, og den anden sætning består ganske vist af ord fra det danske sprog, men sætningen giver ikke logisk mening.

Syntaks

Syntaks beskriver den grammatisk korrekte form på en sætning i sproget.

Ingen af de to ovenstående sætninger er syntaktisk korrekte, hvis vi tager udgangspunkt i det danske sprog.

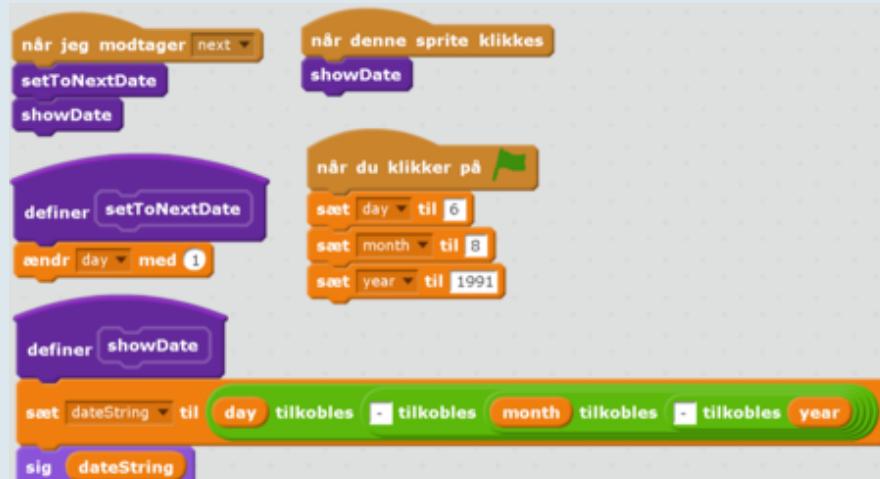
Vi vil gerne lave syntaktisk korrekte sætninger i vores programmer. Det kan de forskellige editorer, som vi skriver koden i, hjælpe med ved f.eks. at bruge farver.

Eksmpel: Farvet programmeringskode

Koden herunder indeholder første implementering af et program, der finder en dato. Koden her og i din editor er formatteret med forskellige farver for at gøre den mere overskuelig for programmøren.

Kode: Funktioner i Dato version 1

Scratch (metoder 1.sb2)



JavaScript (metoder1.html)

```
// Variable til repræsentation af dato
var day;
var month;
var year;

// Funktion til tælle datoen en dag op
function setToNextDate() {
    day = day + 1;
}

// Funktion til at vise datoen
function showDate() {
    alert(day + "-" + month + "-" + year);
}

function test(day, month, year) {
    alert("Dagen efter ...");
    this.day = day;
    this.month = month;
    this.year = year;
    showDate();
    setToNextDate();
    alert("... er ...");
    showDate();
}

// Test af programmet med udvalgte datoer
test(14, 5, 1979);
test(30, 4, 1979);
test(30, 5, 1979);
test(28, 2, 1979);
test(28, 2, 1980);
test(28, 2, 1900);
test(28, 2, 2000);
test(31, 12, 2016);
```

PHP (metoder1.php)

```
<?php
// Variable til repræsentation af dato
\$day;
\$month;
\$year;

/* Ændrer datoen til næste dag */
function setToNextDate() {
    global \$day;
    \$day = \$day + 1;
}

/* Viser datoen i formatet dd-mm-åååå */
function showDate() {
    global \$day, \$month, \$year;
    echo \$day . "-" . \$month . "-" . \$year;
}

function test(\$d, \$m, \$y) {
    echo "<p>Datoen efter ";
    global \$day, \$month, \$year;
    \$day = \$d; \$month = \$m; \$year = \$y;
    showDate();
    setToNextDate();
    echo " er ";
    echo showDate();
    echo "</p>";
}

// Test af programmet med udvalgte datoer
test(14, 5, 1979);
test(30, 4, 1979);
test(30, 5, 1979);
test(28, 2, 1979);
test(28, 2, 1980);
test(28, 2, 1900);
test(28, 2, 2000);
test(31, 12, 2016);
?>
```

Processing (metoder_1.pde)

```
// Standard funktion til opsætning
// Attributter til repræsentation af en dato
int day; // Dag
int month; // Måned
int year; // År
int ypos;

void setup() {
    size(480, 480);
    ypos = 10;
    PFont f;
    f = createFont("Arial",16,true);
    textAlign(f,36);
    fill(255);
    test(14, 5, 1979);
    test(30, 4, 1979);
    test(30, 5, 1979);
    test(28, 2, 1979);
    test(28, 2, 1980);
    test(28, 2, 1900);
    test(28, 2, 2000);
    test(31, 12, 2016);
}

// Standard funktion til tegning i vinduet
void draw() {
}

// Sæt datoen til næste dag
void setToNextDate() {
    // Optælling af dag
    day = day + 1;
}

// Returner en streng i formatet dd-mm-åååå
void showDate(int xpos, int ypos) {
    text(day + "-" + month + "-" + year,xpos,ypos);
}

void test(int d, int m, int y) {
```

```
day = d; month = m; year = y;  
ypos = ypos + 50;  
showDate(10, ypos);  
setToNextDate();  
showDate(220, ypos);  
}
```

Ord farvet med turkis skrift i JavaScript, PHP og Processing kaldes for nøgleord og har en særlig betydning i sproget. De kan ofte ikke bruges som variabelnavne. I Scratch udgør blokkene nøgleord, for eksempel "Når denne sprite klikkes" og er dermed afhængig af det valgte sprog (dansk eller engelsk).

Sætningen "Højden på farven har en anden lyd" er syntaktisk korrekt, men giver ingen logisk mening. Vi vil gerne have at de sætninger vi skriver i vores programmer giver mening, men her kan editorerne ikke hjælpe. Hvis vi skriver programsætninger, der dividerer med 0 eller på anden måde ikke giver mening, kan vi først ved test opdage denne type fejl.

Semantik

Semantikken for et sprog beskriver meningen i en sætning i sproget.

Sætningen "Højden på farven har en anden lyd" er altså ikke semantisk korrekt.

Eksempel: Syntaktisk ugyldige programdele

Følgende programdele fra JavaScript er syntaktisk ugyldige.

Denne sætning er syntaktisk ugyldig på ord-niveau:

```
var 5e = 42;
```

Efter nøgleordet var skal der være navnet på en variabel. Da variabelnavne i JavaScript ikke kan starte med et ciffer, er det en syntaksfejl i sammensætning af tegn til ord.

Denne sætning er syntaktisk ugyldig på sætning-niveau:

```
function var foo() {}
```

Orde function og var er syntaktisk korrekte, men function må ikke efterfølges af var.

Denne sætning er syntaktisk ugyldig på kontekst-niveau:

```
a.substring(1)
```

Hvis variablen a ikke er erklæret som en streng, kan metoden substring ikke kaldes, hvilket giver en syntaksfejl på kontekst-niveau.



Opgave: Syntaks i Scratch

Overvej hvorfor det ikke er nødvendigt at bekymre sig om syntaks i Scratch.



Opgave: Niveauer af syntaks

1. Indtast programdelene fra eksemplet herover i JavaScript eller tilsvarende programdele for et andet programmeringssprog.
2. Hvilken fejlbeskæftigelse medfører det?



Øvelse: Tegn i koden

Diskutér med udgangspunkt i et bestemt programmeringssprog følgende.

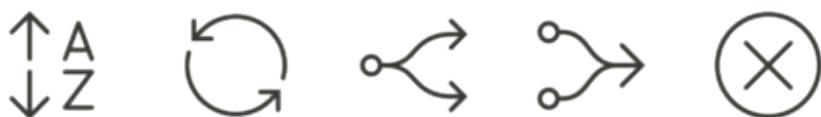
1. I hvilke situationer kan et punktum optræde i koden?
2. I hvilke situationer kan en parentes optræde i koden?

Syntaks afhænger af programmeringssproget. Herunder kan du finde syntaksen for hvert sprog.

| Kilde | Beskrivelse |
|---|-----------------------------------|
| JavaScript syntax | JavaScripts (ECMAScripts) syntaks |
| PHP.net: Basic syntax | PHP syntaks |
| processing.org Language Reference | Forenklet Java-syntaks |

Kontrolstrukturer og funktioner

Et vigtigt område af programmering er styring af programmets kørsel. Dvs. i hvilken rækkefølge sætningerne i programmet afvikles og på hvilket tidspunkt.



iStockphoto.com/lushik

Kørslen af et program afhænger af de regler, som et sprog er bygget op om. For at du kan bestemme, hvordan et program skal fungere, skal du bruge kontrolstrukturer til at styre kørslen.

Kontrolstruktur

En kontrolstruktur er en sætning i et programmeringssprog, der styrer den rækkefølge, et programs sætninger udføres.

Der eksisterer tre kontrolstrukturer:

- Sekvens
- Forgrening
- Løkke (kaldes også iteration)



Opgave: Kontrolstrukturer

Et eksempel på noget man gør i en sekvens er
stå op – børst tænder – tag tøj på.

Giv et eksempel på noget du gør i en forgrenning og noget du gør i en løkke.

Sekvenser

Vi vil se på kontrolstrukturen sekvens.

Eksempel: Køb af sko

Du ønsker at købe sko på en amerikansk webshop, som tilbyder fri fragt. Da prisen er angivet i dollars (USD), vil du gerne beregne, hvad skoene koster i danske kroner (DKK).

Hvis prisen i USD er 200, og kurserne (prisen i DKK for 1 USD) er 7.01, beregnes prisen i DKK ved at gange beløbet i DKK med kurserne:

$$200 \cdot 7.01 = 1402$$



Opgave: Anvende Valuta med sekvens

1. Kør programmet hørende til filen, som du finder i koden herunder.
2. Læs koden hørende til programmet.
3. Sammenlign linjerne i koden med programmets udførelse.

Kode: Valuta med sekvens

Scratch (Sko 1 sekvens.sb2)



JavaScript (sko1sekvens.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Valuta 1: Sekvens</title>
</head>

<body>
<p>Prisen i DKK er: <span id="prcDkk"></span></p>
<script type="text/javascript">
var prcUsd = prompt('Hvad koster dine sko i USD?');
var rateUsd =
    prompt('Hvad koster 1 USD i DKK (f.eks. 7.01)?');
document.getElementById('prcDkk').innerHTML =
    prcUsd * rateUsd;
</script>
</body>

</html>
```

PHP (sko1sekvens.php)

```
<?php
/* Eksempel på kald til programmet:
valuta1sekvens.php?prcUsd=200&rateUsd=7.01 */
\$prcDkk = \$_GET['prcUsd'] * \$_GET['rateUsd'];
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Valuta 1: Sekvens</title>
</head>

<body>
<p>Prisen i DKK er: <?= \$prcDkk ?></p>
</body>

</html>
```

Processing (sko1sekvens.pde)

```
int prcUsd;
float rateUsd;

void setup() {
    size(800,450);
    textSize(32); fill(255);
}

void draw() {
    background(0);
    rateUsd = 3 + float(mouseX)/100.0;
    prcUsd = mouseY;
    text("Hvad koster dine sko i USD (lodret): "
        + prcUsd, 50, 100);
    text("Hvad koster 1 USD i DKK (vandret): "
        + nf(rateUsd, 1, 2), 50, 200);
}

void mousePressed() {
    noLoop();
    text("Prisen i DKK er: " +
        nfc(prcUsd * rateUsd, 2), 50, 300);
}
```

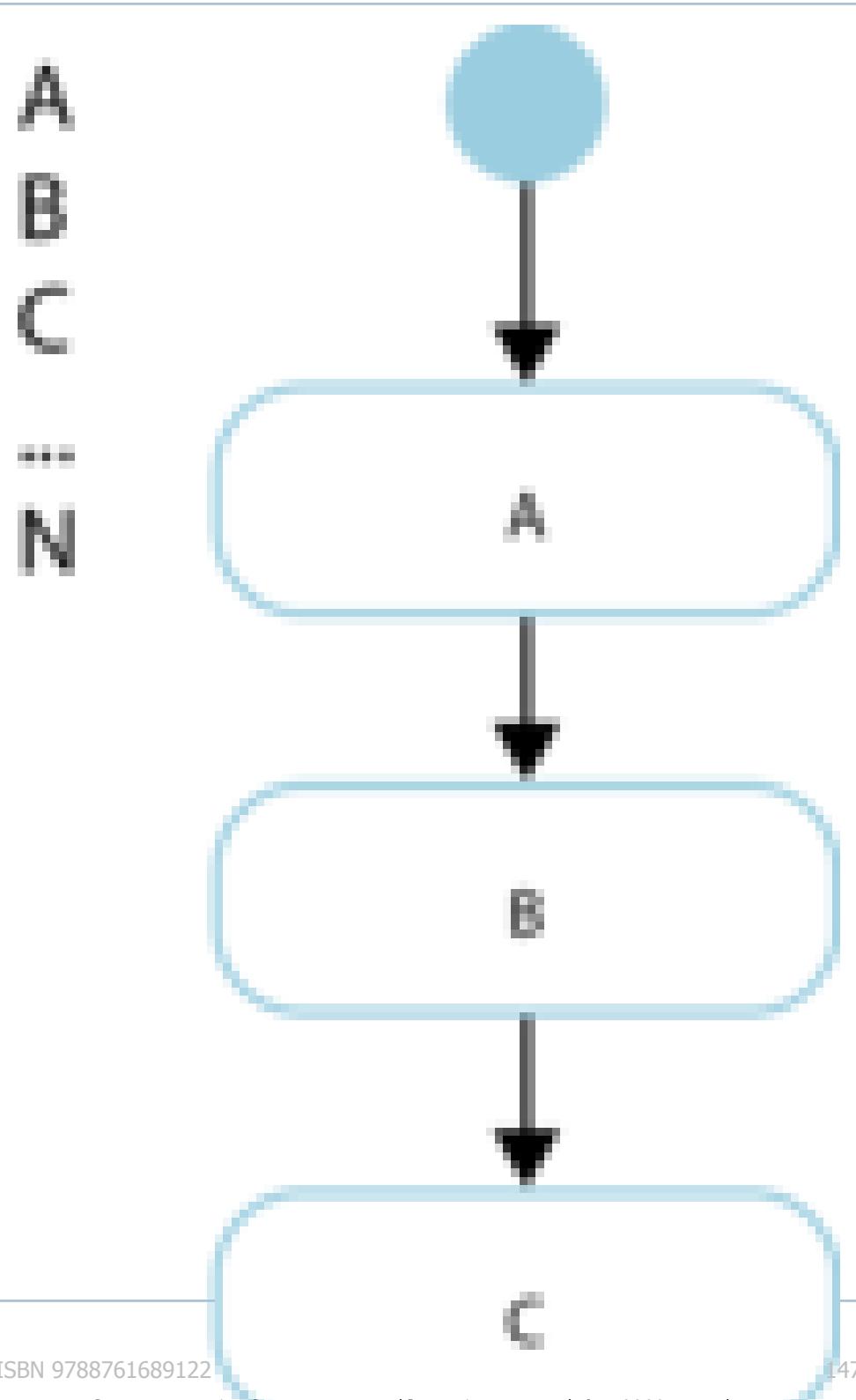
Sekvens

En sekvens er en simpel kontrolstruktur til at udføre en liste af sætninger i rækkefølge.

Programmet bruger udelukkende kontrolstrukturen sekvens til at omsætte dit input med pris og kurs til en pris i danske kroner. Hver sætning bliver kørt en efter en i rækkefølge.

Dette er den mest simple af de tre kontrolstrukturer. Herunder er strukturen for en sekvens beskrevet visuelt med et rutediagram.

Eksempel: Sekvens



Sekvens



Opgave: Fra DKK til USD

Hvis et beløb i DKK skal omregnes til USD, skal beløbet i stedet divideres med kursen.

1. Ret i programmet ovenfor, så brugeren skal indtaste et beløb i DKK i stedet for et beløb i USD.
2. Ret i programmet, så prisen omregnes fra DKK til USD og vises for brugeren.

Forgreninger

Det er sjældent nok at bruge en enkelt sekvens i et program. Vi vil gerne have en sætningsstruktur, der tillader et programs kørsel at bevæge sig i forskellige retninger afhængig af en given betingelse.

Eksempel: Køb af sko med afgift

Når varer indføres i Danmark, pålægges de ofte afgifter, som afhænger af varens pris. Dette skal nu indgå i beregningen af prisen i danske kroner.

Hvis prisen overstiger 1.150 kr., pålægges told, som for visse sko er 17%. Hvis prisen overstiger 80 kr., pålægges moms på 25% og importgebyr på 160 kr.

Priseksempler:

| Beløb | Afgifter | Beregning |
|----------|---------------------------|---|
| 50 kr | Ingen | 50 kr |
| 200 kr | Moms og importgebyr | $200 \text{ kr} \cdot 1,25 + 160 \text{ kr} = 410 \text{ kr}$ |
| 1.500 kr | Told, moms og importgebyr | $1.500 \text{ kr} \cdot 1,17 \cdot 1,25 + 160 = 2353,75 \text{ kr}$ |

Kilde: SKAT (2017), Internethandel uden for EU, tilgængelig på <http://www.skat.dk/SKAT.aspx?oid=2236525&vid=0> [läst 12. april 2017]



Opgave: Anvende programmet Sko med afgifter

1. Brug programmet (se filen herunder) til at lave beregninger for tre beløb. Husk at bruge punktum ved decimaltal:
 1. Beløb uden afgifter
 2. Beløb med moms og importgebyr
 3. Beløb med told, moms og importgebyr
2. Overvej hvordan du ville skrive koden til programmet.
3. Se videoen herunder, der forklarer processen med udvikling af koden.

Bemærk at videoen forklarer udvikling af koden i PHP, men processen er den samme for de andre programmeringsprog.

Kode: Sko med afgifter**Scratch (Sko 2 forgrening.sb2)**

JavaScript (sko2forgrening.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Valuta 2: Forgrening</title>
</head>
<body>
<p>Prisen i DKK er: <span id="prcDkk"></span></p>
<script type="text/javascript">
var prcUsd = prompt('Hvad koster dine sko i USD?');
var rateUsd =
    prompt('Hvad koster 1 USD i DKK (f.eks. 7.01)?');
var prcDkk = prcUsd * rateUsd;
if (prcDkk > 80) { // Varen fortoldes
    if (prcDkk > 1150) {
        prcDkk = prcDkk * 1.17; // Told
    }
    prcDkk = prcDkk * 1.25; // Moms
    prcDkk = prcDkk + 160; // Importgebyr
}
document.getElementById('prcDkk').innerHTML =
    prcDkk;
</script>
</body>
</html>
```

PHP (sko2forgrening.php)

```
<?php
/* Eksempel på kald til programmet:
   valuta2forgrening.php?prcUsd=200&rateUsd=7.01 */
$pvcDkk = \$_GET['prcUsd'] * \$_GET['rateUsd'];
if (\$pvcDkk > 80) { // Varen fortoldes
    if (\$pvcDkk > 1150) {
        \$pvcDkk = \$pvcDkk * 1.17; // Told
    }
    \$pvcDkk = \$pvcDkk * 1.25; // Moms
    \$pvcDkk = \$pvcDkk + 160; // Importgebyr
}
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Valuta 2: Forgrenings</title>
</head>
<body>
<p>Prisen i DKK er: <?= \$pvcDkk ?></p>
</body>
</html>
```

Processing (sko2forgrening.pde)

```
int prcUsd;
float rateUsd, prcDkk;

void setup() {
    size(800,450);
    textSize(32); fill(255);
}

void draw() {
    background(0);
    rateUsd = 3 + float(mouseX)/100.0;
    prcUsd = mouseY;
    text("Hvad koster dine sko i USD (lodret): "
        + prcUsd, 50, 100);
    text("Hvad koster 1 USD i DKK (vandret): "
        + rateUsd, 50, 200);
}

void mousePressed() {
    noLoop();
    prcDkk = prcUsd * rateUsd;
    if (prcDkk > 80) { // Varen fortoldes
        if (prcDkk > 1150) {
            prcDkk = prcDkk * 1.17; // Told
        }
        prcDkk = prcDkk * 1.25; // Moms
        prcDkk = prcDkk + 160; // Importgebyr
    }
    text("Prisen i DKK er: " + prcDkk,
        50, 300);
}
```

Undgå data i koden

I programmet ovenfor er beløbsgrænser og afgifter indeholdt i koden. Det skal så vidt muligt undgås i udvikling af it-systemer, da data bør ligge i databaser, filer eller andre steder, hvor det hører til.

Ved at lægge data andre steder, er det muligt at ændre disse som bruger af it-systemet, så programmernes kode ikke behøver ændres.

For enkelthedens skyld, er data i denne bogs eksempler indeholdt i koden.

Forgrening

En forgrening er en kontrolstruktur som udfører en række sætninger, hvis en givet betingelse er sand.

Betingelse

En betingelse er et udtryk som ved evaluering bliver sandt eller falsk. En betingelse anvendes i en forgrening eller en løkke.

Eksempel: Betingelse

Som det fremgår af definitionen, indgår en betingelse i forgreninger. I den første forgrening herover er betingelsen

$$prcDkk > 80$$

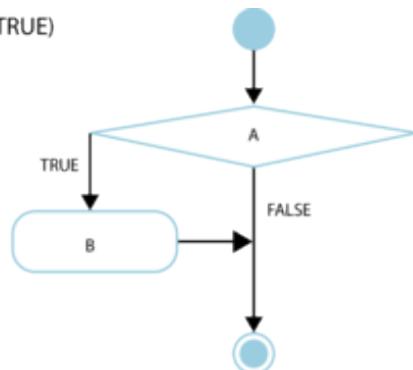
Hvis denne betingelse er sand, udføres den efterfølgende blok af kode. Hvis betingelsen er falsk, springes blokken af kode over.

To typer forgreninger

IF (A = TRUE)

Then B

End IF



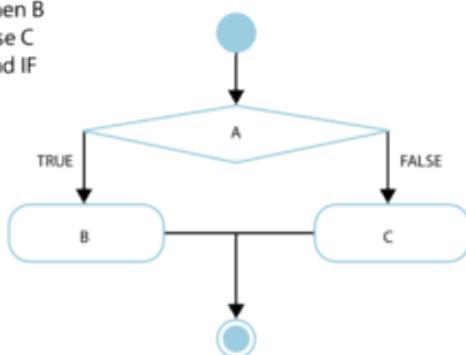
Én gren: Hvis betingelsen A er sand, udføres blok B.

IF (A = True)

Then B

Else C

End IF



To grene: Hvis betingelsen A er sand, udføres blok B. Ellers udføres blok C.

I eksemplet med sko er der kun én gren i forgreningen, som bliver udført, hvis betingelsen er sand. Der kan dog også være to grene, hvor den anden gren bliver udført, hvis betingelsen er falsk.

De to diagrammer herover visualiserer forløbet ved de to typer betingelser.

Én gren: Hvis betingelsen A er sand, udføres blok B.

To grene: Hvis betingelsen A er sand, udføres blok B. Ellers udføres blok C.

Forgrening i Scratch



I koden ovenfor kan du se blokken i Scratch til en forgrening med en gren.

I Scratch anvendes blokken til højre til en forgrening med to grene. Den indsatte kode i det første indhak bliver udført, hvis betingelsen øverst er sand, og den indsatte kode i det andet indhak bliver udført, hvis betingelsen er falsk.

Forgrening i JavaScript, PHP og Processing

Sprogene JavaScript, PHP og Processing er alle indirekte påvirket af sproget C, og de deler derfor på flere områder syntaks. For disse sprog er syntaksen for en forgrening med to grene:

```
if (A) {
```

```
    B;
```

```
} else {
```

```
    C;
```

{}



Opgave: Udvidelse af sko med afgifter

1. Indsæt en forgrening med to grene efter indtastning af beløbet.
2. Hvis beløbet er negativt, skal brugeren se en besked om fejlagtigt input.
3. Ellers skal programmet fortsætte som tidligere med indtastning af valutakurs, beregning og visning af beløbet i danske kroner.



Opgave: Stemmeberettiget

Lav et program, som beder brugeren indtaste sin alder. Hvis personen er under 18 år, skal der vises beskeden "Du må ikke stemme ved folketingsvalg. "Ellers skal der vises beskeden "Du må gerne stemme ved folketingsvalg."



Øvelse: Mere end to grene

1. Læs om anvendelsen af switch-sætningen via linket i oversigten herunder.
2. Lav et program, som viser brugerens aldersgruppe ud fra indtastet alder:
 - 0-12 år: Barn
 - 13-19 år: Teenager
 - 20-64 år: Voksen
 - 65 år eller ældre: Pensionist

Switch

Hvis et program skal kunne forgrene sig i mere end to grene baseret på værdien af en variabel, kan du med fordel bruge en switch-sætning.

Der findes ikke en blok for en switch-sætning i Scratch, så der må du bruge flere "hvis"-blokke.

| Kilde | Beskrivelse |
|---|-----------------------------|
| switch JavaScript MDN | Switch-sætning i JavaScript |
| PHP: switch (PHP.net) | Switch-sætning i PHP |
| switch (processing.org) | Switch-sætning i Processing |

While-løkker

En af styrkerne ved en computer er dens evne til at udføre en handling gentagende gange på kort tid. Mennesker kan også gentage den samme handling om og om igen, men det er forbundet med risiko for fejl, og det tager forholdsvis lang tid for et menneske at udføre mange handlinger.

En computer kan programmeres til at udføre præcis den samme handling eller varianter af den samme handling et vilkårligt antal gange på brøkdele af et sekund.

I dette afsnit skal du lære, hvilken kontrolstruktur bruges til at få et program til at gentage handlinger.

Anvendelse af løkker

Løkker bruges til mange formål. Her er en række eksempler på anvendelsesområder:

- Indlæsning (og bearbejdning) af linjerne i en fil med data for en app
- Beregning af summen af en række værdier i en liste
- Sortering af en liste af fødselsdatoer
- Afspilning af et musiknummers data
- Finde mellemrum i et digt

Eksempel: Flere par sko

Når du køber varer, afhænger prisen per vare af antal købte varer. Det er også tilfældet i eksemplet med sko fra en amerikansk webshop. Det er nemlig hele forsendelsen af samme type sko, som pålægges importgebyr samt told ved en samlet værdi over 1.150 kr.

Derfor giver det god mening at beregne den gennemsnitlige pris ved køb af flere par sko. Så kan du vurdere, hvor mange venner skal gå sammen om en bestilling for at få den laveste pris.

Til det formål er en løkke oplagt at bruge, da beregningen af gennemsnittet bruger samme formel hver gang.



Opgave: Anvende programmet Flere par sko

1. Brug programmet herunder til at lave gennemsnitsberegninger for \$100 per par og en kurs på 5.
2. Overvej hvordan du ville skrive koden til programmet.
3. Se videoen herunder, der forklarer processen med udvikling af koden.

Kode: Flere par sko

Scratch (Sko 3 løkke.sb2)



JavaScript (sko3while.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 3: While-løkke</title>
</head>

<body>
<p>Prisen i DKK er:
<ul>
<li>1 par sko: <span id="prcDkk1"></span></li>
<li>2 par sko: <span id="prcDkk2"></span></li>
<li>3 par sko: <span id="prcDkk3"></span></li>
<li>4 par sko: <span id="prcDkk4"></span></li>
</ul>
</p>
<script type="text/javascript">
var prcUsd = prompt('Hvad koster dine sko i USD?');
var rateUsd =
    prompt('Hvad koster 1 USD i DKK (f.eks. 7.01)?');
var pricesDkk = []; // Array til gennemsnitspriser
var i = 1; // Tæller til løkke
var prcDkk;
while (i != 5) { // Beregning af priser
    prcDkk = prcUsd * rateUsd * i; // Pris for i par
    if (prcDkk > 80) { // Varen fortoldes
        if (prcDkk > 1150) {
            prcDkk = prcDkk * 1.17; // Told
        }
        prcDkk = prcDkk * 1.25; // Moms
        prcDkk = prcDkk + 160; // Importgebyr
    }
    pricesDkk[i-1] = prcDkk / i; // Gennemsnitlig pris
    i = i + 1; // Tæller forøges
}
i = 1;
while (i != 5) { // Visning af priser
    document.getElementById('prcDkk'+i).innerHTML =
        pricesDkk[i-1];
    i = i + 1;
}
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

PHP (sko3while.php)

```
<?php
/* Eksempel på kald til programmet:
   sko3while.php?prcUsd=200&rateUsd=7.01 */
\$pricesDkk = []; // Array til gennemsnitspriser
\$i = 1; // Tæller til løkke
while (\$i != 5) { // Beregning af priser
    // Pris for i par sko:
    \$prcDkk = \$_GET['prcUsd'] * \$_GET['rateUsd'] * \$i;
    if (\$prcDkk > 80) { // Varen fortoldes
        if (\$prcDkk > 1150) {
            \$prcDkk = \$prcDkk * 1.17; // Told
        }
        \$prcDkk = \$prcDkk * 1.25; // Moms
        \$prcDkk = \$prcDkk + 160; // Importgebyr
    }
    \$pricesDkk[\$i-1] = \$prcDkk / \$i; // Gennemsnitspris
    \$i = \$i + 1; // Tæller forøges
}
\$htmlPrices = ""; // HTML-kode indsat længere nede
\$i = 1;
while (\$i != 5) { // Visning af priser
    \$htmlPrices .= "<li> " . \$i . " par sko: " .
        \$pricesDkk[\$i-1] . "</li>";
    \$i = \$i + 1;
}
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 3: While-løkke</title>
</head>

<body>
<p>Prisen i DKK er:
<ul>
    <?= \$htmlPrices ?>
</ul>
</p>
```

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

Processing (sko3while.pde)

```
int prcUsd;
float rateUsd, prcDkk;
// Array til gennemsnitspriser:
float[] pricesDkk = new float[4];

void setup() {
    size(800,450);
    textSize(32); fill(255);
}

void draw() {
    background(0);
    rateUsd = 3 + float(mouseX)/100.0;
    prcUsd = mouseY;
    text("Hvad koster dine sko i USD (lodret): "
        + prcUsd, 50, 100);
    text("Hvad koster 1 USD i DKK (vandret): "
        + rateUsd, 50, 150);
}

void mousePressed() {
    noLoop();
    int i = 1; // Tæller til løkke
    while (i != 5) { // Beregning af priser
        prcDkk = prcUsd * rateUsd * i; // Pris: i par
        if (prcDkk > 80) { // Varen fortoldes
            if (prcDkk > 1150) {
                prcDkk = prcDkk * 1.17; // Told
            }
            prcDkk = prcDkk * 1.25; // Moms
            prcDkk = prcDkk + 160; // Importgebyr
        }
        pricesDkk[i-1] = prcDkk / i; // Gennemsnitspris
        i = i + 1;
    }
    i = 1;
    while (i != 5) { // Visning af priser
        text("Prisen i DKK er: " +
            pricesDkk[i-1], 50, 200 + i * 50);
        i = i + 1;
    }
}
```

{}

While-løkke

En while-løkke er en kontrolstruktur hvor en blok af sætninger gentages, så længe en given betingelse er sand.

While-løkkens opbygning

While (A = TRUE) Do

B

End While

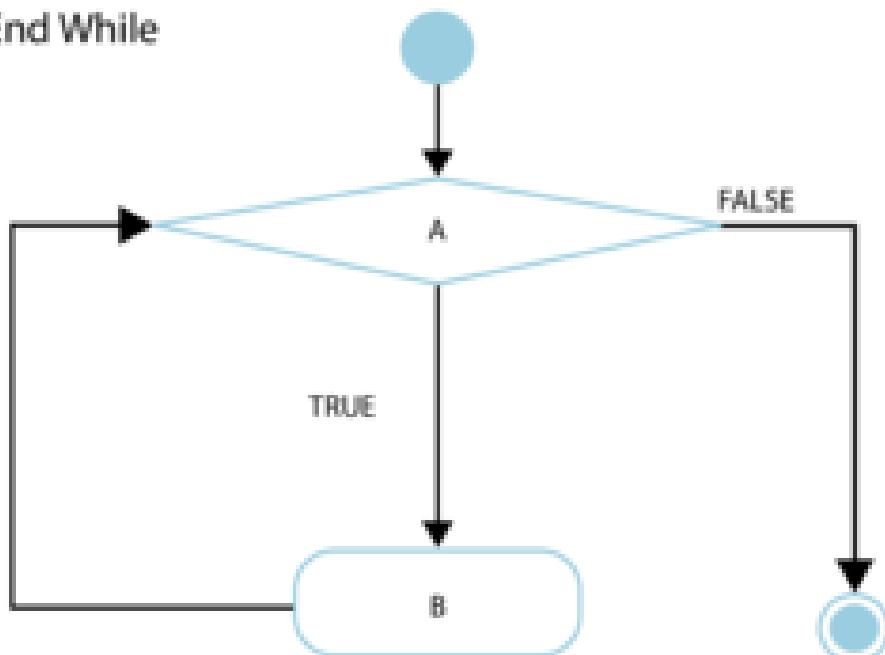
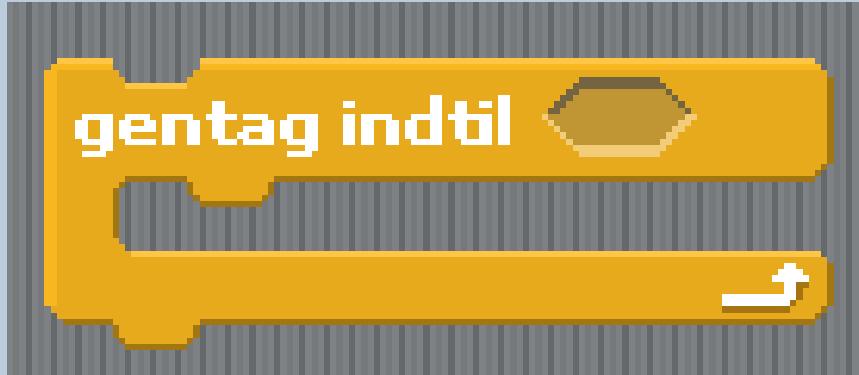


Diagram: Løkke

På figuren til højre kan du se while-løkkens generelle opbygning. En betingelse, A, evalueres til sandt eller falsk. Hvis A er sand, udføres blokken af sætninger, B, og A evalueres påny. Dette fortsætter, indtil A er falsk.

I eksemplet ovenfor bruges et heltal som en tæller til at holde styr på antallet af gennemløb i løkken. En sådan tæller skal altid sættes til en startværdi inden løkken og ændres inde i løkken.



Bemærk at i Scratch er der tale om en lidt anden gentagelsesstruktur end While-strukturen. Her gentages løkken, *indtil* en given betingelse er sand, i modsætning til de andre sprog, hvor det er *så længe* betingelsen er sand.



Opgave: Udvidelse af Flere par sko

Ret i programmet, så det i stedet udskriver den gennemsnitlige pris for 2, 4, 6 og 8 par sko.



Opgave: Syvtabellen

Lav et program, som viser syvtabellen: 7, 14, 21, 28...

Uendelig løkke

Hvis betingelsen i en løkke altid er sand, vil løkken aldrig stoppe. Det kan eksempelvis ske, hvis man glemmer at ændre en variabels værdi, og denne bruges i betingelsen.

I så fald vil fortolkeren aldrig stoppe, og man er nødt til at afbryde den.

Det er dog også i nogle tilfælde ønskeligt, at en løkke fortsætter i det uendelige. F.eks. findes der i Scratch en blok til dette.

En løkke har nogle strukturelle ligheder med en forgrening. I begge kontrolstrukturer bliver en betingelse evalueret til sand eller falsk. I en forgrening afvikles en række instruktioner én gang, hvis betingelsen er sand. I en løkke afvikles en række instruktioner alle de gange, hvor betingelsen er sand.

En løkke kan derfor betragtes som en forgrening, hvor der inde i forgreningens "indmad" hele tiden springes tilbage til betingelsen.



Opgave: While-løkke med gentagende sætninger

Lav et program med en while-løkke, der 20 gange skriver "Jeg gentager ofte mig selv."

Vilkårligt antal gennemløb

Det er ikke altid muligt at forudsige antallet af gentagelser i en while-løkke. Hvis betingelsen for at fortsætte gentagelsene ikke afhænger af optælling af en variabel, kan det ikke forudsiges, hvornår løkkken stopper.

Eksempelvis kan løkkens betingelse afhænge af et brugerinput. Der kan være behov for at fortsætte en løkke, så længe en bruger *ikke* taster et bestemt input.

I øvelsen herunder skal du bruge en kommando til at spørge brugeren om et input, og dette input skal indgå i løkkens betingelse.



Opgave: Løkke med brugerindtastning

Lav et program, hvor brugeren enten kan indtaste et tal, hvorved kvadratet til tallet vises og et nyt kan indtastes – eller indtaste q, hvorved programmet stopper.

På følgende sider fra den officielle dokumentation kan du læse mere om while-løkker.

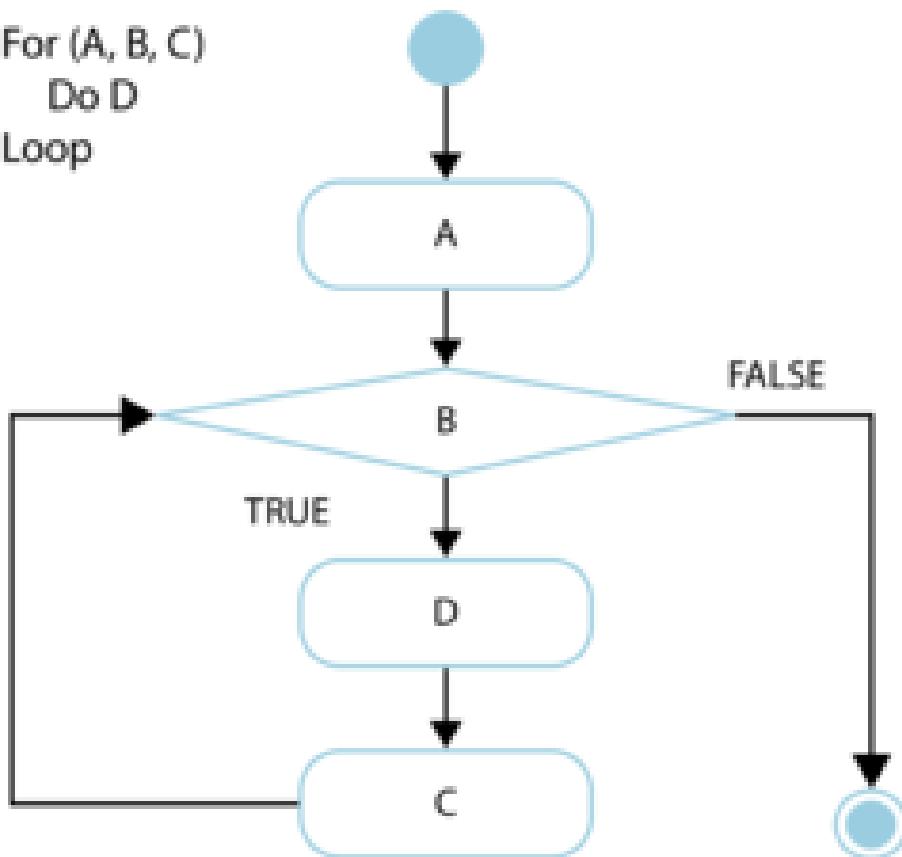
| Kilde | Beskrivelse |
|--|---------------------------------------|
| Repeat until () - Scratch Wiki | While-løkke i Scratch (gentag indtil) |
| MDN JavaScript while | While-løkke i JavaScript |
| PHP.net while | While-løkke i PHP |
| processing.org while | While-løkke i Processing |

For-løkker

For (A, B, C)

Do D

Loop



For-løkke

Når man på forhånd ved hvor mange gentagelser man vil have, kan man med fordel anvende en for-løkke i stedet for en while-løkke.

Eksempel: Flere par sko med for-løkke

Vi vil nu omstrukturere programmet Flere par sko, så den bruger en for-løkke i stedet for en while-løkke.

Scratch har ikke en blok til en for-løkke.

Syntaksen for en for-løkke i JavaScript, PHP og Processing er ens, og kun anvendelse af variable varierer. Koden ses herunder.



Opgave: Flere par sko med for-løkke

1. Se videoen herunder, der forklarer processen med udvikling af koden.
2. Sammenlign koden med while-løkken i forrige afsnit.

Kode: Flere par sko med for-løkke

JavaScript (sko4for.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 4: For-løkke</title>
</head>

<body>
<p>Prisen i DKK er:
<ul>
<li>1 par sko: <span id="prcDkk1"></span></li>
<li>2 par sko: <span id="prcDkk2"></span></li>
<li>3 par sko: <span id="prcDkk3"></span></li>
<li>4 par sko: <span id="prcDkk4"></span></li>
</ul>
</p>
<script type="text/javascript">
var prcUsd = prompt('Hvad koster dine sko i USD?');
var rateUsd =
    prompt('Hvad koster 1 USD i DKK (f.eks. 7.01?)');
var pricesDkk = [] // Array til gennemsnitspriser
var prcDkk;
for (var i = 1; i != 5; i = i + 1) { // Beregning
    prcDkk = prcUsd * rateUsd * i; // Pris for i par
    if (prcDkk > 80) { // Varen fortoldes
        if (prcDkk > 1150) {
            prcDkk = prcDkk * 1.17; // Told
        }
        prcDkk = prcDkk * 1.25; // Moms
        prcDkk = prcDkk + 160; // Importgebyr
    }
    pricesDkk[i-1] = prcDkk / i; // Gennemsnitlig pris
}
for (var i = 1; i != 5; i = i + 1) { // Visning
    document.getElementById('prcDkk'+i).innerHTML =
        pricesDkk[i-1];
}
</script>
</body>
</html>
```

PHP (sko4for.php)

```
<?php
/* Eksempel på kald til programmet:
   sko4for.php?prcUsd=200&rateUsd=7.01 */
\$pricesDkk = []; // Array til gennemsnitspriser
for (\$i = 1; \$i != 5; \$i = \$i + 1) { // Beregninger
    // Pris for i par sko:
    \$prcDkk = \$_GET['prcUsd'] * \$_GET['rateUsd'] * \$i;
    if (\$prcDkk > 80) { // Varen fortoldes
        if (\$prcDkk > 1150) {
            \$prcDkk = \$prcDkk * 1.17; // Told
        }
        \$prcDkk = \$prcDkk * 1.25; // Moms
        \$prcDkk = \$prcDkk + 160; // Importgebyr
    }
    \$pricesDkk[\$i-1] = \$prcDkk / \$i; // Gennemsnit
}
\$htmlPrices = ""; // HTML-kode indsat længere nede
for (\$i = 1; \$i != 5; \$i = \$i + 1) { // Visning
    \$htmlPrices .= "<li> " . \$i . " par sko: " .
        \$pricesDkk[\$i-1] . "</li>";
}
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 4: For-løkke</title>
</head>

<body>
<p>Prisen i DKK er:
<ul>
    <?= \$htmlPrices ?>
</ul>
</p>
</body>

</html>
```

Processing (sko4for.pde)

```
int prcUsd;
float rateUsd, prcDkk;
// Array til gennemsnitspriser:
float[] pricesDkk = new float[4];

void setup() {
    size(800,450);
    textSize(32); fill(255);
}

void draw() {
    background(0);
    rateUsd = 3 + float(mouseX)/100.0;
    prcUsd = mouseY;
    text("Hvad koster dine sko i USD (lodret): "
        + prcUsd, 50, 100);
    text("Hvad koster 1 USD i DKK (vandret): "
        + rateUsd, 50, 150);
}

void mousePressed() {
    noLoop();
    for (int i = 1; i != 5; i = i + 1) { // Beregning
        prcDkk = prcUsd * rateUsd * i; // Pris: i par
        if (prcDkk > 80) { // Varen fortoldes
            if (prcDkk > 1150) {
                prcDkk = prcDkk * 1.17; // Told
            }
            prcDkk = prcDkk * 1.25; // Moms
            prcDkk = prcDkk + 160; // Importgebyr
        }
        pricesDkk[i-1] = prcDkk / i; // Gennemsnitspris
    }
    for (int i = 1; i != 5; i = i + 1) { // Beregning
        text("Prisen i DKK er: " +
            pricesDkk[i-1], 50, 200 + i * 50);
    }
}
```

For-løkke

En for-løkke er en kontrolstruktur, hvor en blok af sætninger gentages for hvert tal i et interval eller hvert element i en datastruktur.



Opgave: Udvidelse af Flere par sko

Ret i programmet, så det i stedet udskriver den gennemsnitlige pris for 2, 4, 6 og 8 par sko.

Indlejrede løkker

Hvis du vil lave et program, som kan vise de mulige kombinationer af terningekast med to terninger, har du brug for at kombinere de seks muligheder for første terning med de seks muligheder for anden terning. I så fald er der brug for en *indlejret løkke*, som er en løkke inden i en løkke. Bemærk at der skal bruges to forskellige variabler, i og j, da der er to løkker.

Eksempel i php

```
\$html = "";
for (\$i = 1; \$i <= 6; \$i = \$i + 1) {
    for (\$j = 1; \$j <= 6; \$j = \$j + 1) {
        \$html .= "<li>" . \$i . ", " . \$j . "</li>";
    }
}
```



Opgave: Løkke med 10 kvadrattal

Lav et program, som med en løkke viser de første 10 kvadrattal (resultatet af et heltal ganget med sig selv: $1*1=1$, $2*2=4$, $3*3=9$...) på skærmen.

Eksempel: Løkke som tæller ned

En løkkes tæller behøver ikke starte ved 0 og tælle opad. Den kan have en vilkårlig startværdi og den kan opdateres som ønsket i hver gentagelse.

I eksemplet herunder ses et program i Processing, som starter med værdien 2016 og tæller ned med 4 i hver gentagelse. Dermed viser den 10 skudår fra år 2016 og tilbage som følgende output:

```
2016 var et skudår.  
2012 var et skudår.  
2008 var et skudår.  
2004 var et skudår.  
2000 var et skudår.  
1996 var et skudår.  
1992 var et skudår.  
1988 var et skudår.  
1984 var et skudår.  
1980 var et skudår.
```

Kode: Skudår i Processing

```
for (int year = 2016; year > 1979; year = year-4) {  
    println(year + " var et skudår.");  
}
```



Opgave: Løkke med nedtælling

Lav et program, som udskriver "Der er 10 linjer tilbage. Der er 9 linjer tilbage... Der er 0 linjer tilbage."

Gentagelse over arrays

Når ethvert element i et array skal behandles, kan det foregå ved gentagelse i en for-løkke. Herunder er en oversigt over muligheden i hvert sprog. Scratch har ikke mulighed for dette.

| Kilde | Beskrivelse |
|---|--------------------------------|
| for...of JavaScript MDN | Iteration over arrays med mere |
| PHP: foreach (PHP.net) | Iteration over arrays |
| for (processing.org) | Iteration over arrays |

På følgende sider fra den officielle dokumentation kan du læse mere om for-løkker.

| Kilde | Beskrivelse |
|------------------------------------|------------------------|
| MDN JavaScript for | For-løkke i JavaScript |
| PHP.net for | For-løkke i PHP |
| processing.org for | For-løkke i Processing |

Funktioner

Når du programmerer, vil du før eller siden opdage, at du kommer til at gentage et mønster i koden, fordi du ønsker at gøre nogenlunde det samme flere steder i koden. Det har imidlertid nogle ulemper. Vigtigst af alt er koden sværere at vedligeholde, fordi kode skal rettes flere steder ved fejl eller ændringer. Det gør arbejdet mere omstændigt, og der er risiko for, at koden ikke bliver rettet alle steder. Endelig vil det fyldes flere linjer kode.

Som en løsning på dette anvendes en *funktion*, som du i dette afsnit vil lære at definere og anvende. En funktion *generaliserer* kode ved hjælp af *parametere*. Begrebet har også andre navne, for eksempel procedure og subroutine, men vi vil i denne bog bruge begrebet funktion.

Eksempel: Sko fra Kina



Colourbox.com

Skoprogrammet skal nu udvides, så de samme beregninger udføres for kinesiske priser. Brugeren skal altså også indtaste prisen på et par sko i den kinesiske valuta, *yuan*, og dennes valutakurs.

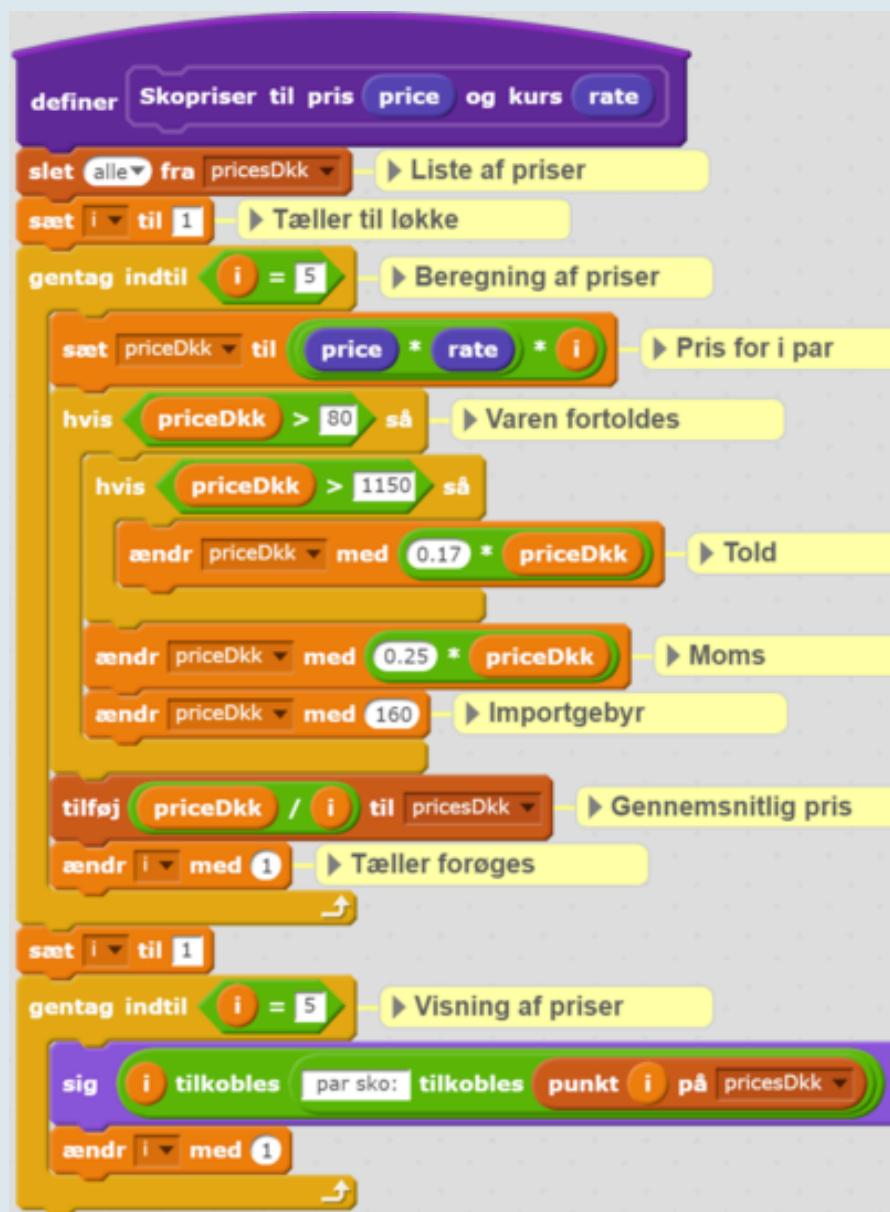


Opgave: Anvende programmet Sko fra Kina

1. Brug programmet (se filen herunder) til at lave beregninger for USA og Kina.
2. Overvej hvordan du ville skrive koden til programmet.
3. Se videoen herunder, der forklarer processen med udvikling af koden.

Kode: Sko fra Kina

Scratch (Sko 5 funktion.sb2)





JavaScript (sko5funktion.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 5: Funktion</title>
</head>

<body>
<h1>Skopriser i USA</h1>
<p>Prisen i DKK er:
<ul>
<li>1 par sko: <span id="prcDkkUS1"></span>
</li>
<li>2 par sko: <span id="prcDkkUS2"></span>
</li>
<li>3 par sko: <span id="prcDkkUS3"></span>
</li>
<li>4 par sko: <span id="prcDkkUS4"></span>
</li>
</ul>
</p>
<h1>Skopriser i Kina</h1>
<p>Prisen i DKK er:
<ul>
<li>1 par sko: <span id="prcDkkCh1"></span>
</li>
<li>2 par sko: <span id="prcDkkCh2"></span>
</li>
<li>3 par sko: <span id="prcDkkCh3"></span>
</li>
<li>4 par sko: <span id="prcDkkCh4"></span>
</li>
</ul>
</p>
<script type="text/javascript">
/* Returnerer array med gennemsnitspriser 1-4 */
function prices(price, rate) {
    var pricesDkk = [] // Array til gennemsnit
    for (var i = 1; i != 5; i = i + 1) { // Beregn
        prcDkk = price * rate * i; // Pris: i par
```

```
if (prcDkk > 80) { // Varen fortoldes
    if (prcDkk > 1150) {
        prcDkk = prcDkk * 1.17; // Told
    }
    prcDkk = prcDkk * 1.25; // Moms
    prcDkk = prcDkk + 160; // Importgebyr
}
pricesDkk[i-1] = prcDkk / i; // Gennemsnitspris
}
return pricesDkk;
}

// USA
var prcUsd = prompt('Hvad koster dine sko i USD?');
var rateUsd =
    prompt('Hvad koster 1 USD i DKK (f.eks. 7.01)?');
var pricesUS = prices(prcUsd, rateUsd);
for (var i = 1; i != 5; i = i + 1) { // Visning
    document.getElementById('prcDkkUS' + i).
        innerHTML = pricesUS[i-1];
}

// Kina
var prcYuan = prompt('Hvad koster dine sko i yuan?');
var rateYuan =
    prompt('Hvad koster 1 yuan i DKK (f.eks. 1.05)?');
var pricesCh = prices(prcYuan, rateYuan);
for (var i = 1; i != 5; i = i + 1) { // Visning
    document.getElementById('prcDkkCh' + i).
        innerHTML = pricesCh[i-1];
}
</script>
</body>
</html>
```

PHP (sko5funktion.php)

```
<?php
/* Eksempel på kald til programmet:
sko5funktion.php?prcUsd=200&rateUsd=7.01&
prcYuan=300&rateYuan=1.05 */

/* Returnerer array med gennemsnitspriser 1-4 */
function prices($price, $rate) {
    $pricesDkk = []; // Array til gennemsnitspriser
    for ($i = 1; $i != 5; $i = $i + 1) { // Beregninger
        // Pris for i par sko:
        $prcDkk = $price * $rate * $i;
        if ($prcDkk > 80) { // Varen fortoldes
            if ($prcDkk > 1150) {
                $prcDkk = $prcDkk * 1.17; // Told
            }
            $prcDkk = $prcDkk * 1.25; // Moms
            $prcDkk = $prcDkk + 160; // Importgebyr
        }
        $pricesDkk[$i-1] = $prcDkk / $i; // Gennemsnit
    }
    return $pricesDkk;
}
// USA
$pricesUS = prices($_GET['prcUsd'], $_GET['rateUsd']);
$htmlPricesUS = ""; // HTML-kode indsat længere nede
$i = 1;
for ($i = 1; $i != 5; $i = $i + 1) { // Visning
    $htmlPricesUS .= "<li> " . $i . " par sko: " .
        $pricesUS[$i-1] . "</li>";
}
// Kina
$pricesCh = prices($_GET['prcYuan'], $_GET['rateYuan']);
$htmlPricesCh = ""; // HTML-kode indsat længere nede
$i = 1;
for ($i = 1; $i != 5; $i = $i + 1) { // Visning
    $htmlPricesCh .= "<li> " . $i . " par sko: " .
        $pricesCh[$i-1] . "</li>";
}
?>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Sko, udgave 5: Funktion</title>
</head>

<body>
<h1>Skopriser i USA</h1>
<p>Prisen i DKK er:
<ul>
<?= \$htmlPricesUS ?>
</ul>
</p>
<h1>Skopriser i Kina</h1>
<p>Prisen i DKK er:
<ul>
<?= \$htmlPricesCh ?>
</ul>
</p>
</body>

</html>
```

Processing (sko5funktion.pde)

```
int price; float rate;
boolean usaSet = false;
float[] pricesUS = new float[4];
float[] pricesCh = new float[4];

/* Returnerer array med gennemsnitspriser 1-4 */
float[] prices(int price, float rate) {
    float[] prices = new float[4];
    for (int i = 1; i != 5; i = i + 1) { // Beregning
        float prcDkk = price * rate * i; // Pris: i par
        if (prcDkk > 80) { // Varen fortoldes
            if (prcDkk > 1150) {
                prcDkk = prcDkk * 1.17; // Told
            }
            prcDkk = prcDkk * 1.25; // Moms
            prcDkk = prcDkk + 160; // Importgebyr
        }
        prices[i-1] = prcDkk / i; // Gennemsnitspris
    }
    return prices;
}

void setup() {
    size(800,450);
    textSize(24); fill(255);
}

void draw() {
    background(0);
    rate = float(mouseX)/100.0;
    price = mouseY;
    text("Hvad koster dine sko i " +
        (usaSet ? "yuan" : "USD") + " (lodret): " +
        + price, 50, 50);
    text("Hvad koster 1 " + (usaSet ? "yuan" : "USD") +
        + " i DKK (vandret): " +
        rate, 50, 75);
}

void mousePressed() {
```

```
if (!usaSet) {  
    usaSet = true;  
    // Beregn for USA  
    pricesUS = prices(price, rate);  
} else {  
    noLoop();  
    // Beregn for Kina  
    pricesCh = prices(price, rate);  
    // Vis alle priser  
    for (int i = 1; i != 5; i = i + 1) { // Beregning  
        text("USA - Prisen for " + i + " par er: " +  
            pricesUS[i-1], 50, 100 + i * 25);  
        text("Kina - Prisen for " + i + " par er: " +  
            pricesCh[i-1], 50, 225 + i * 25);  
    }  
}
```

Funktion

En funktion er en række instruktioner indkapslet i en enhed, som løser en opgave – eventuelt med input og/eller output.

En funktion kan optræde i koden på to måder: Definition og anvendelse.

Ved definitionen af en funktion skal du angive:

- Funktionens navn
- Parametrenes navne
- Eventuelt datatypen for returværdien

Ved anvendelse af funktionen siges det også, at man *kalder* funktionen. Her angives navnet og de nødvendige argumenter.

Argument

Et argument er det faktiske datainput til en funktion.

Eksempel: Argument

Følgende kode i JavaScript beregner den dobbelte værdi af et tal:

```
function double(x) { return 2*x; }

double(5);
```

Her er værdien 5 argumentet til funktionen double.

Parameter

En parameter er en variabel, som henviser til et argument til en funktion.

Eksempel: Parameter

Følgende kode i JavaScript beregner den dobbelte værdi af et tal:

```
function double(x) { return 2*x; }

double(5);
```

Her er variablen x en parameter til funktionen double.



Øvelse: Sko fra endnu et land

1. Vælg et land uden for EU, og find dets valutakurs i forhold til DKK.
2. Udvid programmet ovenfor med det nye land.
3. Ret programmet, så det kun viser gennemsnitsprisen for hhv. 1, 2 og 3 par sko.
4. Lav en funktion, som kan vise priserne for et land, og kald denne funktion.

På følgende sider fra den officielle dokumentation kan du læse mere om funktioner.

| Kilde | Beskrivelse |
|--|-------------------------|
| Custom block - Scratch Wiki | Funktioner i Scratch |
| MDN JavaScript function | Funktioner i JavaScript |
| PHP.net User-defined functions | Funktioner i PHP |
| processing.org return | Funktioner i Processing |

Data og operationer

En fundamental del af it-systemer er de data, som i en eller anden form bliver manipuleret (ændret) i it-systemet og udvekslet med brugeren. I et program repræsenteres data i en variabel. For at få en forståelse af, hvilke variable kan indgå i et program, betragter vi et eksempel på et velkendt it-system.

Eksempel: Variable i Facebook



iStockphoto.com/bombuscreative

Facebook behandler hvert sekund store mængder af data. I kørslen af de programmer, som resulterer i de viste hjemmesider, benyttes variable til midlertidigt at gemme data-værdier, som i sidste ende skal vises på skærmen eller gemmes i en database.

Eksempelvis vil der i programmet være en variabel, som indeholder dit navn øverst i Facebook. Derudover kunne der være en variabel, som indeholder antallet af notifikationer.

Variabel

En variabel er et navn og et stykke lagerplads, som kan indeholde en værdi.

For at et program kan arbejde med data, er computeren nødt til at reservere en del af hukommelsen til arbejdet. Derudover skal variablen have et navn, så det er muligt at henvisе til den.

En variabel kan grundlæggende bruges på tre måder:

1. Den kan *erklæres*, hvilket betyder at der skabes plads i hukommelsen samt gives et unikt navn og eventuelt en datatype (se mere i afsnittet [Datatyper](#)).

2. Den kan *tildelles* en værdi.
3. Dens værdi kan anvendes.

Det er i mange programmeringssprog nødvendigt at erklære en variabel, før den kan tildeles en værdi eller anvendes.

Eksempel: Erklæring og tildeling af værdi

I nedenstående JavaScript-kode erklæres variablen age i første linje, og i anden linje tildeles den værdien 18.

```
var age;
```

```
age = 18;
```

En variabel kan også tildeles en værdi i forbindelse med erklæringen:

```
var age = 18;
```

Tal



Colourbox.com

Vi vil starte med at betragte programmer, som primært anvender data af typen tal og beregninger på disse.

Eksempler på anvendelse af tal i programmer:

- Rejser: Omregner af beløb fra en valuta til en anden - Hvor meget koster et par sko til 69 euro i danske kroner?
- Fysik: Beregning af effekt for omsat energi over en periode - Hvad er effekten af en brødrister, som på 60 sekunder bruger 30.000 Joule?
- Forsøg: Statistiske beregninger til at undersøge hypoteser - Kan middelværdien for to datasæt med god sandsynlighed siges at være ens?



Opgave: Anvendelse af valutaomregner

1. Hent koden til køb af sko [her](#).
2. Hvilke tal indgår i programmet, og hvilke operatorer (+, -, *, /) er nødvendige for at lave omregningen?

Tal

Et tal er en datatype til at indeholde tal.



Opgave: Udvidelse af valutaomregner

1. Ret i programmet, så det kontrollerer, at beløbet er gyldigt (ikke negativt).
2. Udvid programmet, så det også kan regne den modsatte vej.

Operatorer for tal

Fra matematik kender vi de aritmetiske operatorer plus, minus, gange og dividere. Resultatet i alle tilfælde er et tal, og at symbolet for gange ofte er * i stedet for ::

- $5 \text{ (tal)} + 8 \text{ (tal)} = 13 \text{ (tal)}$
- $5 \text{ (tal)} - 8 \text{ (tal)} = -3 \text{ (tal)}$
- $5 \text{ (tal)} * 8 \text{ (tal)} = 40 \text{ (tal)}$
- $5 \text{ (tal)} / 8 \text{ (tal)} = 0,625 \text{ (tal)}$

Bemærk, at der er flere operatorer for tal end ovenstående.

På følgende sider fra den officielle dokumentation kan du læse mere om programmering med tal.

| Kilde | Beskrivelse |
|--|--|
| Number - Scratch Wiki | Variable og operatorer for tal i Scratch |
| MDN JavaScript data types | Datatypen tal i JavaScript |
| PHP.net Integers | Datatypen heltal i PHP |
| PHP.net Floating point numbers | Datatypen kommatal i PHP |
| processing.org int | Datatypen heltal i Processing |
| processing.org float | Datatypen kommatal i Processing |

Strenge



TV-quizzen Lykkehjulet
Colourbox.com

Lykkehjulet var en populær tv-quiz, der var inspireret af det amerikanske Wheel of Fortune og blev sendt på TV2 i 90'erne. Målet med quizzen var at gætte et ord inden for en given kategori, f.eks. et sted eller en person. Deltagerne skulle dreje på et hjul og efterfølgende gætte på en konsonant, som kunne være indeholdt i ordet. Hvis ordet indeholdt konsonanten, blev det afsløret hvor i ordet konsonanten optrådte, og spilleren kunne fortsætte sin runde.

I dette afsnit skal du arbejde videre på en simpel version af Lykkehjulet, som minder om Galgespillet (på engelsk: Hangman). I denne version kan der både gættes på konsonanter og vokaler, og målet er at bruge færrest muligt antal gæt.

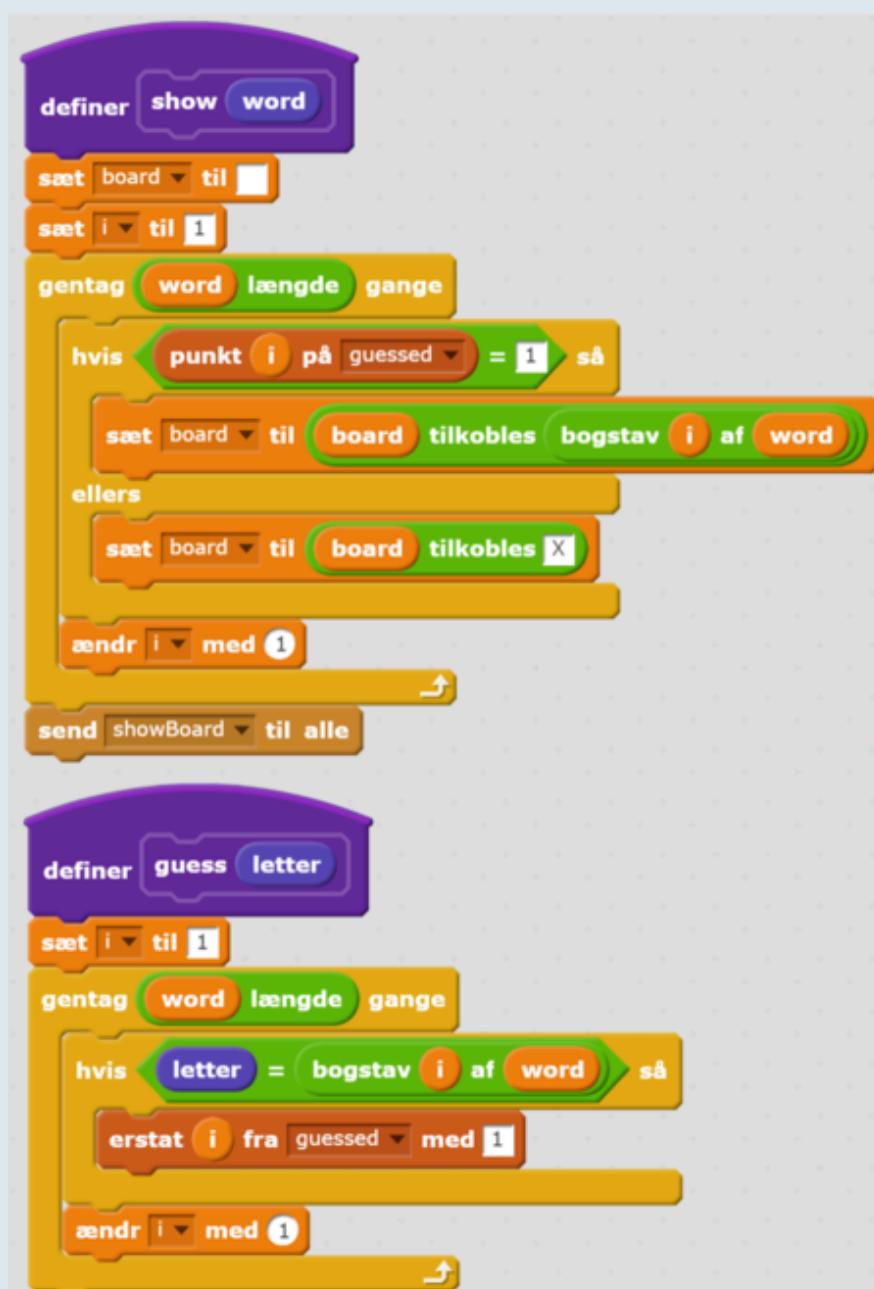


Opgave: Spille Lykkehjulet

1. Kør programmet (se filer herunder), og spil spillet.
2. Hvilke dele mangler spillet, før det fungerer efter hensigten?
3. Læs koden herunder, og læg mærke til de steder, hvor variablen word optræder.
Kan du regne ud, hvordan variablen bruges hvert sted?

Kode: Lykkehjulet

Scratch (Lykkehjulet.sb2)



JavaScript (lykkehjulet.html)

```
/* Lykkehjulet */
var word; // Hemmeligt ord
var attempts; // Antal forsøg
var category = "Dansk by"; // Kategori
document.write("<h2>Kategori: " + category + "</h2>");
// Mulige hemmelige ord
var words = ["København", "Aarhus", "Odense",
"Aalborg", "Esbjerg"];
/* Funktion til at returnere tilfældigt heltal */
function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min))
    + min;
}
// Vælg et tilfældigt ord som skal gættes.
word = words[getRandomInt(0, words.length)];
// Status på spillet i et array
var guessed = [];
/* Funktion til at opdatere tavlen */
function updateBoard() {
    var letter; // Bogstav i gennemløb af ord
    var newBoard = document.createElement("P");
    newBoard.appendChild(document.createTextNode("Tavlen: "));
    // Løb igennem alle bogstaver i ordet
    for (i = 0; i < word.length; i++) {
        if (guessed[i]) {
            letter = word.charAt(i);
        } else {
            // Vis en streg, hvis bogstavet ikke er gættet.
            letter = "_";
        }
        // Tilføj bogstaver på skærmen.
        letterNode = document.createTextNode(letter);
        newBoard.appendChild(letterNode);
    }
    document.body.appendChild(newBoard);
}
/* Gætter på et bogstav ved tastetryk */
var keyEvent = function(event) {
```

```
var letter;
for (i = 0; i < word.length; i++) {
    letter = word.substring(i, i+1).toLowerCase()
    .charCodeAt(0);
    // Undersøg for hvert tegn om det er gættet
    if (letter == event.keyCode) {
        guessed[i] = true;
    }
}
updateBoard();
}
document.body.
addEventListener("keypress", keyEvent, false);
// Start med at vise tavlen.
updateBoard();
```

Processing (lykkehjulet.pde)

```
/* Lykkehjulet */  
String word; // Hemmeligt ord  
int attempts; // Antal forsøg  
String category = "DANSK BY"; // Kategori  
  
// Mulige hemmelige ord  
String[] words = {"København", "Aarhus",  
    "Odense", "Aalborg", "Esbjerg"};  
  
// Status på spillet i et array  
boolean[] guessed;  
  
/* Viser vinduet og sætter ordet */  
void setup() {  
    // Vælg et tilfældigt ord som skal gættes.  
    word = words[int(random(words.length))];  
    guessed = new boolean[word.length()];  
    size(800, 450); // Vis vinduet  
    textSize(36); // Bogstavernes størrelse  
}  
  
/* Opdaterer skærmen */  
void draw() {  
    background(200, 200, 255); // Nulstiller  
    textAlign(LEFT); // Venstrestil kategorien  
    text(category, 105, 100); // Viser kategorien  
    textAlign(CENTER); // Centrer i hver kasse  
    char letter; // Bogstav i iteration af ord  
    // Løb igennem alle bogstaver i ordet  
    for (int i = 0; i < word.length(); i++) {  
        // Sæt bogstavet ind i en firkant.  
        fill(0, 0, 0); // Sort skrift  
        if (guessed[i]) {  
            letter = word.charAt(i);  
        } else {  
            letter = 20;  
        }  
        text(letter, 125 + i*50, 200);  
        noFill(); // Tomme kasser  
        rect(105 + i*50, 200 - 35, 45, 45);  
    }  
}
```

```

}

/* Gætter på et bogstav */
void keyPressed() {
    char letter;
    for (int i = 0; i < word.length(); i++) {
        letter = word.substring(i, i+1).toLowerCase()
            .charAt(0);
        // Undersøg for hvert tegn om det er gættet
        if (letter == key) guessed[i] = true;
    }
}

```

Streng

En streng er en datatype til at indeholde en række tegn.

Eksempel: Streng

I både koden fra Scratch og Processing optræder variablen word, som er af datatypen streng. Til at holde styr på de bogstaver, som er blevet gættet og dermed vendt på tavlen, bruges en liste af boolske værdier. Sådan en liste kaldes et array og vil blive forklaret i et senere afsnit.

I funktionerne guess (Scratch) og keyPressed (Processing) undersøges hvert tegn af ordet word. Hvis et tegn er lig med det bogstav, som der gættes på, sættes den boolske værdi til sand.

I funktionerne show (Scratch) og draw (Processing) gennemgås hvert tegn af ordet word for at afgøre, om bogstavet skal vises eller skjules. I Scratch skjules bogstavet med et stort X (hvorfor er det uhensigtsmæssigt?), og i Processing vises et tomt felt.

Der bruges altså to operationer på strengen: Adgang til et bestemt tegn i strengen og længden af strengen. Dette er to af de typiske operationer på en streng.

Anførselstegn til strengværdier

Værdier af datatypen streng angives i de fleste programmeringssprog med dobbelte ("") eller enkelte ('') anførselstegn. Ofte er dette nødvendigt for at adskille en streng fra en variabel.

Eksempel 1

```
hej = 1;
```

```
x = hej;
```

Eksempel 2

```
hej = 1;
```

```
x = "hej";
```

I Eksempel 1 er værdien af x lig med tallet 1. I Eksempel 2 er værdien af x lig med strengen "hej".

Bemærk at strengværdier i Scratch angives uden anførselstegn.

Globale og lokale variable

Der er forskel på, hvor en variabel er tilgængelig. En variabel kan enten være *global* eller *lokal*.

En global variabel er tilgængelig i hele programmet. Et eksempel på dette i koden ovenfor er variablen *word* i JavaScript og Processing.

En lokal variabel er kun tilgængelig inden for en funktion eller blok af kode, hvor den er erklæret. Et eksempel på dette i koden ovenfor er variablen *letter* i henholdsvis updateBoard (JavaScript) og keyPressed (Processing).



Opgave: Udvidelse af Lykkehjulet

Ovenstående version af Lykkehjulet har nogle mangler i forhold til den endelige version.

1. Indsæt din egen kategori, og udfyld *words* med nye ord.
2. Brug den forberedte variabel *attempts* til at holde styr på antallet af forsøg, og vis antallet for brugeren.
3. Gør det muligt for brugeren at gætte på hele ordet ud over kun at gætte på et bogstav.

Operatorer for strenge

Der eksisterer typisk én operator for strenge: Sammensætning af to strenge. Eksempelvis bliver sammensætningen af "Hej " og "du" til "Hej du"



Opgave: Bagvendt ord

Lav en funktion *reverse*, som ud fra en streng i input returnerer den bagvendte streng (skrevet bagfra) af inputtet som output.

På følgende sider fra den officielle dokumentation kan du læse mere om arbejde med strenge.

| Kilde | Beskrivelse |
|---------------------------------------|--------------------------------------|
| String - Scratch Wiki | Eksempler og funktioner i Scratch |
| MDN JavaScript String | Eksempler og operatorer i JavaScript |
| PHP.net Strings | Eksempler og operatorer i PHP |
| processing.org String | Eksempler og metoder i Processing |

Sandhedsværdier

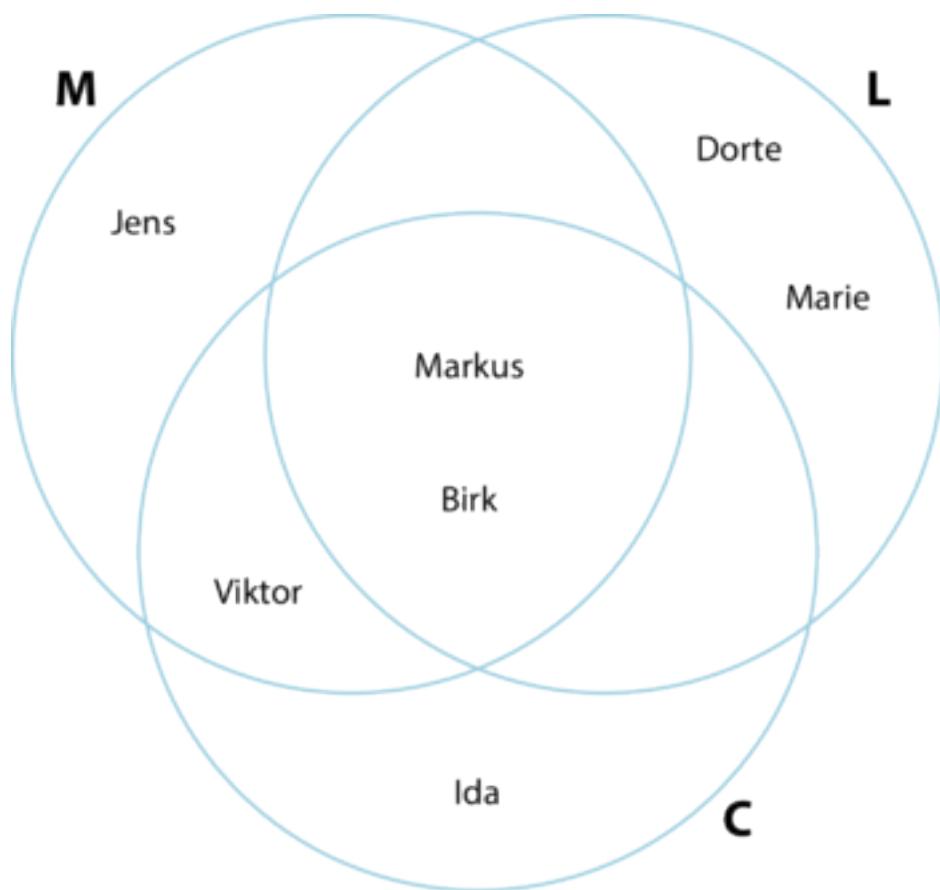
Man kan få brug for at have en variabel, der enten er sand eller falsk. Variablen har altså en sandedsværdi. Sådan en variabel kaldes i programmering for en boolsk variabel opkaldt efter den engelske matematiker George Boole.

Eksempler på anvendelse af sandhedsværdier i programmer:

- Skudår: Afgørelse om et år er skudår.
- Fysik: Afgørelse om vandet er så varmt at det koger.
- Samfundsfag: Afgørelse om den politiske kandidat skal have et mandat eller ej.

Ofte bruges sandhedsværdier i forgreninger og løkker.

Eksempel: Mængder



Denne figur viser tre mængder:

- M: Personer af hunkøn
- C: Personer under 18 år
- L: Venstrehåndede personer

Nedenstående program definerer de tre mængder og implementerer funktioner som kan besvare følgende spørgsmål:

- Er en given person er hunkøn?
- Er en given person en dreng?
- Er en given person en kvinde eller et barn?

Under koden kan du se en oversigt over de operatorer, som kan anvendes på sandhedsværdier.

Kode: Mængder

JavaScript (boolske_variable.html)

```
/** Beregner egenskaber ud fra boolsk algebra ***/  
// Array til mænd  
var M = ["Jens", "Viktor", "Birk", "Markus"];  
// Array til børn  
var C = ["Viktor", "Birk", "Markus", "Ida"];  
// Array til venstrehåndede  
var L = ["Birk", "Markus", "Dorte", "Marie"];  
console.log(isFemale("Jens"));  
console.log(isFemale("Dorte"));  
console.log("---");  
console.log(isBoy("Jens"));  
console.log(isBoy("Dorte"));  
console.log(isBoy("Markus"));  
console.log("---");  
console.log(isWomanOrChild("Jens"));  
console.log(isWomanOrChild("Dorte"));  
console.log(isWomanOrChild("Markus"));  
  
/* Undersøger om name er hunkøn. */  
function isFemale(name) {  
    return !contains(M, name);  
}  
  
/* Undersøger om name er en dreng. */  
function isBoy(name) {  
    return contains(M, name) && contains(C, name);  
}  
  
/* Undersøger om name er kvinde eller barn */  
function isWomanOrChild(name) {  
    return !contains(M, name) || contains(C, name);  
}  
  
/* Undersøger om array indeholder element. */  
function contains(array, element) {  
    for (var i = 0; i < array.length; i++) {  
        // Returner sand, hvis dette er elementet.  
        if (array[i] == element) return true;  
    }  
    // Returner falsk, hvis den ikke blev fundet.
```

```
    return false;  
}
```

PHP (boolean.php)

```
/** Beregner egenskaber ud fra boolsk algebra ***/  
// Array til mænd  
\$M = ["Jens", "Viktor", "Birk", "Markus"];  
// Array til børn  
\$C = ["Viktor", "Birk", "Markus", "Ida"];  
// Array til venstrehåndede  
\$L = ["Birk", "Markus", "Dorte", "Marie"];  
  
function vis(\$besked) {  
    echo \$besked . "<br/>";  
}  
vis(isFemale("Jens") ? "Sandt" : "Falsk");  
vis(isFemale("Dorte") ? "Sandt" : "Falsk");  
vis("--");  
vis(isBoy("Jens") ? "Sandt" : "Falsk");  
vis(isBoy("Dorte") ? "Sandt" : "Falsk");  
vis(isBoy("Markus") ? "Sandt" : "Falsk");  
vis("--");  
vis(isWomanOrChild("Jens") ? "Sandt" : "Falsk");  
vis(isWomanOrChild("Dorte") ? "Sandt" : "Falsk");  
vis(isWomanOrChild("Markus") ? "Sandt" : "Falsk");  
  
/* Undersøger om name er hunkøn. */  
function isFemale(\$name) {  
    global \$M, \$C, \$L;  
    return !contains(\$M, \$name);  
}  
  
/* Undersøger om name er en dreng. */  
function isBoy(\$name) {  
    global \$M, \$C, \$L;  
    return contains(\$M, \$name) && contains(\$C, \$name);  
}  
  
/* Undersøger om name er kvinde eller barn */  
function isWomanOrChild(\$name) {  
    global \$M, \$C, \$L;  
    return !contains(\$M, \$name) || contains(\$C, \$name);  
}
```

```
/* Undersøger om array indeholder element. */
function contains(\$array, \$element) {
    global \$M, \$C, \$L;
    for (\$i = 0; \$i < count(\$array); \$i++) {
        // Returner sand, hvis dette er elementet.
        if (\$array[\$i] == \$element) return true;
    }
    // Returner falsk, hvis den ikke blev fundet.
    return false;
}
```

Processing (boolske_variable.pde)

```
/** Beregner egenskaber ud fra boolsk algebra ***/  
// Array til mænd  
String[] M = {"Jens", "Viktor", "Birk", "Markus"};  
  
// Array til børn  
String[] C = {"Viktor", "Birk", "Markus", "Ida"};  
  
// Array til venstrehåndede  
String[] L = {"Birk", "Markus", "Dorte", "Marie"};  
void setup() {  
    println(isFemale("Jens"));  
    println(isFemale("Dorte"));  
    println("--");  
    println(isBoy("Jens"));  
    println(isBoy("Dorte"));  
    println(isBoy("Markus"));  
    println("--");  
    println(isWomanOrChild("Jens"));  
    println(isWomanOrChild("Dorte"));  
    println(isWomanOrChild("Markus"));  
}  
  
/* Undersøger om name er hunkøn. */  
boolean isFemale(String name) {  
    return !contains(M, name);  
}  
  
/* Undersøger om name er en dreng. */  
boolean isBoy(String name) {  
    return contains(M, name) && contains(C, name);  
}  
  
/* Undersøger om name er kvinde eller barn */  
boolean isWomanOrChild(String name) {  
    return !contains(M, name) || contains(C, name);  
}  
  
/* Undersøger om array indeholder element. */  
boolean contains(String[] array, String element) {  
    for (int i = 0; i < array.length; i++) {
```

```
// Returner sand, hvis dette er elementet.  
if (array[i].equals(element)) return true;  
}  
// Returner falsk, hvis den ikke blev fundet.  
return false;  
}
```



Opgave: Mængder

1. Kør ovenstående program, og læg mærke til dets output i konsollen.
2. Åbn konsollen i Chrome med Ctrl-Shift-I og find den under koden i Processing.
3. Læs koden, og forsøg at gennemskue, hvordan funktionerne beregner deres resultat.

Sandhedsværdi

En sandhedsværdi er en datatype, som enten kan antage værdien sand eller falsk

Operatorer for sandhedsværdier

For de fleste programmeringssprog er der følgende operatorer for sandhedsværdier. Tabellen viser for hver operator outputtet til to givne input.

Eksempel: *sandt ELLER falsk* evalueres til *sandt*

| Operator | JS, PHP, Processing | Scratch | Input A | Input B | Output |
|----------|---------------------|----------|---------|---------|--------|
| OG | && | og/and | sandt | sandt | sandt |
| | | | sandt | falsk | falsk |
| | | | falsk | falsk | falsk |
| ELLER | | eller/or | sandt | sandt | sandt |
| | | | sandt | falsk | sandt |
| | | | falsk | falsk | falsk |
| IKKE | ! | ikke/not | sandt | | falsk |
| | | | falsk | | sandt |



Opgave: Udvide programmet Mængder

1. Lav en funktion isRightHandedMan, som returnerer sand, hvis og kun hvis personen er en voksen, højrehåndet mand.
2. Lav en funktion isLeftHandedWomanOrChild, som returnerer sand, hvis og kun hvis personen er en venstrehåndet kvinde eller venstrehåndet barn.

Operatorer for tal

Ud over de aritmetiske operatorer nævnt i afsnittet [Tal \(se side 192\)](#) er der operatorer til at sammenligne talværdier, hvor den resulterende datatype er en sandhedsværdi:

- $5 \text{ (tal)} < 8 \text{ (tal)}$ => sandt (sandhedsværdi)
- $5 \text{ (tal)} > 8 \text{ (tal)}$ => falsk (sandhedsværdi)
- $5 \text{ (tal)} = 8 \text{ (tal)}$ => falsk (sandhedsværdi)

Læs mere

På disse officielle sider for sprogene kan du læse mere om boolske værdier.

| Kilde | Beskrivelse |
|--|--|
| Boolean Block - Scratch Wiki | Blokke med boolske værdier og typen |
| MDN Boolean JavaScript | Objektet Boolean i JavaScript |
| PHP.net Booleans | Egenskaber og konvertering til boolean |
| processing.org boolean | Syntaks og beskrivelse for Processing |

Arrays

Tal, strenge og sandhedsværdier er typer af data, som kan anvendes i variable til at indeholde enkelte værdier: Ét tal, én streng, én sandhedsværdi.

I dette afsnit introduceres en *datastruktur* til at indeholde en række værdier. Det er hensigtsmæssigt, når samlinger af data skal behandles i et program: Eksempelvis en liste af målinger til et forsøg eller en række personnavne i et register.

**Opgave: Bruge programmet Stjerner**

1. Kør programmet Stjerner herunder.
2. Nøgleordet for bruges til at gennemløbe datastrukturen *array* for variablerne xs, ys og rads. Se om du kan gennemsuke, hvad programmet gør ved de to forekomster af for.
3. Hvordan bliver variablerne xs, ys og rads erklæret?

Kode: Stjerner

JavaScript (stjerner.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Stjerner</title>
</head>
<body>
<canvas id="himmel"
style="background-color: black">
Din browser understøtter ikke canvas</canvas>

<script type="text/javascript">

/* Viser en nattehimmel fyldt med stjerner */
// Induets størrelse;
var width = 1600; var height = 900;
var canvas = document.getElementById("himmel");
canvas.width = width; canvas.height = height;

// Antal stjerner
var number = Math.round(width/7);

// Koordinater og radius
var xs = new Array(number);
var ys = new Array(number);
var rads = new Array(number);

// Sæt tilfældig data for enhver stjerne.
for (var i = 0; i < xs.length; i++) {
    xs[i] = getRandomIntInclusive(0, width);
    ys[i] = getRandomIntInclusive(0, height);
    rads[i] = getRandomIntInclusive(1, 5);
}

// Tegn stjernerne med hvid farve
var himmel = canvas.getContext("2d");
himmel.fillStyle = "white";
for (var i = 0; i < xs.length; i++) {
    // Tegn en fyldt cirkel med funktionen arc
    himmel.beginPath();
```

```
himmel.arc(xs[i], ys[i], rads[i], 0, 2*Math.PI);
himmel.fill();
}

/* Hjælpefunktion til tilfældige tal */
function getRandomIntInclusive(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

</script>

</body>
</html>
```

Processing (stjerner.pde)

```
**** Viser en nattehimmel fyldt med stjerner ***
// Vinduets størrelse;
int width = 1600; int height = 900;

// Antal stjerner
int number = width/7;

// Koordinater og radius
int[] xs = new int[number];
int[] ys = new int[number];
int[] rads = new int[number];
void setup() {
    // Åbn vinduet
    size(1600, 900);
    background(0, 0, 0);
    // Sæt tilfældig data for enhver stjerne.
    for (int i = 0; i < xs.length; i++) {
        xs[i] = int(random(0, width));
        ys[i] = int(random(0, height));
        rads[i] = int(random(1, 5));
    }
    // Tegn stjernerne
    noStroke();
    fill(255, 255, 255);
    for (int i = 0; i < xs.length; i++) {
        ellipse(xs[i], ys[i], rads[i], rads[i]);
    }
}
```

Kloner i Scratch

Eksemplet med stjerner kan i Scratch med fordel laves med kloner. Så vil der i stedet for tre lister med 100 elementer blive lavet 100 kloner, og hver klon vil indeholde data om koordinater og radius.

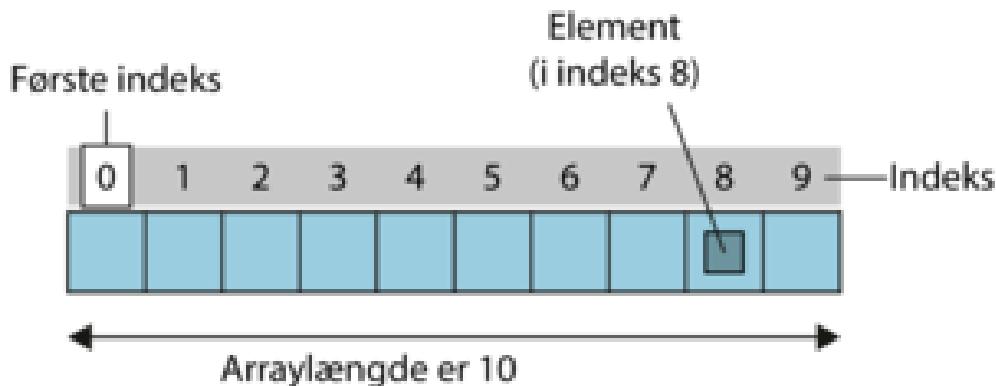
Array

Et array er en samling af elementer af samme type, som kan tilgås med et indeks (heltal) eller anden type nøgle.

Operationer på et array

Operationer på et array kan eksempelvis være:

- Indsæt værdien "Oliver" med indeks 4.
- Hvilket element er i indeks 7?
- Slet elementet med indeks 3.
- Hvad er længden af arrayet?



Det ville være naturligt, at indeks starter med 1. Imidlertid er første indeks 0. Det vil sige, at man i programmering tæller 0, 1, 2, 3 osv.

I Scratch er første indeks 1 og ikke 0.

Forskellige arrays

Arrays implementeres forskelligt i hvert programmeringssprog, og de har forskellige egenskaber og anvendelsesmuligheder. Her er en kort oversigt.

1. Scratch: Array med funktioner til at undersøgelse af medlemsskab, tilføje elementer og slette elementer
2. JavaScript: Array med funktioner til at tilføje og slette elementer
3. PHP: Ordnet symboltabel (engelsk: map), hvor indeks f.eks. kan være strenge, samt funktioner til at tilføje og slette elementer
4. Processing: Array uden mulighed for at ændre længden



Opgave: Udvidelse af Stjerner

1. Sæt antallet af stjerner til 1/10 af bredden i stedet for 1/7, og lav stjernernes radius større.
2. Lav et nyt array, *colours*, som skal indeholde stjernernes farver. Det skal have samme størrelse, som de andre arrays.
3. Sæt farven til en tilfældig farve i det første gennemløb af arrayet.
4. Brug farven i andet gennemløb til at tegne cirklen med den givne farve.



Opgave i Scratch: Mængder

Afsnittet Sandhedsværdier indeholdt et eksempel på anvendelse af arrays i programmet Mængder. Lav med udgangspunkt i [koden i JavaScript og Processing \(se side 216\)](#) det tilsvarende program i Scratch. Anvend blokken "indeholder" til at undersøge om en liste indeholder en given streng.



Opgave i PHP: Arrays og databaser

1. Læs koden fra eksemplet om [interaktion med databaser \(se side 293\)](#) med fokus på løkken.
2. Hvilken variabel indeholder et array?
3. Hvor mange elementer har dette array?
4. Hvad er specielt ved denne type array?

På disse sider kan du lære mere om arrays og HTML canvas, som bruges i JavaScript-programmet.

| Kilde | Beskrivelse |
|--|---|
| W3C HTML Canvas 2D | Specifikation af metoder til at tegne på websider |
| List - Scratch Wiki | Egenskaber og eksempler for lister i Scratch |
| MDN Array i JavaScript | Eksempler på oprettelse og anvendelse af array |
| php.net arrays | Forklaring på anderledes array (map) i PHP |
| processing.org Array | Eksempler og beskrivelse af array i Processing |

Datetyper

OBS

Eleverne skal kunne analysere typer og data og udvælge og anvende typer af data på b-niveauet. Dette får de mulighed for i dette kapitel.

I de forrige afsnit har du lært, hvordan variable kan have datatypen tal, streng eller sandhedsværdi.

Ikke alle programmeringssprog gør det muligt at angive datatypen ved definitionen af en variabel. Hvis sproget ikke understøtter det, vil datatypen automatisk blive angivet. Således vil der altid – eksplisit eller implicit – være knyttet en datatype til en variabel eller konstant i et program.

Der er flere fordele ved at knytte en datatype til data. For det første kan man lave operationer på data, som afhænger af datatypen. Eksempelvis giver det mening at lægge to heltal sammen med operatoren plus, men hvad skal der ske, hvis man "lægger" sandt og falsk sammen eller "trækker" ordet "Hej" fra ordet "Viktor"?

En anden fordel er optimering af den plads, som en computer bruger på at opbevare værdien af en variabel. En sandhedsværdi som sandt eller falsk kan gemmes som en enkelt bit, som enten er 0 eller 1. Derimod fylder et tal eller et tegn mere i hukommelsen.

Datatype

En datatype er en kategorisering af data, som fortæller hvilken type af data en variabel kan have.



Opgave: Fordeler ved datatyper

Forklar fordelene ved at have datatyper i et programmeringssprog.

Klasser af datatyper

Datatyper deles op i forskellige klasser. På laveste niveau findes *primitive* datatyper, som er indbyggede, grundlæggende datatyper i et sprog. Ud fra primitive datatyper kan *sammensatte* datatyper konstrueres som *datastrukturer*.

Oversigt: Datatyper

Tabellen herunder giver et overblik over udvalgte datatyper.

| Datatype | Klasse | Beskrivelse |
|---------------|-----------|--------------------------------------|
| Tal | Primitiv | Eksempelvis heltal eller kommatal |
| Sandhedsværdi | Primitiv | Kan antage værdien sandt eller falsk |
| Streng | Sammensat | Streng af tegn |
| Array | Sammensat | Samling af elementer med indeks |



Opgave: Vælg datatyper

Vælg en passende datatype for hver af følgende data:

- En persons højde
- En virksomheds overskud
- Et bynavn
- En persons køn
- Registrering om en fodboldkamp er spillet
- En bluses farve



Opgave: Datatyper i persondata

Skriv et program, som definerer fem variable for persondata med minimum tre forskellige datatyper. Programmet skal derefter tildele værdier til disse variable.



Opgave: Sammenligning af datatyper

Sammenlign pladsbehovet for datatyperne heltal, kommatal, sandhedsværdi og tekststreng, og sortér dem i ordnet rækkefølge.

Konstanter

En variabels værdi kan, som navnet antyder, variere over tid, hvilket eksempelvis er hensigtsmæssigt, hvis værdien af en variabel repræsenterer en sum, som løbende bliver opdateret og derfor varierer.

I dette afsnit skal vi se på en værdi, som ikke kan variere, og derfor benævnes den en konstant. Et eksempel er moms som i Danmark er 25%.

Fordelen ved at bruge konstanter frem for blot at skrive værdien direkte er, at konstantens navn så kan bruges flere steder i et program, og konstantens værdi kan rettes af programmøren ét sted, hvis programmet skal laves om.

Ikke alle programmeringssprog understøtter konstanter. Hvis sproget understøtter det, og en konstant forsøges at blive ændret, vil der opstå en fejl. Hvis konstanter ikke understøttes i et sprog, må man bruge en variabel og selv sørge for, at dennes værdi ikke ændres under udførelse af programmet.

Det er normalt at skrive konstanternes navne med store bogstaver og variables navne med små bogstaver, så læseren af et program kan se forskel.

Konstant

En konstant er et navn og et stykke lagerplads, som kan indeholde en værdi, der i modsætning til en variable ikke kan ændres under programkørsel.

Eksempel: Dimensioner af billede



iStockphoto.com/borojoint

I nedenstående eksempel anvendes en konstant til at indeholde værdien af et billedes bredde, mens en variabel indeholder værdien af billedets højde. Derefter beregnes henholdsvis areal og omkreds for billedet.

Programmet udskriver følgende på skærmen:

Areal: 24

Omkreds: 22

Areal: 40

Omkreds: 26



Opgave: Konstanter

Find fem eksempler på talværdier der bør erklæres som konstanter.

Kode: Billede

JavaScript (billede.html)

```
const BREDDE = 8; // Billedets konstante bredde
/* Visning af areal */
function udskrivAreal(hoejde) {
    var areal = BREDDE * hoejde;
    document.write("Areal: " + areal + "<br/>");
}
/* Visning af omkreds */
function udskrivOmkreds(hoejde) {
    var omkreds = 2 * (BREDDE + hoejde);
    document.write("Omkreds: " + omkreds + "<br/>");
}
var hoejde = 3; // Højden på billede 1
udskrivAreal(hoejde); udskrivOmkreds(hoejde);
hoejde = 5; // Højden på billede 2
udskrivAreal(hoejde); udskrivOmkreds(hoejde);
```

PHP (billede.php)

```
/define("BREDDE", 8); // Billedets konstante bredde
/* Visning af areal */
function udskrivAreal($hoejde) {
    $areal = BREDDE * $hoejde;
    echo "Areal: " . $areal . "<br/>";
}
/* Visning af omkreds */
function udskrivOmkreds($hoejde) {
    $omkreds = 2 * (BREDDE + $hoejde);
    echo "Omkreds: " . $omkreds . "<br/>";
}
$hoejde = 3; // Højden på billede 1
udskrivAreal($hoejde); udskrivOmkreds($hoejde);
$hoejde = 5; // Højden på billede 2
udskrivAreal($hoejde); udskrivOmkreds($hoejde);
```

Processing (billede.pde)

```
/** Beregner areal og omkreds af billede ***/
final int BREDDE = 8; // Billedets bredde
final int XPOS = 10; // Venstre margin
int ypos;
void setup() {
    size(480, 480);
    ypos = 50;
    PFont f;
    f = createFont("Arial",16,true);
    textAlign(f,36);
    fill(0);
    int hoejde = 3; // Højde på billede 1
    udskrivAreal(hoejde); udskrivOmkreds(hoejde);
    hoejde = 5; // Højde på billede 1
    udskrivAreal(hoejde); udskrivOmkreds(hoejde);
}
/* Visning af areal */
void udskrivAreal(int hoejde) {
    int areal = BREDDE * hoejde;
    text("Areal: " + areal, XPOS, ypos);
    ypos += 50;
}
/* Visning af omkreds */
void udskrivOmkreds(int hoejde) {
    int omkreds = 2 * (BREDDE + hoejde);
    text("Omkreds: " + omkreds, XPOS, ypos);
    ypos += 50;
}
```



Opgave: Forklaring af konstant

1. Kør programmet.
2. Forklar hvor konstanten bliver brugt i programkoden.
3. Forklar fordelene ved at bruge en konstant frem for en variabel.



Opgave: Areal af en cirkel

Skriv et program, som beregner arealet af en cirkel ved at anvende en konstant for pi med værdien 3,14159.

På disse sider kan du lære mere om konstanter. I Scratch er der ikke mulighed for at erklære en konstant.

| Kilde | Beskrivelse |
|--|-------------------------|
| MDN const - JavaScript | Konstanter i JavaScript |
| php.net Constants | Konstanter i PHP |
| processing.org final | Konstanter i Processing |

Kommentarer i koden

I programmeringssprog er det muligt at indsætte kommentarer, som ignoreres af computeren, når programmet udføres. Dette er hensigtsmæssigt af flere grunde.

Oftest arbejder flere mennesker sammen om at udvikle et program. Derfor er det vigtigt, at man supplerer sin kode med kommentarer, så andre forstår hensigten med koden. Det er ikke altid nemt at gennemskue et program, hvis man ikke selv har skrevet det.

Derudover kan det hjælpe ens egen forståelse af programmet, hvis der går måneder inden man skal videreudvikle sit program. Det sker ofte, at man ikke kan huske tankerne bag den skrevne kode.

I kommentarer kan man blandt lave beskrivelser af variable og funktioner samt skrive kommentarer henvendt til andre programmører. Derudover kan man omkransse kode af kommentartegn, så det midlertidigt ignoreres af computeren.

Kode: Kommentarer til clickCanvas (JavaScript)

```
/* Opretter en checkbox og tegner en trekant */
var clickCanvas = function(event) {
    // Hent HTML-formularen, som indeholder checkboxe.
    var form = document.getElementById('triangleForm');
    var newTriangle = document.createElement('input');
    newTriangle.type = 'checkbox';
    newTriangle.name = 'triangles';
    // Placér checkboxene under hinanden.
    newTriangle.style = 'display: block';
    var x = event.clientX; var y = event.clientY;
    // Sæt koordinaterne sammen til en streng.
    newTriangle.value = [x, y];
    newTriangle.checked = true;
    //newTriangle.checked = false;
    newTriangle.addEventListener('change',
        triangleChecked, false);
    // Tilføj checkboxen til formularen for visning.
    form.appendChild(newTriangle);
    drawTriangle(newTriangle);
}
```

Blok- og linjekommentarer

I koden er der to typer af kommentarer. Først ses en kommentar omkranset af /* og */. Disse kommentarer kan fylde flere linjer og kaldes *blokkommentarer*.

Derefter ses en række kommentarer, som startes med //. Disse kan kun fylde én linje og kaldes *linjekommentarer*.

Anvendelser af kommentarer

Kommentarer i koden har flere anvendelser.

| Anvendelse | Placering | Beskrivelse |
|-----------------------|--------------|---------------------------------------|
| Metadata | Øverst i fil | Versioner, datoer, forfattere, formål |
| Planlægning og review | Før/i linjer | Pseudokode |
| Forklaringer til kode | Før/i linjer | Programmørens hensigt |
| Fejlfinding | Før/i linjer | Kode kommenteres ud. |

Hvornår indsættes kommentarer?

- Kommentarer som metadata, funktionsbeskrivelser samt planlægning og review bør skrives, inden koden skrives. Andre kommentarer skrives, når koden skrives eller bagefter.

Hvor meget af koden skal kommenteres?

- Det er en balancegang og afhænger af læseren. Erfarne programmører har ikke behov for at få kommenteret nær så meget som nybegyndere.

Hvad skal ikke kommenteres?

- Åbenlys og simpel kode bør ikke kommenteres, hvis det forklarer sig selv. Generelt bør koden laves simplere, frem for at forklare kompliceret kode.



Opgave: Anvendelse af kommentarer

1. Læs kommentarerne i koden herover.
2. Afgør for hver kommentar, hvilken anvendelse der finder sted i henhold til oversigten herover.

Kommentarer i Scratch, PHP og Processing

Kommentarer i Scratch vedhæftes blokke som gule sedler. De indsættes ved at højreklikke på en blok og vælge Tilføj kommentar.

Kommentarer i PHP og Processing følger samme syntaks som i JavaScript.

I denne oversigt kan du se, hvordan man skriver kommentarer.

| Kilde | Beskrivelse |
|---|--|
| Scratch Wiki: Comment | Kommentarer i Scratch |
| MDN: JavaScript basics | Kommentarer i JavaScript |
| PHP.net: Comments | Kommentarer i PHP |
| processing.org: comment | Linjekommentarer i Processing |
| processing.org: multiline | Blokkommentarer i Processing |
| processing.org: doc | Kommentarer til dokumentation i Processing |

Opgaver til programmering



Opgave: Glædelig jul og godt nytår

1. Skriv et program, som beder brugeren om at indtaste navnet på indeværende måned. Hvis den indtastede måned er "December", skal programmet udskrive "Glædelig jul". Uanset den indtastede måned, skal programmet slutte af med at skrive "Godt nytår".
2. Udvid programmet, så det i stedet for "Glædelig jul" udskriver "Bare det snart var jul", hvis den indtastede måned ikke er "December".



Opgave: Skudår

Ifølge Wikipedia gælder der følgende om skudår:

I den gregorianske kalender er et år skudår, hvis årstallet er deleligt med 4. Dette gælder dog ikke årstal delelige med 100, bortset fra dem, der er delelige med 400. År 1900 var således ikke et skudår, men år 2000 var.

Du skal nu lave et program, som kan afgøre om et år er skudår.

1. Hvilke årstal er det nødvendigt at teste programmet med?
2. Lav et program som ud fra et indtastet årstal kan afgøre, om det er et skudår.
3. Udvid programmet med en funktion, som ud fra et årstal som input afgør om det er et skudår.



Opgave: Investering med fast rente

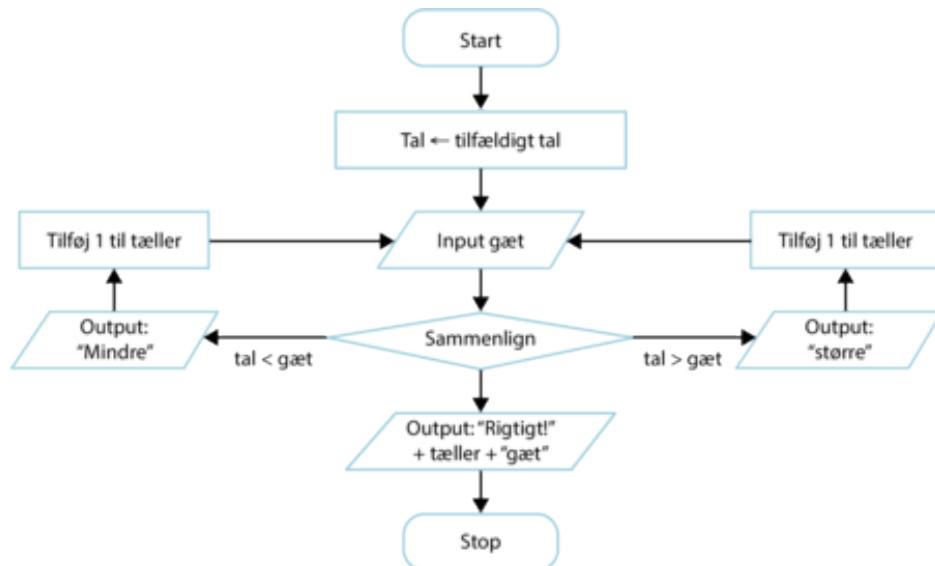
Du ønsker at investere 10.000 kr. til en fast rente på 2% per år.

Lav et program som anvender en løkke til at udskrive værdien af din investering for hvert år de første 10 år. Hint: Investeringen skal hvert år ganges med 1,02.



Opgave: Gæt et tal mellem 1 og 100

Lav et spil, hvor brugeren skal gætte et bestemt tal mellem 1 og 100 med færrest mulige gæt. Det skal fungere, så brugeren efter hvert gæt får at vide, om gættet er højere eller lavere end tallet. Programmet skal først stoppe, når brugeren har gættet tallet.



Opgave: Karaktergennemsnit

1. Lav et program, som gentagende beder brugeren om at indtaste en karakter og trykke på enter. Når brugeren i stedet for en karakter indtaster "Slut", skal programmet vise gennemsnittet af de indtastede karakterer.
2. Udvid programmet, så det ved hvert input kontrollerer om den indtastede værdi er en gyldig karakter (-3, 0, 2, 4, 7, 10 eller 12).



Opgave: Byttepenge

Når en kassedame modtager sedler som betaling, skal hun beregne, hvilke byttepenge skal gives retur.

Hvis kunden for eksempel skal betale kr. 260,50 og betaler med kr. 500, skal hun finde ud af, hvilke sedler og mønter kunden skal have retur. I dette tilfælde kunne det være:

$$200 + 20 + 10 + 5 + 2 + 2 + 0,50$$

De mulige sedler og mønter er 1.000, 500, 200, 100, 50, 20, 10, 5, 2, 1 og 0,50.

1. Lav et program, som ud fra de to beløb som parametre kan oplyse om byttepenge.
2. Er det altid entydigt, hvilke byttepenge kunden skal have retur?



Opgave: Fakultet

Fakultet er en funktion i matematik, som givet et tal *input* returnerer produktet af de positive hele tal fra 1 til og med *input*. Det skrives i matematik med et udråbststegn efter tallet.

Eksempel:

$$6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$$

Lav funktionen *fakultet*, som givet parameteren *input* returnerer *input!*.



Opgave: Anvendelse af trekantsberegner

1. Anvend [programmet](#).
2. Find formlerne for retvinklede trekantre i din formelsamling.
3. Regner programmet rigtigt?
4. Hvilke operatorer (+, -, *, /) og funktioner (sin, cos, tan) er nødvendige for at lave beregningen?

Kode: Trekantsberegner

JavaScript (trekanter.html)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Trekanter</title>
</head>
<body>
    <h1>Retvinklet trekant</h1>
    <p>Hypotenusen: <input type="text" id="hyp"
        autofocus/></p>
    <p>En spids vinkel:
        <input type="text" id="ang"/></p>
    <p><input type="button" id="button"
        value="Beregn"></p>
    <p>Modstående side til vinklen:
        <span id="opp"></span></p>
    <script type="text/javascript">

        /* Beregner modstående ud fra hypo og vinkel */
        function oppFromHyp(hyp, ang) {
            // Omregn fra radianer til grader ved Math.sin
            var angDeg = ang * Math.PI / 180;
            // Beregn modstående side ud fra formel.
            var opposite = hyp * Math.sin(angDeg);
            return opposite;
        }

        /* Ved klik skal argumenterne gives til funk. */
        var clickEvent = function( event ) {
            var hyp = document.getElementById('hyp').value;
            var ang = document.getElementById('ang').value;
            document.getElementById('opp').innerHTML =
                oppFromHyp(hyp, ang);
        }

        // Knyt hændelsesfunktionen til tryk på knappen
        document.getElementById("button").addEventListener("click", clickEvent, false);
    </script>
```

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

PHP (trekanter.php)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Trekanter</title>
</head>
<body>
<h1>Retvinklet trekant</h1>
<form action="trekanter.php" method="get">
<p>Hypotenusen: <input type="text" name="hyp"
   autofocus/></p>
<p>En spids vinkel:
   <input type="text" name="ang"/></p>
<p><input type="submit" value="Beregn"></p>
</form>

<?php
/* Beregner modstående ud fra hypo og vinkel */
function oppFromHyp(\$hyp, \$ang) {
    // Omregn fra radianer til grader ved Math.sin
    \$angDeg = \$ang * pi() / 180;
    // Beregn modstående side ud fra formel.
    \$opposite = \$hyp * sin(\$angDeg);
    return \$opposite;
}

/* Ved indtastede værdier beregnes siden. */
\$hypEntered = !empty($_GET['hyp']);
\$angEntered = !empty($_GET['ang']);
if (\$hypEntered && \$angEntered) {
    \$hyp = $_GET['hyp'];
    \$ang = $_GET['ang'];
    echo "<ul><li>Hypotenusen: \$hyp</li>";
    echo "<li>En spids vinkel: \$ang</li>";
    echo "<li>Modstående side til vinklen: ";
    echo oppFromHyp(\$hyp, \$ang) . "</li></ul>";
}
?>
</body>
</html>
```



Opgave: Udvidelse af trekantsberegner

1. Ret i programmet, så det kontrollerer, at input i tekstfelterne er matematisk gyldig ($h > 0$ og $0^\circ < v < 90^\circ$).
2. Udvid programmet, så det også kan beregne hypotenusen ud fra modstående side og en vinkel.



Opgave: Pythagoras

Lav et program, som bruger Pythagoras' læresætning til at beregne en katete a ud fra en anden katete b og en hypotenuse c .

Hjælp til opgaven:

1. Kopier filen fra trekantsberegneren, og tag udgangspunkt i denne.
2. Ret tekster på skærmen, så de passer til det nye program.
3. Lav en funktion `pythSide` med parametrene `side` og `hypotenuse`.
4. Brug formlen $a = \sqrt{c^2 - b^2}$ til at færdiggøre funktionen.



Opgave: Navnebytte

Lav et program, som bytter om på to variables værdier for personnavne, som er indtastet af brugeren.

Hint: Det er nødvendigt med tre variable.



Opgave: Palindrom

Et *palindrom* er et ord (eller en sætning), som kan læses ens fra venstre og højre, for eksempel kajak.

1. Diskutér hvordan et program skal kunne afgøre, om et ord er palindrom, idet du tager udgangspunkt i et par eksempler.
2. Lav en funktion `isPalindrome`, der får en streng som parameter og returnerer en sandhedsværdi.
3. Test funktionen med palindromer og ikke-palindromer.



Opgave: Søg efter tegn

Lav en funktion som ud fra to parametre *haystack* (strenge) og *needle* (strenge) returnerer sandt, hvis *needle* er en delstrenge af *haystack*, og ellers returnerer falsk.



Opgave: Søge i et array

Lav en funktion som ud fra de to parametre *list* (array) og *element* (tal) returnerer sandt, hvis *element* ligger i *list*, og ellers returnerer falsk.



Opgave: Sortering af array

Lav en funktion, som ud fra parametren *list* (array af tal) returnerer et array af tal, som er elementerne fra *list* i sorteret rækkefølge.

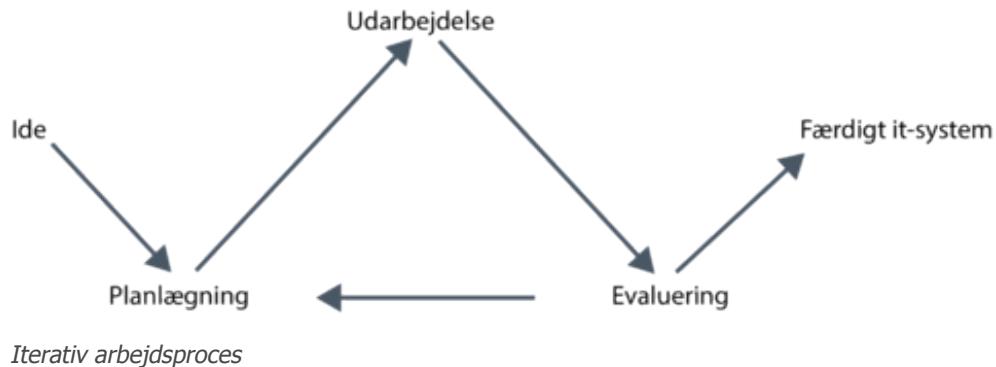


Opgave: Liste af tegn

Lav en funktion *opdel*, som givet strengen *ord* som parameter returnerer en liste med tegnene i *ord*.

4. Evaluering af et it-system

Som du så i kapitel 1, skal man efter planlægning og udarbejdelse evaluere sit it-system. Disse tre faser skal man måske igennem flere gange. Derfor evaluerer man både løbende og efter, at man mener, man er færdig med sit it-system.



Følgende trin gennemgås ofte:

- test
- brugervejledning
- dokumentation

Det er nu, man har glæde af den løbende dokumentation af arbejdet med at planlægge og udarbejde it-systemet. Denne dokumentation gør, at ovenstående trin hurtigere er skrevet og med et bedre indhold i forhold til, hvis man skriver det hele til sidst.

Innovativt?

I tillæg til at teste det it-system, som man har udviklet, kan man lave en kvalificeret vurdering af, om it-systemet er [innovativt](#).

Test



iStockphoto.com/LisLud

Når man evaluerer, undersøger man, om [interaktionsdesignet](#) er blevet som forventet, og om [kravspecifikationerne](#) er overholdt. Er der en kunde involveret, deltager denne i relevante dele af evalueringen.

Undersøgelsen af interaktionsdesignet og kravspecifikationerne foregår ved, at man laver en test. Testen skal forberedes og udføres, og bagefter skal den dokumenteres.

Der er mange måder at tilrettelægge en test på. Man kan bede den, der tester, om at tænke højt og fortælle, hvordan han oplever det at bruge it-systemet. Man kan også filme en testperson for at se, hvordan han bruger it-systemet. Endelig kan man lade testpersonen beskrive skriftligt, hvordan han oplever det at bruge it-systemet.



Opgave: Undersøgelse af interaktionsdesign

Find et it-system på nettet med et dårligt interaktionsdesign.

Hvorfor er det dårligt?

Hvad kan laves om, så det bliver bedre?



Opgave: Sammenhæng mellem kravspecifikation og it-system

Find et it-system på nettet, hvor kravspecifikationen indeholder krav, som it-systemet ikke kan leve op til.

Tænke-højt-test

Tests er kernestof på niveau B. Tænke-højt-test er et eksempel.

Tænke-højt-test

Når man tester, kan man teste brugervenlighed, men man kan også teste funktionaliteten i det it-system, man er ved at lave. Vi vil her have fokus på et lidt bred testtype, hvor der ikke nødvendigvis er et særligt fokus. Vi vil se på tænke-højt-test.

Ved en tænke-højt-test sætter man brugeren til at anvende it-systemet og fortælle højt om den umiddelbare opfattelse af it-systemet. I denne proces opnår man bl.a. overblik over:

- de funktionaliteter, det er sværest for brugeren at anvende, og de der er problemfrie.
- hvordan brugeren opfatter menupunkter og ikoner.
- hvordan brugeren gennemløber en brugssekvens i it-systemet.
- brugerens reaktionsmønster, når han ikke kan finde ud af dele af it-systemet.

Denne testtype kræver, at man er ret langt i designet af it-systemet (fx i form af papirprototyper), så der reelt er et it-system af afprøve. Hvis man gerne vil teste tidligere i designforløbet, er det bedre at lave f.eks. fokusgruppeinterview eller spørgeskemaer.

Forberedelse, gennemførelse og afrapportering

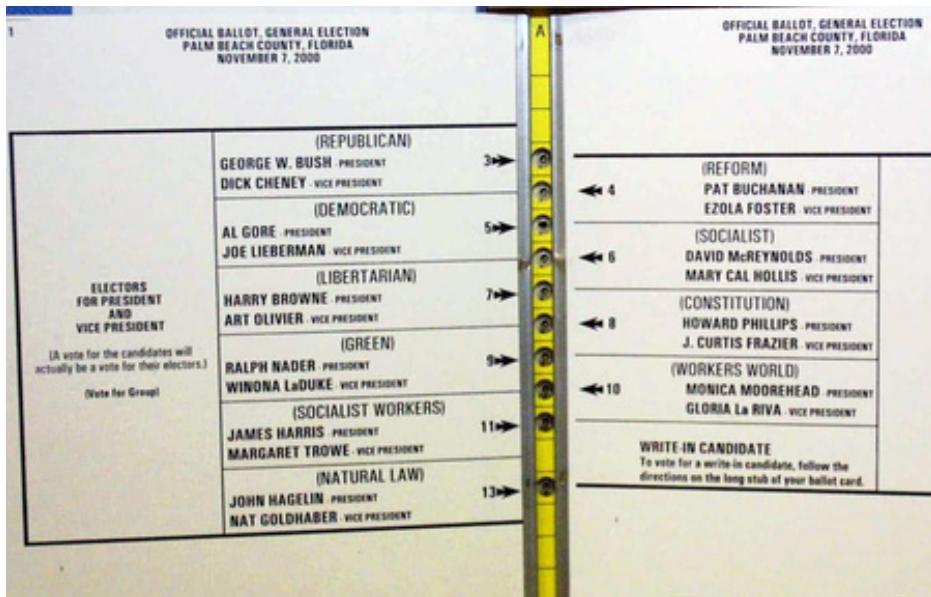
Når man forbereder en tænke-højtest, kan man lave en række opgaver, som brugeren skal løse i it-systemet. Dette sikrer at man kommer gennem alle de dele af it-systemet, man er interesseret i at teste. Hvis man vælger en friere tilgang, hvor brugeren selv lægger vejen gennem testen af it-systemet, kan man få viden om, hvilke dele af it-systemet brugeren synes ser særligt interessant, ud og hvilke dele han undgår.

Man skal være opmærksom på, at brugeren kan synes, at testsituationen er underlig. Derfor er det en god idé at starte med en indledning, hvor man for eksempel siger til brugeren: "Jeg vil gerne have, at du tænker højt og fortæller om, hvad du gør, hvorfor du gør det, og hvad du synes om it-systemet".

Under gennemførelsen han brugerens udsagn skrives ned eller optages som lyd eller på video. Går brugeren i stå i fortællingen under testen, kan man spørge ind til det, der foregår. Man kan overveje at lade det være i orden, at brugeren stiller spørgsmål under testen. En anden mulighed er at lade to brugere teste. Fordelen her er, at det kan give en god dialog om it-systemet.

Ved afrapportering er det vigtigt, at man forholder sig neutral i forhold til det, brugeren har sagt om it-systemet. Man kan overveje efter testen at lade brugeren fortælle om sin oplevelse mere generelt. Dermed får man brugerens helhedsvurdering af it-systemet, og her kan komme punkter frem, som brugeren ikke fik sagt under testen.

Eksempel: her burde have været testet



Ved det amerikanske valg i 2000 så stemmesedlen i Palm Beach som ovenfor.

Efter valget påstod en del, at de havde villet stemme på Al Gore, men at de i stedet var kommet til at sætte krydset ved Pat Buchanan. Fejlen skete, fordi Al Gore var nummer 2 på sedlen, men man skulle krydse af tre punkter nede, hvis man ville stemme på ham.

Konsekvensen blev, at alle stemmesedlerne blev kasseret.



øvelse: e-Boks

Tilrettelæg, udfør og evaluér en tænke-højt-test af [e-Boks](#).

Hvad er din konklusion?



øvelse: DSB

Tilrettelæg, udfør og evaluér en tænke-højt-test af [DSB](#).

Hvor ser du særligt muligheder for forbedring?

Lav et forbedret design.

Brugervejledning



iStockphoto.com/vectorplusb

Når man mener, at man er færdig med at programmere et it-system, vil man udover ovenstående dele til evalueringen, også lave en brugervejledning og dokumentere it-sytemet.

Brugervejledningen indeholder en beskrivelse af

- hensigten med it-systemet
- hvordan it-systemet bruges

En brugervejledning behøver ikke være en skrevet beskrivelse. Det kan også være en video eller lignende.



Opgave: Brugervejledning

Vælg et it-system, og lav en brugervejledning. Du kan f.eks. vælge en app fra din smartphone eller et it-system på din pc.

Få en anden til at læse din brugervejledning og kommentere denne i forhold til det it-system, du valgte.



Opgave: Film som brugervejledning

Find på nettet en film, som er en brugervejledning.

Vurdér, om den er en god brugervejledning – og hvorfor.

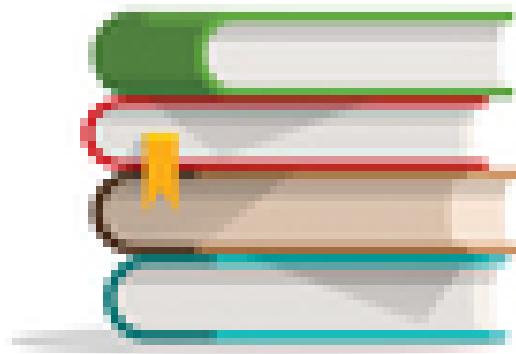


Opgave: Brugervejledning og målgrupper

I hvilke situationer foretrækker du en skrevet brugervejledning, og i hvilke situationer foretrækker du en film som brugervejledning?

Tror du, at der er forskel på, hvilke [målgrupper](#) der foretrækker film frem for tekst som brugervejledning?

Dokumentation



iStockphoto.com/vladwel

Dokumentationen af et it-system afhænger af typen af it-system, og i hvilken sammenhæng it-systemet er udviklet. Typisk vil mange af de elementer, der har været brugt i planlægningen og udarbejdelsen af et it-system, indgå i dokumentationen. Det kan være f.eks. beskrivelse af målgrupper, databaser og algoritmer.

Desuden kan dokumentationen indeholde

- kravspecifikation
- tests og testresultater
- tekniske oplysninger om installation og software-/hardwarekrav

Er der tale om et it-system til styring af automatpiloten til et fly, er dokumentationen langt mere detaljeret end et it-system som f.eks. ved spillet Tetris. Er der tale om et it-system udviklet i undervisningen, er dokumentationen en rapport.



Opgave: Lav en dokumentation

Lav en lille del af dokumentationen til denne iBog.

Din dokumentation skal indeholde

- tekniske oplysninger om installation og software/hardware-krav
- målgruppe

5. Andet materiale



iStockphoto.com/IconicBestiary

Når man udvikler, programmerer og evaluerer et it-system, arbejder man med forskellige former for relevant materiale.

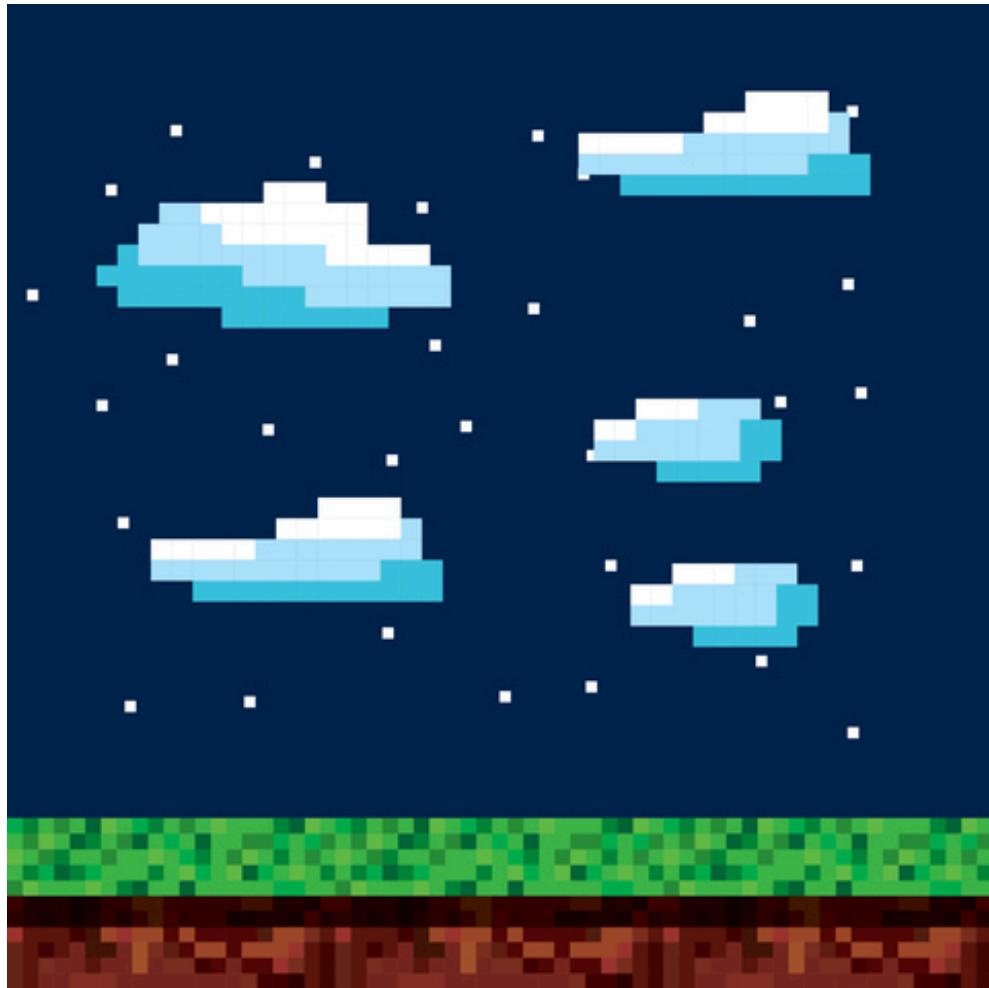
Udover dette materiale, er der andre emner, der er relevante, når man beskæftiger sig med it. I dette afsnit kan du læse om nogle af disse emner.

Billeder og farver

OBS

Dette er kernestof på B-niveau.

Når vi arbejder med billeder, arbejder vi med data. Vi arbejder bl.a. med farver, komprimering og farvernes betydning.



iStockphoto.com/johavel



Opgave: Billedformater

Gå på internettet, og find de forskellige filtyper til billeder. Hvornår bruger man de forskellige typer?

Bitmap-billeder

Et såkaldt bitmap-billede er opbygget af pixels (forkortelse for picture element), der er en inddeling af billedefloden i tern.

Størrelsen på denne inddeling skrives med antal pixels i bredden x antal pixels i højden. Jo flere tern, jo højere oplosning.

Hver pixel har en farve. Antallet af mulige farver kaldes farvedybden.

Her er nogle eksempler på forskellige oplosninger og forskellige farvedybder:

300 pixels



300 pixels i bredden og 30 farver.



300 pixels i bredden og 20 farver.



300 pixels i bredden og 10 farver.

200 pixels



200 pixels i bredden og 30 farver.



200 pixels i bredden og 20 farver.



200 pixels i bredden og 10 farver.

100 pixels



100 pixels i bredden
og 30 farver.

100 pixels i bredden
og 20 farver.

100 pixels i bredden
og 10 farver.

iStockphoto.com/Fnadya76 og Systime

Jo højere opløsning og jo mere farvedybtde, jo bedre billede, men også jo større fil.

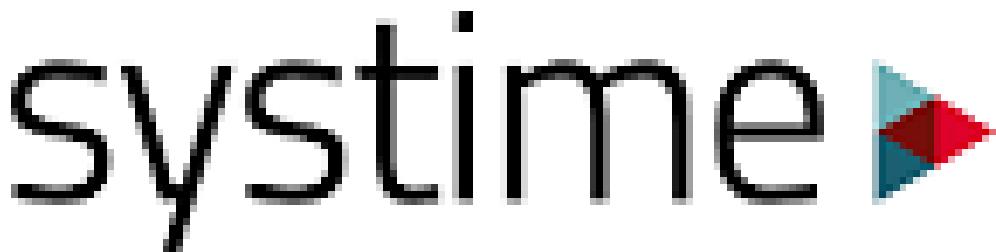


Opgave: Skærmopløsning

En computerskærm kan for eksempel have opløsningen 640 x 480, 800 x 600 eller 1280 x 1024 pixels.

Undersøg, hvad skærmopløsningen er på din pc og på din mobiltelefon.

Ved bitmap-billeder opstår der et problem hvis man ønsker at skalere for eksempel et logo. Hvis man skalerer et bitmap-billede op, bliver det grynet, fordi den enkelte pixel fysisk bliver større.



Dette problem kan løses ved i stedet at anvende vektorgrafik.

Vektorgrafik

Vektorgrafik

Når man anvender vektorgrafik, beskrives delene på et billede ved matematiske ligninger, der beskriver figurer som cirkler, kegler og kuber. For eksempel består billedet af de olympiske ringe af ligningerne for fem cirkler, deres bredde i forhold til radius, deres placering og information om de fem farver.



Hvis man skalerer dette billede op, bliver det ikke grynet. Når man skalerer billeder i vektorgrafik, laver man matematiske transformationer på de matematiske figurer.



Opgave: Filformater til vektorgrafik

Hvilke filformater er vektorgrafik-formater?

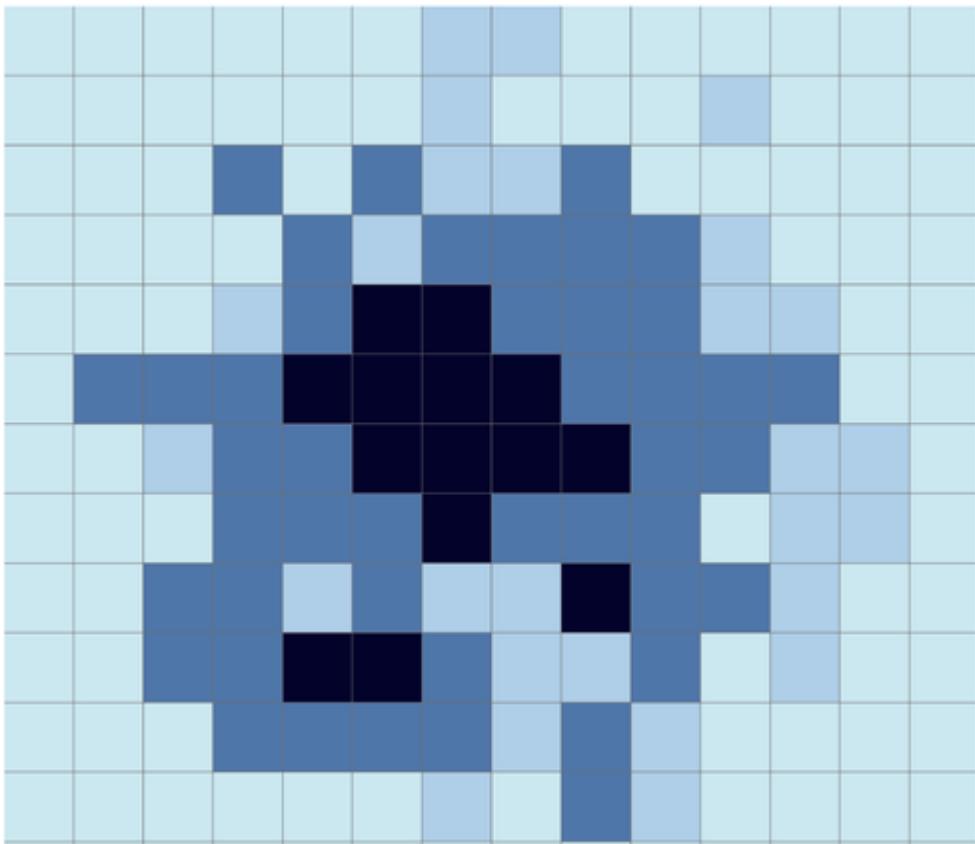
Komprimering

Gode og detaljerede billeder er store datafiler. Dette kan være et problem, hvis de for eksempel skal være på en hjemmeside. Den tid, som det tager at uploadet, kan blive urimelig stor.

Generelt kan man komprimere filer – altså gøre filstørrelsen mindre – på to måder: med og uden tab. Komprimering uden tab betyder, at man kan genskabe den oprindelige fil, hvis man ønsker det. Komprimering med tab betyder, at man for at gøre filstørrelsen mindre ved at fjerne data fra filen på en måde, så filen ikke kan genskabes. Lad os se to eksempler på komprimering af et bitmap-billede.

Tabsfri komprimering

Et bitmap-billede består typisk af store områder, hvor pixels har samme farve. På følgende billede med blomsten ses, at næsten hele den øverste række pixels har samme farve.



iStockphoto.com/Fnadya76 og Systime

Konkret er der fire farver på billedet, lad os kalde dem 1, 2, 3, 4 med den lyseste farve som nr. 1. Hvis vi vil repræsentere det ved at beskrive det fra øverste venstre hjørne, kunne en repræsentation af billedet se sådan ud:

111111221111111111211121111131322311111111323333211111

1234433221113334444333311123344443322111333433312211

133443223121111333323211111111213211111111111111111111

Det fylder 168 tegn.

Hvis vi repræsenterer billedet ved at skrive noget om, hvor mange der er i træk af samme farve, kunne det se sådan ud:

16221621132113133111312231151431113122311513213142332212

113344341212213244322211333413311221122211211241322112

123242312231112112133421312114162111312114

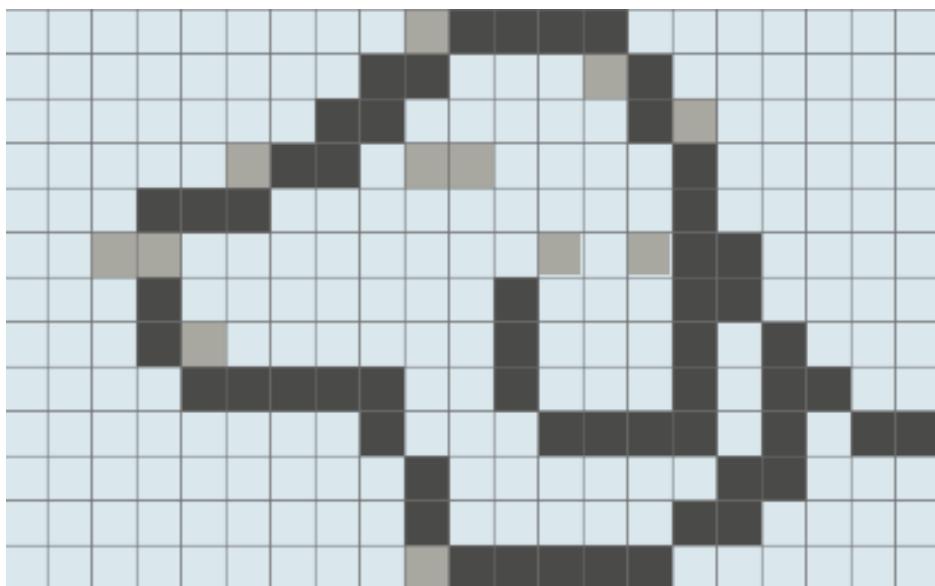
Nu fylder det 154 tegn, og dermed fylder det komprimerede billede mindre end det originale billede. Læg mærke til, at komprimeringen er tabsfri, og vi kan genskabe originalbilledet, hvis vi vil.



Opgave: Tabsfri komprimering

Opskriv en repræsentation som ovenfor for dette billede af hovedet af en hund, og opskriv en repræsentation med tabsfri komprimering.

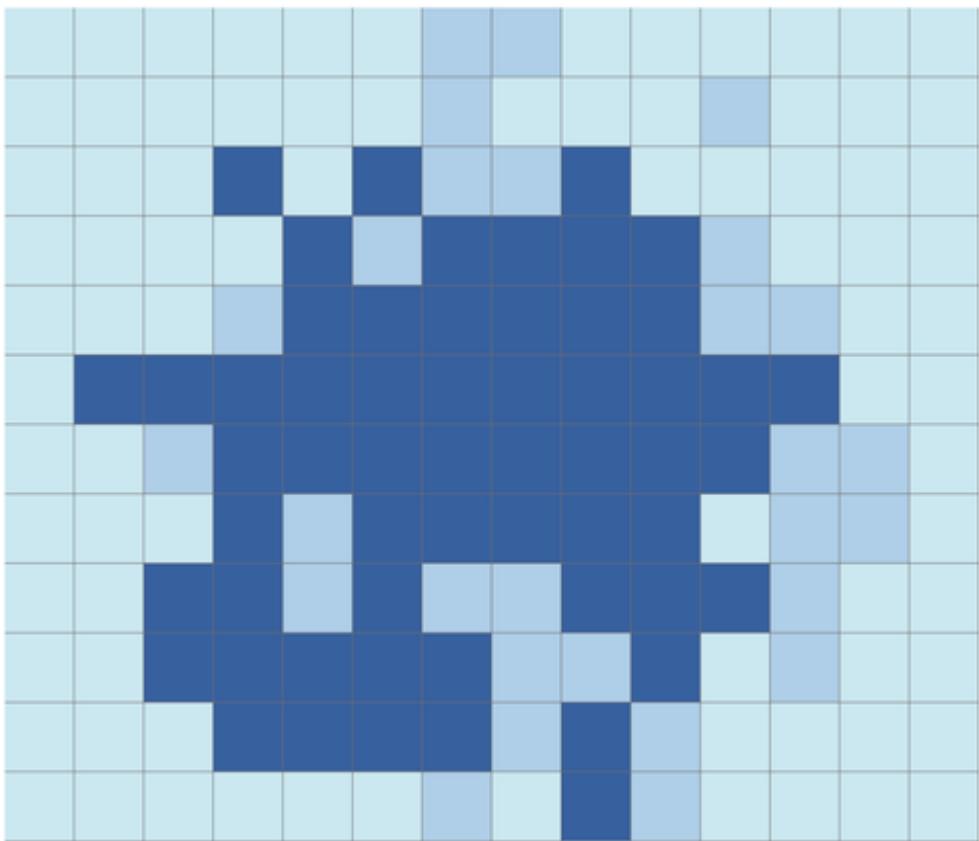
Hvor mange procent mindre blev billedet?



iStockphoto.com/AVIcons og Systime

Komprimering med tab

Ser vi på det samme billede som før og fjerner en farve, ser det sådan ud:



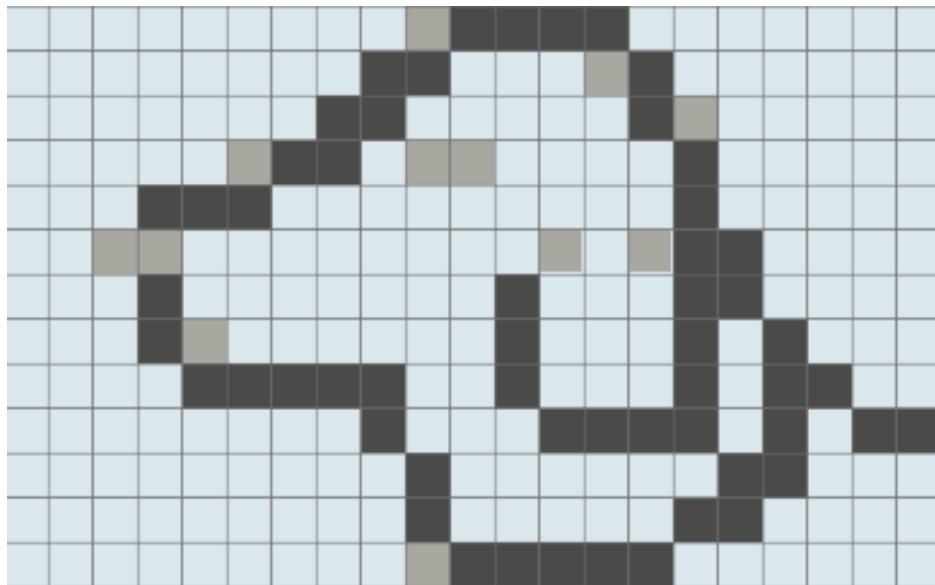
iStockphoto.com/Fnadya76 og Systime

Med dette udgangspunkt er vi ikke i stand til at genskabe det originale billede, fordi vi har mistet informationen om, hvilke af pixlerne der havde den fjerde farve.



Opgave: Komprimering med tab

Fjern en farve for dette billede ved at farve alle grå pixler sort. Lav en repræsentation af billedet og beregn hvor meget mindre billedet blev.



iStockphoto.com/AVIcons og Systime



Opgave: Komprimering af billeder

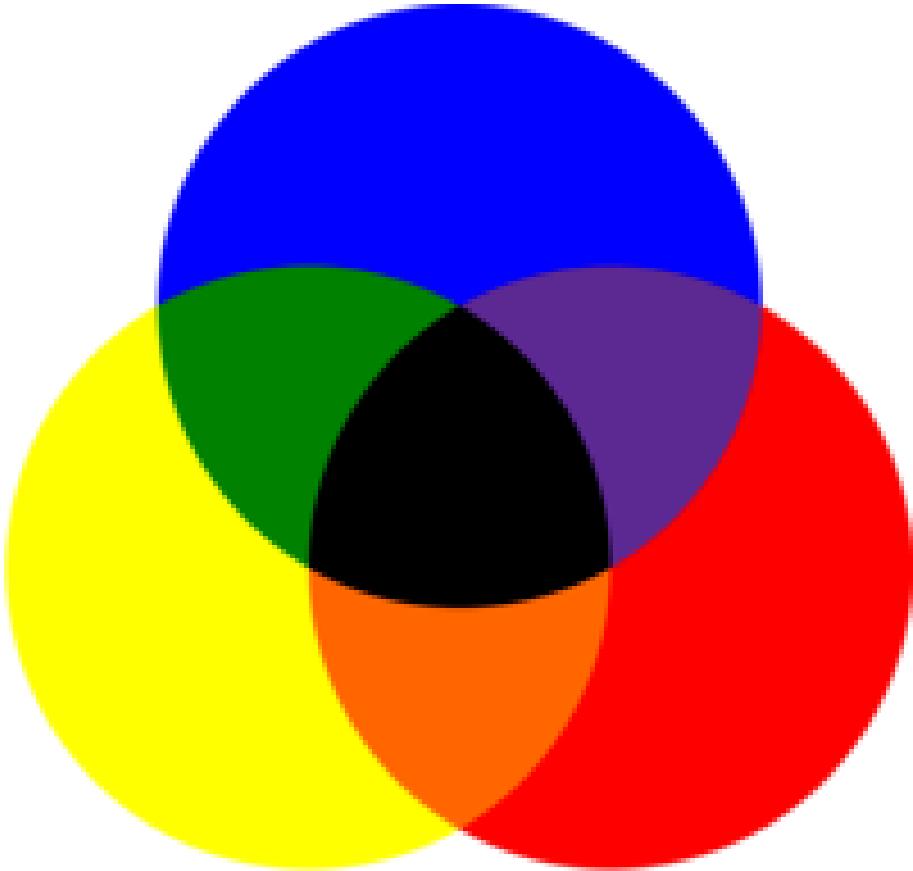
Find et tegneprogram, der kan komprimere billeder. Lav eksperimenter, hvor du komprimerer billeder med og uden tab – se, hvordan det forandrer filstørrelsen.

Farver

Her vil vi se på farver og farveblandinger. Grundlæggende er der forskel på pigmentfarver (maling, blyant o.l.) og farver til skærm.

Når man arbejder med pigmentfarver, er der tale om det subtraktive farveprincip, fordi pigmentfarver opstår ved at materialet med farve reflekterer og absorberer lys.

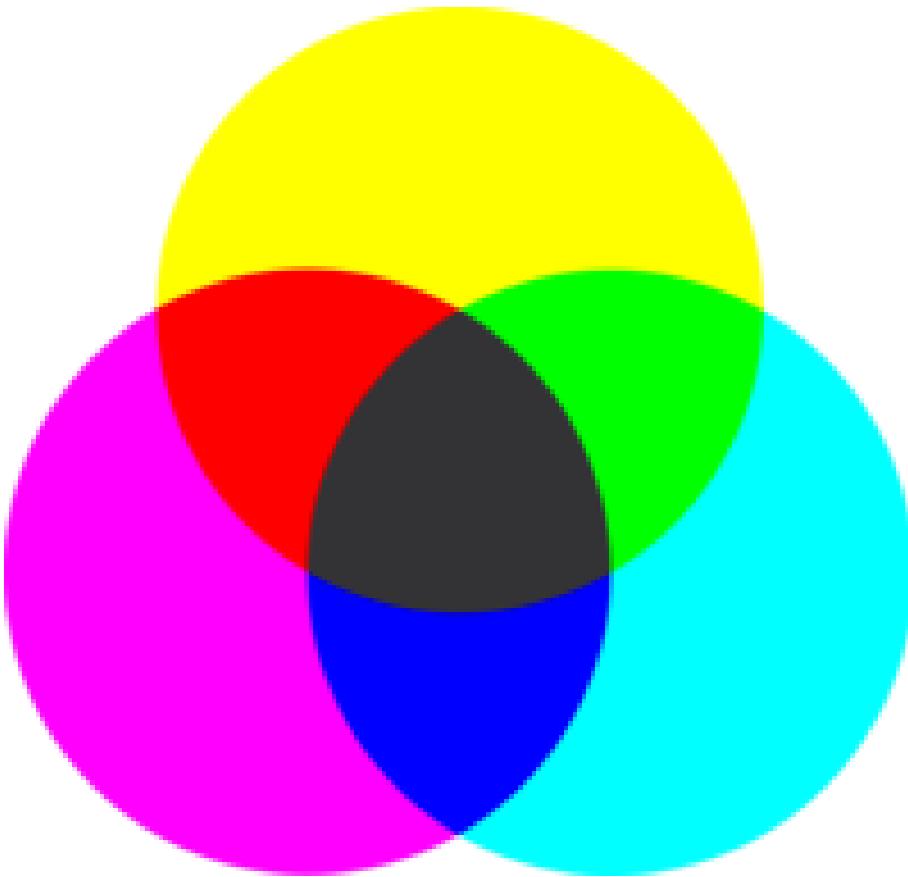
Her er to eksempler på grundfarver til det subtraktive farveprincip. I begge tilfælde giver sammenblandingen af alle tre grundfarver sort, da lyset her ikke reflekteres. Ingen farvetil sætning giver hvid (hvis dette er farven på det, man maler på).

Rød, blå og gul

Når man køber farver til at male billeder, er det nok at købe grundfarverne rød, blå og gul.

Ud fra disse tre grundfarver kan man lave alle andre farver. På figuren ses, at de sekundære farver er lilla, orange og grøn.

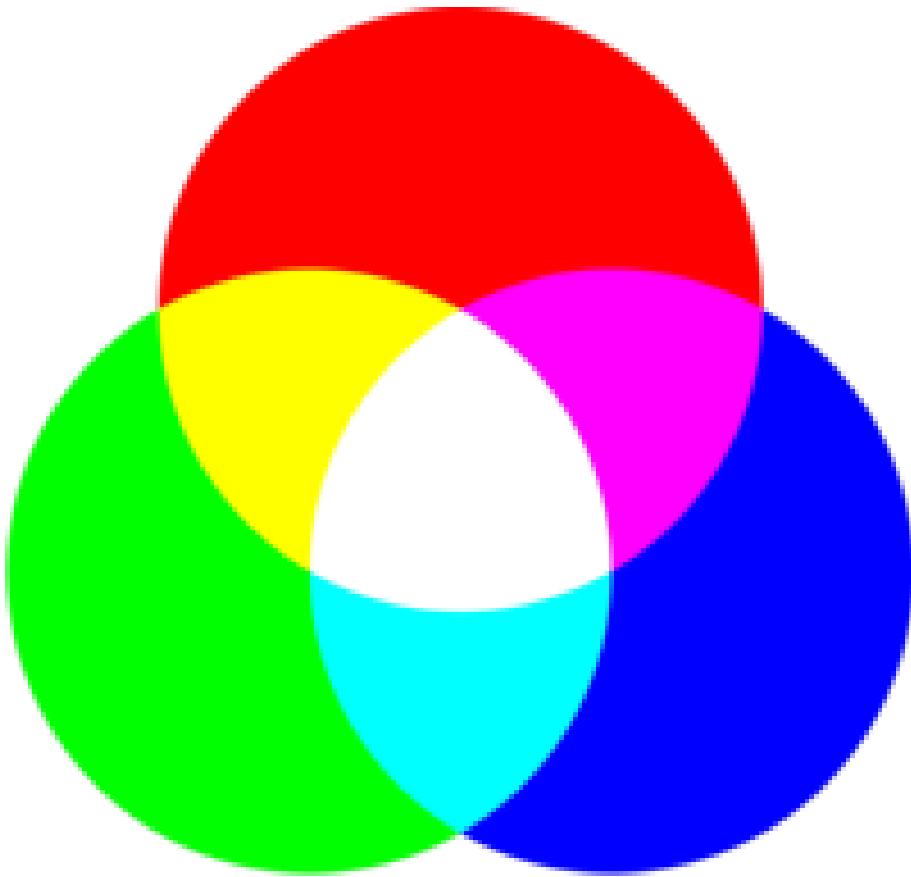
Cyan, magenta og gul (CMYK)



Når man køber farver til sin printer, køber man grundfarverne cyan, magenta og gul (yellow). Desuden køber man key-farven sort, fordi en blanding af de tre grundfarver ikke giver en sort farve i høj nok kvalitet.

Sekundærfarverne er rød, blå og grøn.

Rød, blå og grøn (RGB)



Når man arbejder med farver til skærm, er der tale om det additive farveprincip, fordi der er tale om farver i form af bølger, der lægges sammen. Her er primærfarverne rød, blå og grøn. Sammenblandingen af alle tre primærfarver giver hvid. Ingen farve giver sort, fordi der så intet lys udsendes.

Sekundærfarverne er cyan, magenta og gul.

Farvernes betydning

Farver kan tillægges betydning. For eksempel når gul benyttes i reklamer, betyder det ofte "tilbud" eller "billigt". Farvers betydning er ikke den samme på tværs af kulturer.



Opgave: Farvernes betydning

Find ud af, hvilken betydning der tillægges farverne rød, blå, gul, grøn, sort og hvid.
Har disse farver en anden betydning i andre kulturer?

Computeren

En computer er hardware, der styres af software.

Følgende er eksempler på hardware: processoren, hukommelse, tastatur, skærm, harddisk, printer, grafikkort, bundkort, netkort, DVD-drev.



Computer



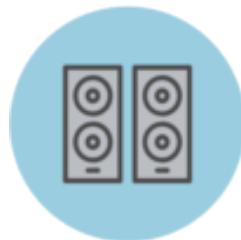
iPad



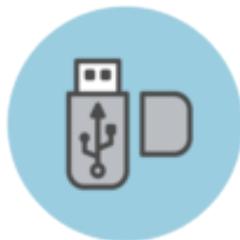
Mobil



Tastatur + mus



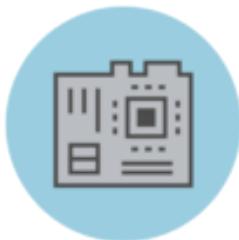
Højttalere



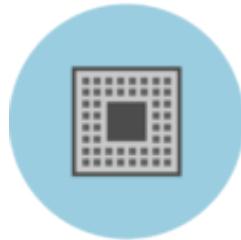
USB-nøgle



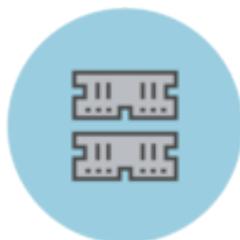
Printer



Bundkort



CPU



RAM



Harddisk



DVD-drev

Eksempler på hardware.

iStockphoto.com/fonikum/IconicBestiary/Alex Belomlinsky / Systime

Software er for eksempel internet-browsere, tekstbehandlingsprogrammer og filmafspilningsprogrammer – men også operativsystemer, sms-ordbøger, GPS-systemer og elektroniske automatpiloter til rutetrafik med fly.



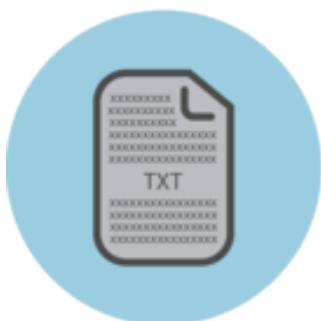
Browser - fx Chrome



Søgemaskine



Chatprogram



Tekstprogram - fx Word



Tegneprogram



Kalenderprogram

*Eksempler på software.
iStockphoto.com/fonikum / Systime*

Processor (CPU) og hukommelse (RAM)

En computers opgave er basalt set at afvikle programmer. Det gøres med to enheder,

- *processor*: afvikler programmer,
- *hukommelse*: opbevarer de programmer, der er under afvikling.

Tilsammen kan man kalde processoren og hukommelsen for "computerens hjerne".

Processoren kaldes også for *CPU'n* (engelsk: *central processing unit*). Programafvikling er faktisk bare *behandling af data* – det kaldes også *dataprocessering*, deraf ordet "processing" i navnet CPU.

Hukommelsen kaldes også for *RAM'en* (*random acces memory*).

Hardware-opbygning

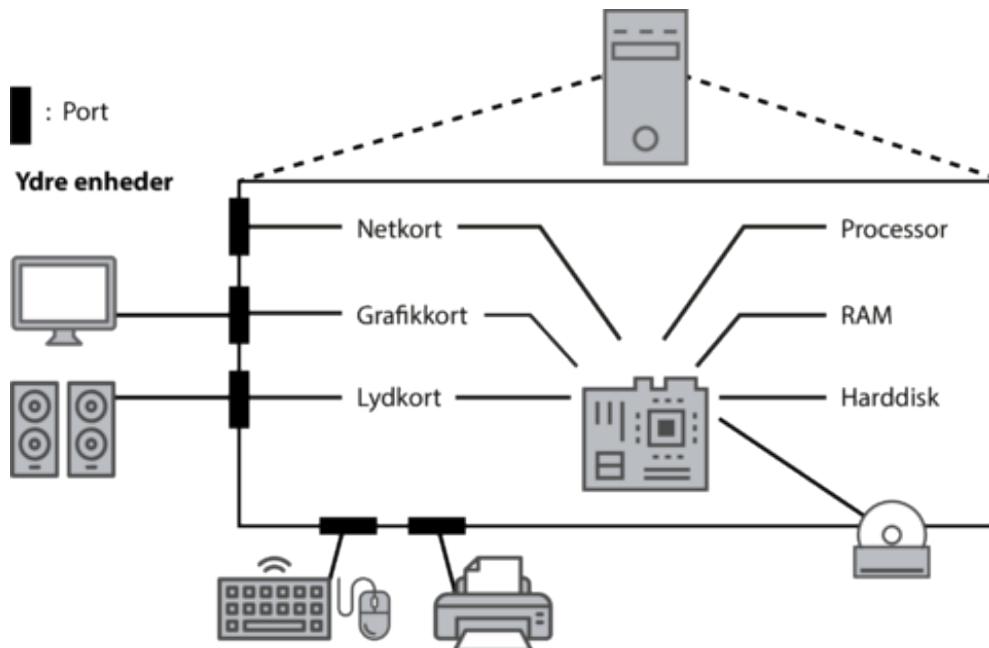
En computer er fysisk opbygget af processor, hukommelse samt yderligere tilkoblede enheder (engelsk: *devices*).

Ser vi på en bærbar computer, er enheder såsom skærm og tastatur ydre enheder selvom det hele er samlet i en enhed.

Eksempel: Enheder

Tastatur, skærm, mus, touchpad, printer, netkort, hukommelse og processor er alle eksempler på enheder.

Alle enheder er forbundet til computerens *bundkort*.

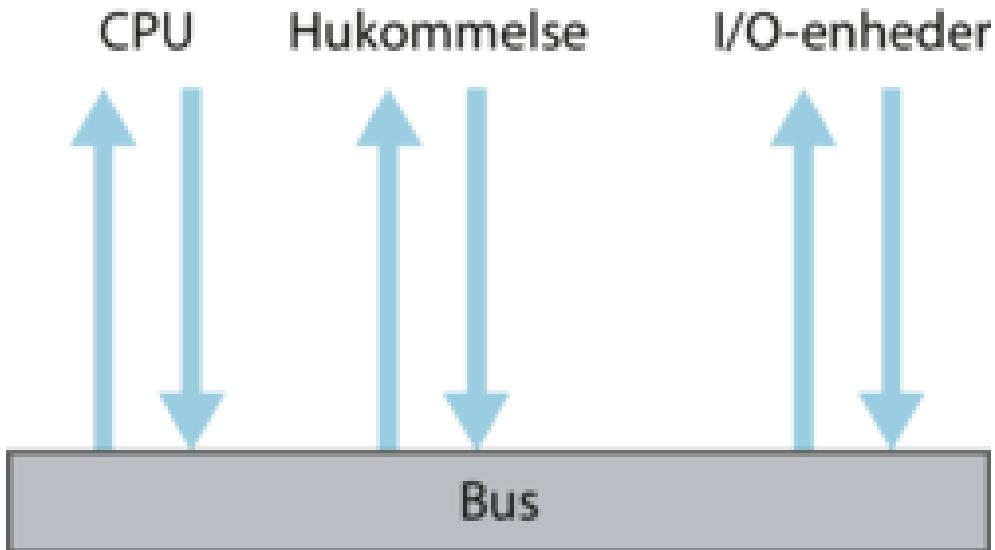


Computerens opbygning: Enheder koblet på bundkortet.

iStockphoto.com/fonikum/IconicBestiary / Systime

Ydre enheder tilkobles særlige porte på computeren, der er forbundet til bundkortet.

Via ledninger på bundkortet kan enhederne med elektriske signaler snakke med processoren og hukommelsen. Ledninger kaldes tilsammen *bussen*.



Computerens bus.

I/O-enheder

En I/O enhed er en enhed, der ikke er processor eller hukommelse.

Forkortelsen I/O står for Input/Output. Enhver I/O-enhed kan en eller begge af følgende:

- formidle beskeder til andre enheder (levere input),
- modtage beskeder fra andre enheder (modtage output).

Eksempel: Mus, skærm og netkort er I/O-enheder

Tastatur, skærm, touchpad, printer og netkort er alle eksempler på I/O-enheder. En skærm modtager beskeder (output) fra processoren om, hvad den skal vise på displayet, men leverer ikke noget input. En touchpad formidler bruger-bevægelser som signaler (input) til processoren, men modtager ikke noget output. Et netkort formidler data modtaget over et netværk (input) og modtager data, der skal sendes over netværk (output).



En mus formidler beskeder fra omverdenen til processor og hukommelse.
iStockphoto.com/fonikum/artvea / Systime

Hver enhed kan betjenes gennem et sæt af specifikke kommandoer, dvs. kommandoer, der er helt specielt beregnet til lige netop den pågældende enhed. Operativsystemet på computeren vil gerne kommunikere med enheden ved hjælp af mere generelle kommandoer. Operativsystemet på computeren kommunikerer derfor med hver enhed gennem enhedens *driver*, en software-stump, der oversætter mellem operativsystemets generelle kommandoer og enhedens specifikke kommandoer.

Opbygning af RAM

Computerens hukommelse, RAM'en, er den eneste lagerenhed, som CPU'en kan kommunikere direkte med.

Hukommelse (RAM)

Enhed, der opbevarer programmer under afvikling samt data, som disse programmer behøver.

Hukommelsen er opbygget som en fortløbende række af celler, der alle har samme størrelse og indeholder data. Hver celle har en adresse. Adressen angives med et tal. Data i celler repræsenteres med tal. Således indeholder hver celle et tal. I en computer er alle talværdier *binære*, dvs. opbygget af symbolerne 0 og 1. Symbolerne 0 og 1 kaldes *bits*.

| Adresse | Datacelles værdi |
|---------|---|
| 000 | 1001 0100 0111 1011 1110 1110 0011 1000 |
| 001 | 1101 1100 0101 1001 0010 0000 1101 0111 |
| 002 | 0011 0100 0000 1001 0100 0100 1110 0111 |
| ... | ... |

| Adresse | Datacelles værdi |
|---------|---|
| 999 | 1000 0100 0111 1111 0111 0111 0101 1000 |

En hukommelse med 1.000 adresser kan tegnes som et skema med 1.000 rækker. Hver datacelle indeholder her et 32 bit langt binært tal.

En hukommelse har en *controller*, der styrer adgangen til hukommelsen. Vil man *læse* værdien i en bestemt datacelle, fortæller man controlleren cellens adresse, hvorefter controlleren udleverer den angivne celles værdi. Vil man *skrive* i en bestemt celle, fortæller man controlleren adressen på den celle, der skal skrives i, samt hvad der skal skrives i cellen; herefter sørger controlleren for at overskrive cellens dataindhold.

Andre navne for RAM

Computerens hukommelse kaldes også for internt lager, primært lager, main memory, primary storage og, som nævnt, RAM.

Harddisk og andet lager

Det ville være ideelt, hvis alle de programmer og alt det data, vi ønsker at arbejde med, kunne opbevares i hukommelsen permanent. Det er dog umuligt. Når der slukkes for strømmen til computeren, *slettes* hukommelsen.

Computere er derfor udstyret med sekundært lager, fx i form af harddiske. En harddisk kan rumme masser af data, og data slettes ikke, hvis strømmen til den slukkes. Til gengæld har CPU'en ikke direkte adgang til harddisken, så data fra harddiske skal indlæses i hukommelsen, før CPU'en kan arbejde på det.

Læsninger og skrivninger til harddiske tager væsentlig længere tid end læsninger og skrivninger til hukommelsen. En vigtig del af en effektiv computer er derfor et system til *lageradministration*, som sørger for at mindske trafikken mellem harddiske og hukommelse.

Harddiske er ikke det eneste sekundære lager. Der er også eksterne lagermedier som fx CD'er, DVD'er, USB-nøgler, eksterne harddiske eller backup-servere.

Filer

Data lagres som filer. Det kan være billedfiler, videoklip, tekstdokumenter eller adressebøger til emailprogrammer. Også programmer lagres i filer. Filer opbevares som regel over længere tid på fx en harddisk. Hver enkelt fil har et filnavn og filer har ofte en *filtype* (også kaldet et *filformat*), der signalerer filens indholdstype. Et eksempel på en filtype er .jpg som er et billedformat.



iStockphoto.com/Tiyas

Gennem computerens *filesystem* kan man oprette, slette, læse, redigere, omdøbe og flytte filer. Filsystemet oversætter handlinger udtrykt på et menneskevenligt, abstrakt plan (såsom "slet fil", "se filstørrelse" eller "åbn fil") til konkrete handlinger på maskinniveau (fx at tilgå bestemte adresser i hukommelsen, indlæse bestemte harddisk-områder i hukommelse eller skrive bestemte dataværdier i bestemte harddisk-områder).

Liste af kendte filtyper

| | | | | | | | | | | |
|----------|--|--|--|--|--|--|--|--|--|--|
| vector | | | | | | | | | | |
| image | | | | | | | | | | |
| text | | | | | | | | | | |
| audio | | | | | | | | | | |
| video | | | | | | | | | | |
| ebook | | | | | | | | | | |
| archive | | | | | | | | | | |
| internet | | | | | | | | | | |
| other | | | | | | | | | | |
| bonus | | | | | | | | | | |

iStockphoto.com/-1001-

- **.bmp:** en billedfil i bitmap-format.
- **.docx:** et Word-dokument.
- **.exe:** en programfil, der kan afvikles. exe kommer af det engelske ord "executable", der betyder "kan afvikles".
- **.html:** en HTML-fil.
- **.jpg:** en billedfil i jpeg-format.
- **.txt:** en tekstfil.
- **.zip:** en zip-fil - et komprimeret fil-arkiv. Også 7z, ace og gz er kendte kompressions-formater.
- **.xlsx:** et Excel-ark.

Bits og bytes

Bits

En *bit* er et af symbolerne 0 eller 1. En computer kan *repræsentere alle tal* kun ved hjælp af bits (det sker med det såkaldte *binære talsystem*). Alle programmer og alt data lagres som tal, så har vi tallene, har vi også programmer og data.

Inde i en computer foregår al databehandling med de fornævnte binære tal, hvor der kun er de to cifre 0 og 1. Det er praktisk for computeren, fordi de to cifre blot svarer til to forskellige fysiske tilstande. Fx kan en computer repræsentere cifrene 0 og 1 med to elektriske signaler, der har forskellig spænding.

Vil vi forstå det binære talsystem, må vi først repetere et par velkendte ting omkring vores sædvanlige 10-talsystem, det *decimale talsystem*.

Det decimale talsystem

I det decimale talsystem benytter vi symbolerne 0, 1, ..., 9. Symbolerne kaldes de *decimale cifre*. Der er 10 forskellige decimale cifre, og talsystemet siges derfor at have grundtal 10. Hvis vi vil vide, hvad tallet med de decimale cifre 98443 er for et tal, kan vi regne det ud, fordi det decimale talsystem er et *positionssystem*. I et positionssystem *bestemmer hvert ciffers position, hvilken potens af grundtallet, cifret repræsenterer*.

Et eksempel: Tallet 98443 tolkes ud fra cifrenes positioner (9: position 4, 8: position 3, 4: position 2, 4: position 1, 3: position 0) som følgende tal:

$$\begin{aligned} 98443 &= \\ 9 \cdot 10^4 + 8 \cdot 10^3 + 4 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0 &= \\ 9 \cdot 10 \cdot 10 \cdot 10 \cdot 10 + 8 \cdot 10 \cdot 10 \cdot 10 + 4 \cdot 10 \cdot 10 + 4 \cdot 10 + 3 \cdot 1 &= \\ 9 \cdot 10000 + 8 \cdot 1000 + 4 \cdot 100 + 4 \cdot 10 + 3 & \end{aligned}$$

Det binære talsystem

Det *binære talsystem* er opbygget med to cifre i alt: 0 og 1. Det binære talsystem består af 2 forskellige cifre og har derfor grundtal 2.

Fordi det binære talsystem er et positionssystem, kan vi regne ud, hvilken værdi f.eks. tallet 11001 svarer til. Hvert ciffers position bestemmer grundtallets potens, så vi kan udregne

$$\begin{aligned} 11001 &= \\ 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 &= \\ 1 \cdot 16 + 1 \cdot 8 + 0 + 0 + 1 & = 25 \end{aligned}$$

Her har vi angivet det binære tal 11001's værdi som det decimale tal "25", altså femogtyve. For ikke at sammenblande talsystemer kan man angive hvert tals talsystem med et fodtegn:

$$11001_2 = 25_{10}$$



Opgave: Omskrivning mellem binær og decimal repræsentation

Omskriv følgende tal til binær repræsentation:

- 12_{10}
- 15_{10}
- 100_{10}

Omskriv følgende tal til decimal repræsentation:

- 1111111111_2
- 100_2
- 1_2

Ekstra opgave: Omskriv 100_{10} til det heksadecimale talsystem. Her er cifrene 0 1 2 3 4 5 6 7 8 9 A B C D E F

Bytes

En *byte* en samling af 8 bits. Fx er både 00110111 , 11110001 og 11111111 bytes. En byte repræsenterer et tal mellem 0 og 255, hvis den fortolkes som et binært tal. Fx kan vi udregne, at

$$00110111_2 = 55_{10}, \quad 11110001_2 = 241_{10} \text{ og } 11111111_2 = 255_{10}$$

Når man mäter lagerplads eller hukommelseskapacitet på en computer, sker det som regel med udgangspunkt i bytes. 1 byte kan som regel lagre data svarende til 1 tegn (se ASCII).

Man bruger følgende enheder

- kilobytes (kB) = 1.000 bytes
- megabytes (MB) = 1.000 kilobytes = 1.000.000 bytes
- gigabyte (GB) = 1.000 MB = 1.000.000.000 bytes
- terabyte (TB) = 1.000 GB = Et tusind millarder bytes

Et par eksempler:

En kilobyte data svarer til teksten i en almindelig email.

En megabyte data svarer til omkring teksten fra en roman eller et minuts MP3-musik med standard-bitrate.

Det nu kortlagte menneskelige genom kan lagres med omkring 0,8 gigabytes plads. Med en gigabyte kan man lagre teksten til en container fuld af krimier.

Med en terabyte kan man lagre data svarende tekstdindholdet af samtlige bøger på Københavns Hovedbibliotek.

Opgaver til Computeren



Opgave: Binære tal

- Hvilket decimalt tal svarer det binære tal 10 til?
- Hvilket decimalt tal svarer det binære tal 0100 til?
- Hvilket decimalt tal svarer det binære tal 10111010011 til?
- Omskriv det decimale tal 23 til et binært tal.



Opgave: Hexadecimale tal

- Hvilket decimalt tal svarer det hexadecimale tal E til (ja! det er et let spørgsmål)?
- Hvilket decimalt tal svarer det hexadecimale tal B910E til?
- Udregn EDB - ABE og angiv svaret som decimalt tal.
- Omskriv decimaltallet 123 til hexadecimale cifre.



Opgave: Hexadecimale tal

2 hexadecimale cifre efter hinanden angiver et tal mellem 0 og 255 (hvorfor)? Derfor kan man notere talværdien af en byte med to hexadecimale cifre, fx A4.

- Skriv byten 1100 1001 med to hexadecimale cifre.
- Hvilken byte svarer det hexadecimale tal 6F til?



Opgave: Regning med binære tal

Man regner med binære tal, som man gør med decimale tal. Vi kan fx udregne plusstykket $01001 + 11100$ ved at opskrive et sædvanligt udregningsskema for plusstykker,

$$\begin{array}{r}
 10_2 \quad 10_2 \\
 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \\
 + \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \\
 \hline
 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1
 \end{array}$$

Herunder et eksempel på minusstykke:

$$\begin{array}{r}
 1 \quad 1 \quad 1 \quad 0 \\
 - \quad 1 \quad 0 \quad 1 \quad 0 \\
 \hline
 0 \quad 1 \quad 0 \quad 0
 \end{array}$$

- Udregn $010010 + 011110$ uden at omskrive til decimaltal.
- Udregn $111100 - 110011$ uden at omskrive til decimaltal.
- Gennemtænk, hvordan du normalt ganger to store tal sammen i hånden. Forsøg at gange de to binære tal 1010 og 1000 sammen i hånden efter samme procedure (uden at omskrive til decimaltal). Virker proceduren for multiplikation også for binære tal? (Du kan kontrollere, om du har fået det rigtige svar ved at omskrive resultatet af dit regnestykke til decimaltal; resultatet er 80).



Opgave: Tegnsæt og kodning af tegn

- Hvad er Unicode?
- Undersøg, hvilket Unicode-alfabet dit tekstbehandlingsprogram bruger.
- Hvad er UTF-8? Hvad står 8-tallet for?

Deleøkonomi



iStockphoto.com/filo

OBS

Dette er kernestof på B-niveau.

Når man har behov for en seng at sove i, bestiller man et hotelværelse, og når man har behov for at blive transporteret, bestiller man en taxa. I begge tilfælde er der en virksomhed, der varetager opgaven og får penge for det. Virksomheden er underlagt love som sikrer bl.a. skatteregler og forsikringer.

Med internettets udbredelse er der opstået forskellige eksempler på deleøkonomi, hvor privatpersoner stiller fx værelser eller biler til rådighed for andre privatpersoner. Her er tale om en slags udlejning, og derfor skal der en anden form for lovgivning til.

I deleøkonomi deler man ofte ting, der i perioder ikke bliver brugt. Man stiller sit overskud til rådighed.



Opgave: Deleøkonomi

- Find eksempler på tjenester, der tilbyder værelser eller køreture.
- Find eksempler på andre deleøkonomi-tjenester.
- Analysér fordelene og ulemperne ved deleøkonomi i forhold til bl.a. indtægter og udgifter.
- Vurdér konsekvenserne af deleøkonomi for brugerne og for erhvervslivet.



Opgave: Boligbytte

Gå på nettet og vælg et eksempel på en tjeneste til deleøkonomi, der handler om boligbytte i ferier.

- Undersøg hvad vilkår og krav er for et boligbytte på den valgte tjeneste.

Sammenlign i følgende spørgsmål med leje af et hotelværelse:

- Hvilke forventninger vil du have til din gæsts adfærd, hvis du er udlejer af dit værelse?
- Vil der være forskel på din opførsel som gæst i forhold til behandlingen af inventaret?
- Vil der være en forskel på den måde du som gæst forlader stedet på?
- Er det de samme etiske regler der gælder for leje af hotelværelse og for boligbytte?

Digital selvbetjening



iStockphoto.com/-VICTOR-

I 2012 blev loven om ”Obligatorisk digital selvbetjening” vedtaget. Det betyder, at for eksempel fornyelse af fisketegn foregår via internettet. Man går på hjemmesiden borger.dk og ansøger her.

Borger.dk er en internetportal, hvor man kan hente ydelser fra mange forskellige offentlige instanser. Vil man læse om fritidsjob, kan man søge på Borger.dk og herunder finde et link til for eksempel Jobpatruljen.dk.



Opgave: Hvilke muligheder har man på borger.dk?

Gå på borger.dk, og find svarene på følgende spørgsmål:

- Hvad er NemID? - hvad bruges det til?
- Hvad står der om fotografering af NemID-nøglekort?
- Hvilke to steder kan du læse din digitale post fra det offentlige?
- Hvilke kriterier skal man opfylde for at slippe for at få sin post fra det offentlige digitalt?
- Find seks emner, du enten har brugt eller i nær fremtid skal bruge fra borger.dk.
- Hvad er NemKonto?
- Hvad er Udbetalning Danmark, og hvad er registersamkøring? Hvorfor benytter Udbetalning Danmark registersamkøring?
- Er der krav om, at en ansøgning om økonomisk fripladstilskud til en folkeskole-plads SKAL være digital? (svaret står i en lovetekst).



Opgave: Borger.dk og brugeradfærd

Lav en analyse af borger.dk's påvirkning af din og andres adfærd i forhold til det offentlige.

Vurdér fordele og ulemper ved bl.a. registersamkøring.

Vurdér etiske overvejelser ved registersamkøring.

HTML og CSS

HTML

HTML er en forkortelse for **HyperText Markup Language**, og det er således et markerings-sprog og *ikke* et programmeringssprog. HTML bruges til at opmærke indholdet på en hjemmeside og skal ses som en måde at strukturere indhold på.

HTML definerer en række markeringskoder, kaldet *tags*, som benyttes, når en hjemmeside skal konstrueres.

Eksempel: HTML struktur

Den simple struktur af en (tom) hjemmeside ses herunder:

```
<html>
  <head>
    ...
  </head>

  <body>
    ...
  </body>
</html>
```

En hjemmeside er det samme som et HTML-dokument. En browser er det program, man bruger til at se hjemmesiderne/HTML-dokumenterne gennem. Browseren omdanner HTML-dokumentet til et grafisk billede på computerskærmen.

HTML-dokumenter, af typen .html, indeholder ren tekst. En tekst-editor til at skrive hjemmesider i, behøver derfor kun at kunne skrive ren tekst.

Tags og attributter

Tags forekommer i 2 varianter: *container-tags* og *separator-tags*. Container-tags består altid af et start-tag og et slut-tag, som fx `<html>....</html>`. Separator-tags kun består af ét tag, som fx ``.

Tags kan forsynes med attributter, som fx ``, hvor attributten er `src` (**s**ource) med værdien "bil.jpg" (et navn på en grafik-fil). En anden skrivemåde kunne være:

```
<
  img src="bil.jpg"
>
```

Kommentarer i HTML

I HTML kan man angive kommentarer som ignoreres af browseren, som følger:

```
<!-- Dette er en kommentar -->
```

Validering og W3C

Man kan sikre sig, at det HTML-dokument, man har lavet, overholder de gældende regler for opmarkering af hjemmesideindhold ved at konsultere WorldWideWeb-konsortiets (W3C's) [hjemmeside](#) og her få hjemmesiden valideret.

CSS

CSS er en forkortelse for Cascading Style Sheets. CSS bruges til at definere layout på HTML-dokumenter. Fordelen ved at adskille *indhold/struktur* (HTML) fra *præsentation/layout* (CSS) er flere HTML-dokumenter kan have deres layout defineret i ét CSS-dokument.

Hvis man fx vil lave en hjemmeside, hvor baggrundsfarven er sort og tegnene hvide, kan man i HTML skrive: <body bgcolor="black" text="white">...</body>.

Da layout bør laves i CSS, kan vi i stedet i CSS skrive:

```
body{
    background-color: black;
    color: white;
}
```

Integrering af CSS i HTML

Der er 3 måder, hvorpå CSS kan integreres i et HTML-dokument:

Styling direkte på HTML-element

```
<html>
    <head>
        ...
    </head>

    <body style="background-color:black; color:white">
        ...
    </body>
</html>
```

Denne metode kan bestemt IKKE anbefales, idet vi efter har mudret HTML-strukturen til med en layout-attribut: style.

Style-tag i -afsnit

```
<html>
    <head>
        ...
    </head>
```

```
<style> body{  
    background-color:black;  
    color:white;  
</style>  
</head>  
<body>  
...  
</body>  
</html>
```

Denne metode er lidt bedre end den foregående. Vi har indsat ét eneste style-element, hvorved stylingen af diverse tags nu er placeret ét sted i HTML-dokumentet.

Link til style-anvisninger

Link til style-anvisninger (= et stylesheet) i en selvstændig, ren tekstfil af typen .css

```
<html>  
  <head>  
    <link href="stil.css" rel="stylesheet" type="text/css">  
  </head>  
  
  <body>  
  ...  
  </body>  
</html>
```

Filen stil.css indeholder præcis nedenstående:

```
body{  
    background-color:black;  
    color:white;  
}
```

Denne metode er klart at foretrække. Stylingen skrives i en selvstændig fil, og der linkes til denne CSS-fil via et `<link>`-tag i HTML-dokumentets `<head>`-afsnit. Dette kan gøres i hvert eneste HTML-dokument, der skal styles. Herved har man fået adskilt indhold og layout.

Det er nu enkelt at ændre udseendet af samtlige HTML-dokumenter i en applikation, idet man blot redigerer indholdet af én fil: stil.css.

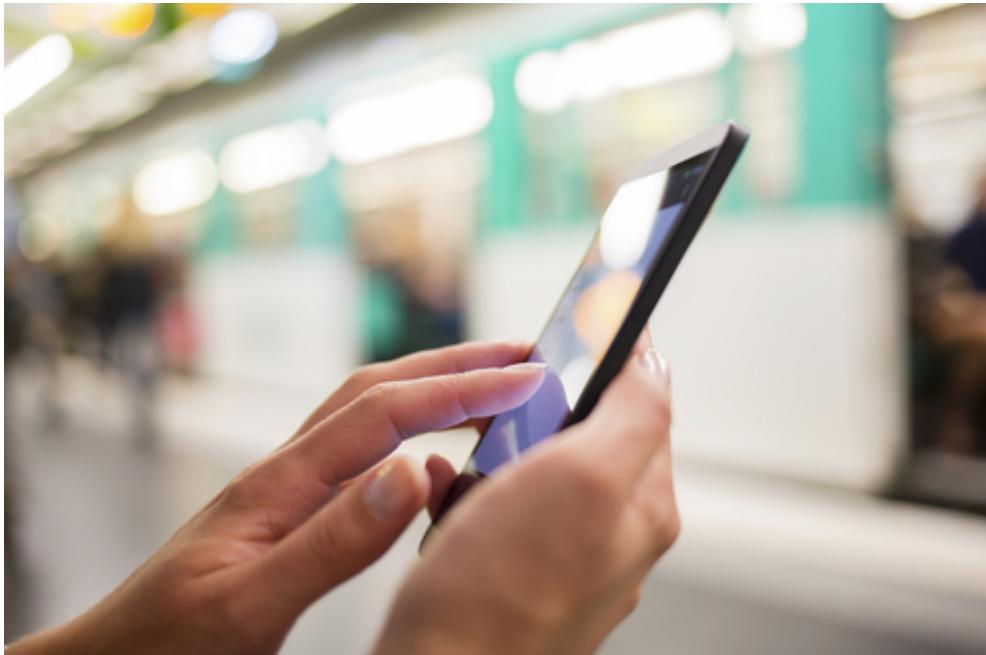
Flere style sheets

Man kan linke til mere end ét style sheet i et HTML-dokument. Rækkefølgen af disse links er afgørende, da det senest angivne style sheet overskriver eventuelle style-deklarationer fra de foranstående style sheets.

Innovation i IT

Begrebet innovation bruges i alle mulige sammenhænge, og vi kan også tale om innovation i it. Her vil vi se på forskellige it-produkter og afgøre, om de er innovationer.

Smartphone



iStockphoto.com/LDProd

Lad os se på en smartphone som et eksempel.

En smartphone er et it-produkt, hvor man har samlet forskellige elementer såsom gps, telefon, spil, mail og film. Desuden kan den fyldes med apps som den enkelte bruger vælger efter behov.

Dermed tilpasser den enkelte ejer den til sine personlige behov, og en smartphone bliver en personlig ejendel.

Innovationsbegrebet

Innovationsbegrebet kan defineres på mange måder. Når vi har brug for et innovationsbegreb, der kan afgøre, om et it-produkt er en innovation, kan vi bruge følgende definition:

Innovationsbegrebet i IT

Man kan sige at et it-system er en innovation, hvis det er

- nyt
- nyttigt
- relevant

Med denne definition vil man kunne sige, at en smartphone er en innovation. Den lever op til alle tre krav i definitionen:

- Den er ny, da man samlede gamle teknologier i ét samlet produkt. Desuden opstod der i sammenspillet mellem de enkelte produkter noget nyt. For eksempel kan en håndholdt GPS kombineret med et elektronisk kort bruges til at navigere rundt med. Dette er en erstattning af brugen af et papirkort.
- Den er nyttig, da man kan nøjes med ét produkt som erstattning for en række produkter.
- Den er en relevant. Et eksempel er anvendelsen af apps til at finde pårørende efter naturkatastrofer.

Eksempel på ikke-innovation



Foto: iStockphoto.com/AnnSteer

En hjemmeside til en gårdbutik, der går online, er ikke en innovation. Den er ny i forhold til den konkrete gårds daglige praksis, men typen af hjemmeside er ikke ny. Der findes andre gårdbutikker, der er gået online.



Opgave: Innovative IT-produkter

- Find et it-produkt, og argumentér for at det er en innovation. Dvs. at det skal opfylde alle tre krav fra definitionen.
- Find et it-produkt, der ikke er eksempel på en innovation. Dvs. at det skal fejle på mindst et af de tre krav.

Når man vha. definitionen har afgjort, at et it-produkt er en innovation, kan man undersøge innovationen nærmere. Det vil vi gøre i de to næste afsnit.

4p-modellen for innovation

Når vi har et it-system, der er en innovation, kan vi vurdere, hvordan der er sket en ændring set i forhold til produkt, proces, position og paradigme. Da alle fire kategorier starter med p, kaldes det 4p-modellen for innovation.

- Produkt-innovation. Her har vi fokus på ændringen af et produkt.
Et eksempel er vores førstomtalte smartphone. De enkelte dele, såsom gps, telefon og spil, findes i forvejen, men samlet i en smartphone er der tale om et andet produkt.
- Proces-innovation. Her er der fokus på ændringen af en proces i en virksomhed.
Et eksempel er firmaet ChiliHouse. De har ændret deres arbejdssprocesser, fordi handlen ikke længere foregår i en fysisk butikker, men på nettet.
- Positions-innovation. Her er fokus på ændringen af den position (kontekst) it-systemet anvendes i set fra brugerens/kundens synsvinkel.
Et eksempel er is. Her er fokus på ændringen fra, at is er slik til børn, til at is er en luksusspise for voksne.
- Paradigme-innovation. Her er fokus på ændringen af den grundlæggende selvforståelse i en virksomhed.
Et eksempel er firmaet IBM. Firmaet startede som producent af hardware, men i dag sælger de i højere grad software og konsulentydeler.

Læg mærke til, at det samme it-system kan være eksempel på flere kategorier.

Inden for hvert kategori taler man om, hvor stor eller lille en ændring innovationen er.

Eksempel: Ford Model T



iStockphoto.com/Peter Mah

Et eksempel på en innovation, der omfatter alle fire innovationskategorier, er bilproducenten Fords masseproducerede Model T, der blev lavet i starten af det 20. århundrede. Overordnet set ønskede Ford at producere en bil, der var for alle.

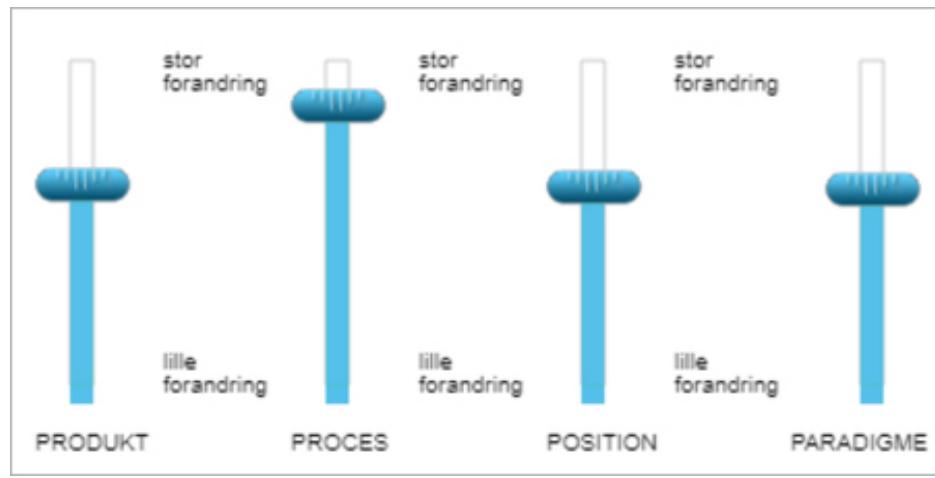
Modellen var en produktinnovation. Tidligere biler havde nemlig været skräddersyet til den enkelte køber, og Model T var i stedet en masseproduktion, hvor alle kunder fik det samme produkt. Dette satte prisen på bilen betragteligt ned.

Der var tale om procesinnovation, da det var helt nyt at lave masseproduktion. Fremstillingsprocessen blev markant ændret, da det ikke længere var én mand, der lavede en bil præcis efter én kundes ønsker og pengepung, men flere personer der lavede hver sin lille del på en række biler, som var ens. Masseproduktion som metode blev straks efter overført til produktion af andre produkter.

Da Model T var så billig, at markedet pludselig åbnede sig fra at bestå af de rigeste til at bestå af en meget bredere del af befolkningen, var der tale om positionsionsinnovation. Nu var Model T ikke mere en luksusvare til søndagsture, men alle mands øje og et almindeligt transportmiddel, der sås alle vegne.

Bilen var en paradigmændring, fordi Ford gik fra at se sig selv som producent af en luksusvare til en eksklusiv og pengestærk kundekreds, til at producere en meget billigere vare til en bredere kundekreds.

Her ses fire skydere, der viser et billede af, hvor stor (eller lille) ændring Model T var i forhold til de fire kategorier. Bemærk, at der er forholdsvis stor ændring på alle fire kategorier, men procesinnovation er valgt lidt højere, da den nye produktionsform bredte sig til andre produktionsvirksomheder.





Opgave: Er der tale om innovation?

Se på følgende eksempler:

- e-Boks til lønsedler og offentlige breve
- email
- pacemaker
- elbiler
- eBøger
- cowboybukser
- onlinemusik

Find ud af, om de er innovationer i forhold til om de er nye, nyttige og relevante. I så fald er de it-innovationer.

Hvis de er innovationer, placeres de i forhold til de fire p'er. Bemærk, at der ikke nødvendigvis er et eksempel til hvert p, og nogle af eksemplerne kan være relevante ved flere af innovations-p'erne.



Opgave: Find IT-produkter

Find fire IT-produkter. Er de innovationer?

Hvis de er innovationer, hvilke kategorier er de så innovationer indenfor? Og hvor stor/lille forandring er der tale om?

Brug nedenstående skydere.



Opgave: Se videoer

Se videooptagelserne af Preben Mejer her i iBogen. Udvælg nogle af de it-produkter, han omtaler. Argumentér for, at der er tale om innovative IT-produkter og argumentér for, hvilke kategorier der er i spil.

Brug nedenstående interaktive skydere.

Video/lyd/interaktiv opgave findes i iBogen (se <https://informatik.systime.dk/index.php?id=1022&L=0>)

Radikal og inkrementel innovation

Når vi har et it-system, der er en innovation, kan vi undersøge *hvordan* der er tale om en innovation. Er der tale om en forandring? Eller en forbedring?

Radikal innovation er, når der er tale om en markant innovation. Det vil sige at der er tale om en forandring.

Inkrementel innovation er, når der er tale om en forbedring i forhold til det eksisterende.

Bemærk at et it-system ikke både kan være en inkrementel innovation og en radikal innovation på samme tid.

Eksempel på radikal innovation

En radikal IT-innovation er denne særlige anvendelse af en mobiltelefon: Man kan bruge mobiltelefon-sensorerne til at afdække, om brugerens fysisk aktivitet er meget fysisk aktiv eller lidt fysisk aktiv.

Vi forestiller os, at brugerens diagnosen bipolar (tidligere brugte man ordet manio-depressiv). Ved at opsamle data om brugerens fysiske aktivitetsniveau fra mobiltelefon-sensorerne kan man undersøge, om personen nærmer sig en ydertilstand – manisk/fysisk aktiv eller depressiv/fysisk passiv. Her er tale om at "gøre noget anderledes" i forhold til, hvordan man ellers afgør, om en person er på vej til en af ydertilstandene.

Hør mere i denne video:

Eksempel på inkrementel innovation

Den nyeste måler til måling af luftkvalitet er et eksempel på en inkrementel innovation. Der er tale om gammel teknologi, men sensorerne er blevet markant billigere at producere, og dermed er anvendelsesmulighederne blevet mange flere. Bl.a. kan måleren installeres hjemme hos almindelige forbrugere. Der er tale om en forbedring, da man før i tiden målte luftkvalitet ved at se, om der var fugt i vinduerne etc.

Hør mere i denne video:



Opgave: Radikal eller inkrementel innovation

Se på følgende produkter. Er der tale om radikal eller inkrementel innovation?

- MobilePay
- Pulsur
- Tesla
- Sonos-rummåler

Interaktion med databaser

Meget af teksten på Facebook, webshops og andre hjemmsider bliver hentet i databaser, inden de bliver vist i browseren. Disse kaldes *dynamiske hjemmesider* og er karakteriseret ved, at deres indhold afhænger af brugeren.

På Facebook kan du eksempelvis se en liste over samtaler. Hver af disse samtaler kunne stamme fra en tabel i en database. I en webshop bliver du præsenteret for en liste over varer, og i din kurv kan du se, hvilke varer du vil købe.

I dette afsnit skal du bruge et program, hvis output er en hjemmeside med grundstoffer. Disse grundstoffers numre, navne og kemiiske symboler er input til programmet og hentes fra en database. Man kan forestille sig, at disse data kan bruges i en udvidelse af siden, hvor man kan sammensætte atomer til molekyler i undervisningen.



Opgave: Bruge programmet Grundstoffer

1. Åbn XAMPP og start modulerne Apache (webserveren) og MySQL (databasen).
2. Åbn phpMyAdmin i din browser. phpAdmin er et databaseværktøj som følger med XAMPP. Find det ved at åbne en browser og gå til adressen localhost/phpmyadmin, når XAMPP kører.
3. Importer filen [grundstoffer.sql](#) (*Filen kan downloades fra ibogen se <https://informatik.systime.dk/index.php?id=1099&l=0>*) (Ligger i pakket zip-fil).
4. Opret brugeren "programmering" med adgangskoden "systime" i phpMyAdmin, eller ret brugernavn og adgangskode i PHP-filen, så det passer til din database.
5. Kør programmet i browseren ved at skrive stien til PHP-filen.



Opgave: Udforskning af koden

Værdien af variablen `sql` i koden herunder er forespørgslen til databasen, som er skrevet i sproget SQL. Kort fortalt hentes egenskaberne nr og navn for ethvert grundstof i databasetabellen.

1. Gå ind på siden Grundstoffer og vis HTML-kildeteksten (Ctrl-U i Chrome, F12 i Edge).
2. Læs PHP-koden herunder og sammenhold det med HTML-kildeteksten.
3. Redegør ud fra ovenstående for linjerne i PHP-kode.

Kode: Grundstoffer (grundstoffer.php)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Grundstoffer</title>
</head>
<body>
<h1>Grundstoffer</h1>
<table>
<tr>
<th>Nr</th>
<th>Navn</th>
</tr>

<?php
\$dbh = new PDO("mysql:host=localhost;dbname=test",
    "programmering", "systime");
\$sql = "SELECT nr, navn FROM grundstoffer ";
\$sql .= "ORDER BY nr;";
\$stmt = \$dbh->prepare(\$sql);
if (\$stmt->execute()) {
    while (\$row = \$stmt->fetch()) {
        echo "<tr>";
        echo "<td>" . \$row["nr"] . "</td>";
        echo "<td>" . \$row["navn"] . "</td>";
        echo "</tr>";
    }
}
?>

</table>
</body>
</html>
```



Opgave: Ændringer til Grundstoffer

1. Ret i værdien af variablen `sql`, så resultatet sorteres ud fra `navn` i stedet for `nr`.
2. Ret i `sql`, løkken i PHP og HTML, så tabellen også viser egenskaben `symbol`.
3. Tilføj en forgrening inde i løkken, så celler for metaller (grundstof nr. 3 og 4) markeres med gul baggrundsfarve. Hint: Tilføj i HTML-tagget `td style="background: yellow"`.



Opgave: Lav en hjemmeside med udvalgte træsorter

Lav en hjemmeside, som viser egenskaber for udvalgte træsorter og henter sit indhold fra [denne tabel](#) ([Filten kan downloades fra ibogen se https://informatik.systime.dk/index.php?id=1099&L=0](https://informatik.systime.dk/index.php?id=1099&L=0)) (Ligger pakket i en zip-fil).

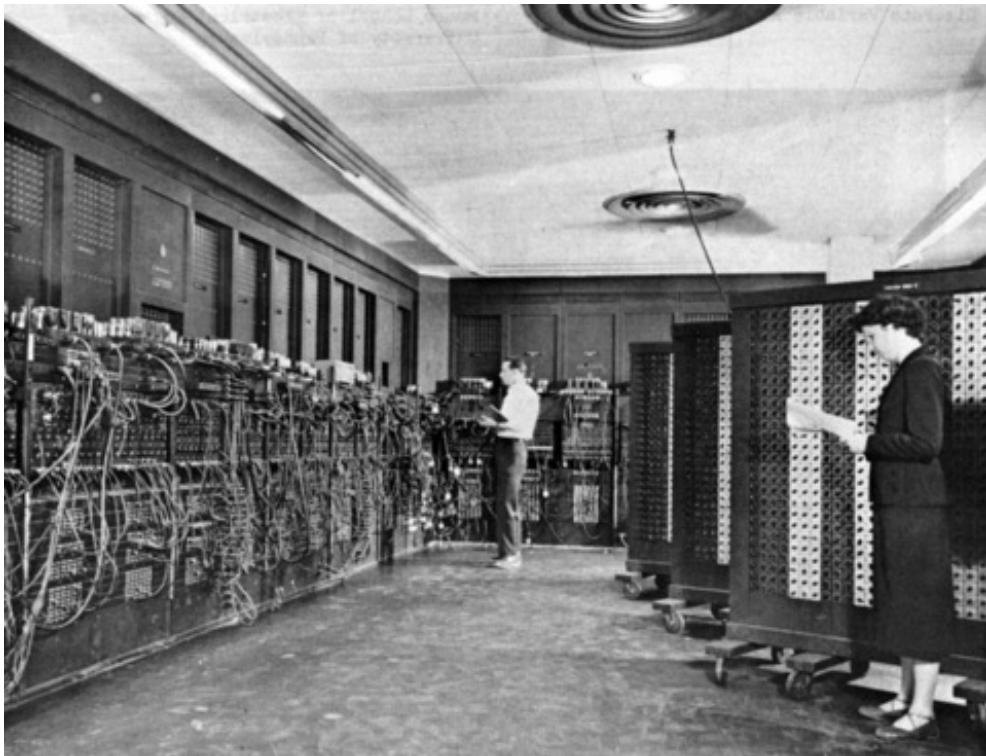
Hjælp:

- Importér tabellen i phpMyAdmin, og undersøg tabellens kolonner.
- Tag udgangspunkt i koden fra ovenstående eksempel. Lav en kopi af filen og ret i denne.
- Ret i variablen sql, så de relevante kolonner hentes fra den nye tabel.

Læs mere om databaseadgang i PHP på disse officielle sider.

| Kilde | Beskrivelse |
|---------------------------------------|----------------------------------|
| PDO::__construct | Skaber forbindelse til databasen |
| PDO::prepare | Forbereder en SQL-sætning |
| PDOStatement::execute | Udfører en forberedt SQL-sætning |
| PDOStatement::fetch | Henter næste række i et resultat |

IT-historie



ENIAC – den første elektroniske datamaskine.

U.S. Army Photo/Wikimedia Commons - Public Domain

Revolution betyder 'omvæltning', underforstået at der er tale om noget der går hurtigt. Og man kan roligt kalde den digitale udvikling i slutningen af 1990'erne for en revolution.

Som du kan se på listen nedenunder, begyndte computerens udvikling ganske vist alle rede under 2. Verdenskrig (1939-1945). Og længe inden var der lavet adskillige maskiner, der kan betragtes som forløbere for computeren. De var dog ikke elektriske, og man regner derfor Konrad Zuses *elektromekaniske Z3-maskine* fra 1941 som verdens første egentlige computer.

Den første *elektroniske* datamaskine var den amerikanske hærs *ENIAC* fra 1945. ENIAC står for Electronic Numerical Integrator and Calculator. Den blev bygget til at beregne, hvordan projektiler og raketter mv. opførte sig, når de var blevet affyret.

Den første danske datamaskine var *DASK* fra 1957. Navnet er en forkortelse for Dansk Aritmetisk Sekvens Kalkulator. Dele af den kan i dag ses på Teknisk Museum i Helsingør.

Disse første elektroniske datamaskiner fungerede ved hjælp af *radiorør* og *hulkort*. Radiorør kræver megen plads, og første-generations computerne var derfor meget store. De afgav også megen varme. Desuden havde radiorørene begrænset levetid, og hvis blot ét radiorør var 'sprunget', regnede maskinen forkert!

Transistorer, kredsløb og chips

I 1950'erne kom anden-generationsmaskinerne, der fungerede vha. *transistorer*. Regnecentralen i Danmark udviklede i 1962 en sådan transistormaskine, *GIER*, Geologisk Instituts Elektroniske Regnemaskine.

Transistorer er meget mindre – og i øvrigt også meget mere stabile – end radiorør, så transistormaskinerne fylde meget mindre end radiorørs-maskinerne.

Allerede i 1960'erne kom de første tredje-generationsmaskiner imidlertid; de fungerede vha. elektroniske kredsløb. Hermed blev det endnu en gang muligt at gøre maskinerne mindre.

Regnecentralens første maskine af denne type var *RC 4000* fra 1968. Det var dengang en af verdens mest avancerede computere. Hovedmand bag udviklingen var *Per Brinch Hansen*, der senere fra 1987 til sin død i 2007 var professor ved Syracuse-universitetet i New York.

RC4000 var flere meter lang og ca. 2 meter høj, men den havde alligevel kun en ganske lille del af den effektivitet som nutidens små bærbare computere har.

I 1970'erne begyndte man at producere fjerde-generationsmaskiner. Også de fungerer vha. kredsløb, men nu samlet på såkaldte chips, små firkantede plader af grundstoffet silicium. Her er kredsløbene præget ind i overfladen ved hjælp af en særlig fotografisk teknik. Og som bekendt er man siden blevet stadig dygtigere til gøre chipsene mindre – og alligevel gøre maskinernes ydeevne større.

Markedet var der, og det voksede hurtigt. Virksomheder med store skrive- og regneopgaver begyndte at købe maskinerne, ofte i form af såkaldte terminaler, der egentlig blot bestod af et tastatur, en skærm og en kabelforbindelse til en server, som var placeret på virksomhedens hovedkontor. Det var bl.a. den måde, banker og sparekasser blev digitaliserede på.

Hvad enten der var tale om selvstændige maskiner eller terminaler, havde de kun én opgave. Enten kunne de bruges til at skrive på, lave beregninger eller arbejde med registre. Eller de var computere ude på fabrikker, hvor de kunne udføre enkle rutineprægede arbejdsopgaver.



Der arbejdes med Piccoline fra Regnecentralen, ca. 1985. I den hvide kasse i baggrunden står disketterne, de var 8 tommer (godt 20 cm) store og temmelig sårbare. Bemærk også den lille skærm.

Kirsten Søndergård

Den personlige computer

Allerede omkring 1970 begyndte både forskellige firmaer og computerinteresserede opfindere imidlertid at eksperimentere med at bygge mindre og mere fleksible maskiner, bl.a. Apple 1 fra 1976.

I 1981 lancerede IBM deres PC 5150; det var den maskine, der kom til at danne standard for den personlige computer. I 1984 kom så Apples første *macintosh* – med grafisk brugergrænseflade.

Også i denne periode formåede det danske firma Regnecentralen at være med; firmaet byggede *Piccoloen* og *Piccolinen*, der især blev solgt til mange skoler, men også til erhvervslivet. I længden kunne Regnecentralen dog ikke følge med. Deres maskiner blev for dyre, og i 1993 lukkede firmaet.

Selvom computeren ret hurtigt blev udviklet fra de kæmpestøre radiorørsmaskiner til de små personlige computere, så gik den udvikling alligevel meget langsomt sammenlignet med udviklingen de sidste omkring 25 år. Og når vi taler om 'den digitale revolution', er det først og fremmest tiden fra omkring 1994, det handler om.

Det skyldes primært udviklingen af internettet.

Internettet

Det var med internettet, at den helt store revolution kom. Pludselig kunne man bruge sin personlige computer til andet end relativt almindelige kontoropgaver. Og da pc'erne efterhånden kom ned i pris, steg antallet af internetbrugere i løbet af år fra nærmest 0 til langt over halvdelen af befolkningen. Internettet bygger imidlertid ligesom også pc'erne på ældre opfindelser. Med internettet skal vi tilbage til 1960'erne hvor det amerikanske militær begyndte at forske i, hvordan man kunne opbygge et kommunikationsnetværk, der kunne fungere under en evt. krig med Sovjetunionen. Det var dette netværk, der efterhånden blev til internettet.

Der har været en række milepæle i udviklingen frem mod det moderne internet; her vil vi nøjes med at nævne Tim Berner-Lees http-protokol fra 1990 og Mosaic, den første browser, fra 1993. De gjorde det muligt at udvikle World Wide Web (www, på dansk ofte 'nettet') som en tjeneste på internettet. Og så gik det ellers stærkt!

Der er imidlertid også sket en stor ændring i den måde, man bruger nettet på. I de første år var brugen helt passiv; man kunne vælge – 'surfe' – mellem forskellige tilbud. Først var de alle gratis, typisk finansieret af reklamer, men hurtigt kom der også sider, man skulle betale for at bruge. I nettets første år var porno det vigtigste søgemål for brugerne; det spiller i dag en langt mindre rolle – og de kommercielle pornosider har i øvrigt fået stor konkurrence af almindelige menneskers egne billeder og film.

I det hele taget er der efterhånden kommet stadig mere interaktion på nettet. Og i dag bliver en meget stor del af indholdet produceret af 'almindelige' brugere, især brugere af de nye sociale medier. Og i dag er vi for længst nået til anden generation af internettet, det såkaldte www2.

www2 og de sociale medier

Ganske vist var der mulighed for interaktion på internettet længe før der var noget der hed www. Allerede i 1972 blev det første e-mail program udviklet. Det var dengang man fandt på at bruge @-tegnet til at adskille brugernavnet fra servernavnet. Det nye program blev for så vidt en stor succes, men de første mange år var antallet af brugere naturligvis meget lille.

En anden måde at kommunikere på er chat. De første chat-programmer blev udviklet først i 1990'erne.

I de seneste år har de såkaldt *sociale medier* fået meget stor succes. Det er medier, der bygger videre på chat-teknologien, men som samtidig har en række andre muligheder for udveksling af informationer af forskellig art. Især er spredning af billeder blevet populært.

Man kalder ofte disse nye medier for *web 2.0* for at markere, at hjemmesiderne her er dynamiske, og at det langt hen ad vejen er brugerne selv, der bestemmer, hvordan siderne ser ud. Også blogs – dvs. web-dagbøger – og et leksikon som *Wikipedia* er web 2.0-værktøjer.

De seneste led i udviklingen er for det første, at der kommer stadig flere forskellige slags personlige computere, og at grænsen mellem computere, smartfones og fjernsyn bliver flydende. Og for det andet at både programmer og lagre flytter ud 'i skyen', dvs. på store servere med offentlig adgang.



Vigtige begivenheder og personer i it-historie

Gå på nettet og søg viden om nedenstående vigtige begivenheder og personer i it-historiens udvikling.

1. Mobiltelefonens historiske udvikling
2. Internettets historiske udvikling
3. Historien om hackerangreb gennem tiden
4. Det amerikanske valg 2016 og et muligt hackerangrebs betydning for Trumps sejr
5. Historien om søgemaskiner og deres betydning for vores brug af internettet
6. Historien om Richard Stallman og hans betydning for den it-historiske udvikling
7. Historien om Bill Gates og hans betydning for den it-historiske udvikling
8. Historien om Steve Jobs og hans betydning for den it-historiske udvikling
9. Historien om Linux-projektet, Linus Torvalds og open source
10. Bill Gates brev "[An open Letter to hobbyists](#)" og problematikken omkring rettigheder
11. Den historiske udvikling af it-kriminalitet
12. Den historiske udvikling af hardware (Mainframe, PC, bærbar, mobiltelefon etc.)
13. Den historiske udvikling af browsere og søgemaskiner
14. Store danskere i it-historien

IT-sikkerhed



iStockphoto.com/vladwel

Sikkerhed er generelt en velfungerende *beskyttelse af værdier*, fx menneskeliv, natur, penge eller kunst. Inden for IT-sikkerhed er de værdier, der skal beskyttes, *information*, fx kreditkortnumre, kodeord, personfølsomme data eller oplysninger om et atomkraftværks IT-sikkerhedssystem. Uretmæssig omgang med eller øDELæggelse af information, der skal beskyttes, er et sikkerhedsbrud. Får en spion fat i statshemmeligheder, kan han videresælge dem til terrorister. Får en tyv fat i et kreditkortnummer, kan han hæve penge, der ikke er hans. Ødelagte konto-data i en bank kan resultere i en personlig fallit – eller at banken skal af med millioner af kroner.

Visse ideer fra traditionel fysisk beskyttelse af genstande kan oversættes til IT-sikkerhed (fx ideen om adgangskontrol). Der er imidlertid en lang række forskelle på traditionel sikkerhed og IT-sikkerhed:

- Hvis et maleri bliver stjålet, er det væk. Hvis en information aflyttes, fx et kreditkortnummer, har tyven blot en kopi. Man kan altså ikke umiddelbart vide, om ens kreditkortnummer er blevet stjålet, bare ved at tjekke, om man stadig har det.

- Modsat traditionel sikkerhed kan det være svært at forstå, *hvor* IT-sikkerhed skal beskytte – informationer er mere uhåndgribelige end malerier, guld eller fladskærme.
- IT-kriminalitet udøves typisk af en person, der fysisk er langt væk fra det der stjæles. Derfor kræver det teknisk indsigt at sikre sig.

Sikkerhed er helt afgørende for mange IT-systemer.

Eksempel: Systemer, hvor sikkerhed er kritisk

- Et usikkert kontrolsystem på et atomkraftværk, i et fly eller i en rumfælge kan skade mennesker, dyr eller miljø.
- En usikker hæveautomat kan tillade ondsindede personer at hæve andre folks penge.
- Et usikkert hospitalssystem kan udstille personfølsomme oplysninger som fx patientjournaler.
- En usikker typerialarm på et museum kan tillade tyveri af uerstattelige kunstværker.
- En usikker hjemmecomputer risikerer at blive inficeret med ondsindede programmer, aflyttet eller overtaget af hackere.

Fortrolighed, integritet og tilgængelighed

Normalt skelner man mellem 3 overordnede typer af informationsbeskyttelse:

- *Fortrolighed*: Kun personer og systemer, der er autoriseret til at læse informationen, har adgang til informationen.
- *Integritet*: Kun personer og systemer, der er autoriseret til at ændre eller slette informationen, har adgang til informationen.
- *Tilgængelighed*: Alle personer og systemer, der er autoriseret til at læse og/eller ændre informationen, kan komme til det.

Eksempel: Fortrolighed, integritet og tilgængelighed: Bankkonto

Et banksystem beskytter en kundes konto, herunder hendes kreditkortnummer og saldo.

Fortrolighed er, at kun hun og hendes bankrådgiver kender til kontooplysningerne – og ikke afslører det til tilfældige personer på gaden.

Integritet er, at hendes saldo er "som den bør være": Saldoen bliver kun ændret, hvis hun sætter penge ind eller hæver penge på kontoen, og saldoen bliver i så fald opdateret korrekt.

Tilgængelighed er, at hun kan hæve og indsætte penge – og på andre måder tilgå sin konto – når hun har ret til og brug for det.

Eksempel: Fortrolighed, integritet og tilgængelighed: Forretningshemmelighed

Et firma sælger hotdogs. Chefen udtaenker nu *cold dog'en*. Den kolde hotdog skal lanceres i firmaets pølsevogne inden for et år. Indtil da er opfindelsen en forretningshemmelighed. Firmates chef lægger en tekst-fil med en beskrivelse af cold dog'en på firmaets hjemmeside, så de ansatte kan lære, hvordan cold dog'en tilberedes.

Fortrolighed er, at kun firmaets ansatte kan læse beskrivelsen. Det skal altså sikres, at konkurrenten ikke opsnapper beskrivelsen.

Integritet er, at kun firmaets chef kan ændre i beskrivelsen. Da det er chefen, der har udtaenkt cold dog'en, skal ansatte forhindres i (med vilje, eller ved en fejl) at ændre beskrivelsen til noget forkert.

Tilgængelighed er, at alle firmaets ansatte faktisk kan komme til at læse beskrivelsen når som helst. Firmaets hjemmeside skal altså ikke gå ned, og den skal have et system, der sikrer alle medarbejdere adgang.

CIA-modellen

På engelsk forkortes confidentiality (fortrolighed), integrity (integritet) og availability (tilgængelighed) ofte til "CIA", og man taler om *CIA-modellen* for IT-sikkerhed.

Privacy



iStockphoto.com/Lightcome

Fortrolighed på det personlige plan kaldes *privacy* (engelsk for "privat-hed"). Der er mange ting fra privatsfæren, man som enkeltperson (eller familie) ikke ønsker udbasuneret til alle og enhver – fra politisk ståsted, religion, hårfarve, økonomisk status, sygdomme eller medicinforbrug til information om, hvor man befinner sig på hvilke tidspunkter, hvornår man står op og ligger ned, hvad man spiser til morgenmad eller hvad man skriver sms'er til sine kammerater om.

IT-systemernes stadige udbredelse i hverdagsslivet har gjort debatten om privacy aktuel: Hvilken information må hjemmesider lagre om deres besøgende? Må en virksomhed overvåge de ansattes brug af internettet? Må politiet aflytte borgeres internetbrug?

Selvom enkeltlante lovgiver omkring fx lagring af persondata, er der et stykke vej fra at håndhæve disse love til fx at kunne sige, at vi færdes anonymt på internettet:

- All internettrafik går gennem en internetudbyder, der ofte lagrer information om, hvilke computere, der har kontaktet hvilke andre computere via internettet og hvilke data, de kommunikerende computere har udvekslet
- Søgemaskiner husker søgninger fra de enkelte computere
- Internetforretninger husker købsinformation om kunder
- Sociale online-fora har personadresser mv. lagret

Men selvom vi stoler på vores internetudbyder og kun afslører information, vi ønsker at afsløre, til internetforretninger, søgemaskiner, mv. – er der stadigt langt til anonymitet: Blot det at sende en enkelt pakke på et netværk afslører information: En pakke fra computer A til computer B afslører fx, at A og B snakker sammen. Hverken A eller B er altså som udgangspunkt *anonyme* – og selve det faktum, at der er trafik imellem A og B, kan observeres. Det kunne sagtens tænkes, at A og B gerne ville snakke sammen, uden at nogen vidste, at der overhovedet fandt en samtale sted.

Brugere og hackere



iStockphoto.com/Lightcome

Et computersystem er ikke mere sikkert end de mennesker, der betjener sig af det – brugerne. De færreste brugere har kriminelle hensigter, men kan alligevel

- være skødesløse med kodeord eller andre følsomme oplysninger,
- lade sig narre.

Menneskelige fejl tegner sig for størsteparten af alle brud på sikkerheden i ethvert IT-system. En stor del af sådanne brud på sikkerheden sker, hvor naive eller fortravlede brugere afgiver følsomme oplysninger. Følsomme oplysninger afgives typisk på to måder:

- *Intetanende:* Det er svært at undgå at afgive oplysninger, hvis man færdes på internettet: Fx kan hjemmsider sagtens lokke en masse information om en internetsurfers computer ud af hans browser, og offentliggør man et dokument på internettet, kan det indeholde information om forfatterens computer og det netværk, den er tilkoblet.
- *Frivilligt:* Brugere narres til at udføre falske instruktioner – som at åbne en bestemt hjemmeside, åbne en vedhæftet fil i en email, angive et kodeord, eller lignende. Hjemmesiden eller den vedhæftede fil indeholder ondsindet kode, der udnytter de følsomme oplysninger, som brugerne har afgivet frivilligt. Brugerne lader sig narre, fordi instruktionerne tilsyneladende kommer fra fx IT-afdelingen i deres firma eller en ven fra adresselisten. En ondsindet aktør udgiver sig altså for at være en bekendt – det kaldes *social engineering*.

Der er ikke én bestemt type person, der skriver ondsindede programmer: Der er unge og gamle, professionelle og amatører, europæere og asiater, rige og fattige. Men *hvorfor* overhovedet skrive ondsindet kode? Her er nogle af de væsentligste svar på spørgsmålet:

- *Penge:* Computerbrugere verden over har i stigende grad brug for at logge ind på forskellige portaler og betale med kreditkort i netbutikker. Ved at stjæle kodeord eller kreditkortnumre kan ondsindede personer stjæle penge. De kan også videresælge information eller lave mere organiseret kriminalitet.
- *Politiske årsager:* Er man utilfreds med det politiske indhold på en hjemmeside, kan man jo bare gøre hjemmesiden utilgængelig eller ændre indholdet på den – så er den klaret! Det er ulovligt, men ikke desto mindre en almindelig type angreb.
- *Berømmelse:* Både unge og erfarne programmører kan ønske berømmelse eller anseelse – og forsøger at opnå det ved at skrive en effektiv computervirus (uheldigt!).
- *Forskning og udvikling af antivirus-software:* Professionelle programmører udfordrer konstant sikkerheden i eksisterende operativsystemer, browsere og andre programmer. Finder de et sikkerhedshul, sørger de for at lave sikkerheds-software, der kan lukke hullet. På den måde kan man være på forkant med de IT-kriminelle.

Hacker

En hacker er en person, der tiltvinger sig uretmæssig adgang til et IT-system, uden nødvendigvis at have ondt i sinde.



Opgave: Begreber

Gå på nettet og find definitioner af begreberne hacker, security hacker og cracker.

Kodeord og Adgangskontrol

Kodeord

Kodeord giver adgang til information, som kun kenderen af kodeordet bør have adgang til. Alligevel regnes kodeord ikke for en særlig sikker adgangskontrol. Mange brugere

- *bruger gætbare kodeord* som fx kærestens fødselsdag eller hundens navn.
- *bruger svage kodeord*, der kan knækkes på kort tid med en kodeordsknækker. Et kodeord regnes for svagt, hvis det er kort, gætbart, eller danner eksisterende ord. Fx er "?}_1", "Bastian" og "Jegerenfalk" svage kodeord.
- *skriver kodeord ned* for at huske dem.
- *afslører kodeordet* til ondsindede personer, der udgiver sig for at være en anden end de er (social engineering).
- *bruger samme kodeord flere steder*: De fleste skal huske en stor bunke kodeord til mange forskellige systemer – PIN-kode, kode til email, kode til arbejdscomputeren, mv. For at kunne huske kodeordene, bruges så vidt muligt det samme kodeord alle steder.

De bedste kodeord er genereret tilfældigt – på den måde kan de ikke sammenkædes med brugeren. Fx er kodeordene

"_t[71WLNiNx", "x?[3qD€ct(GnM" og "+(LaNqZ-/67-8"

genereret tilfældigt og meget lidt gætbare. Til gengæld er de svære at huske.

En bedre løsning kan være *smart cards*, der genererer et nyt, tilfældigt kodeord hver gang, kodeordet skal bruges. På den måde undgås genbrug af kodeord fuldstændigt.

Adgangskontrol

En måde at beskytte data på, er at adgangskontrollere den der ønsker adgang.

Her er to eksempler:

Eksempel: Brugernavn og kodeord

Mange brugersystemer tilgås gennem en adgangskontrol, hvor brugernavn og kodeord skal indtastes. Her består adgangskontrollen i

- identifikation = opgivelse af brugernavn,
- autenticitetsoplysning = opgivelse af kodeord,
- verifikation = tjek af, at brugernavnet er en gyldigt og er knyttet til kodeordet.

Hvis et ugyldigt brugernavn oplyses, vil adgangskontrollen nægte adgang med begründelsen "bruger ukendt".

Oplyses et korrekt brugernavn, men en forkert adgangskode, vil adgangskontrollen opfatte den oplyste identifikation som ikke-autentisk – fordi der er uoverenstemmelse mellem brugernavn og kodeord. I denne situation nægter adgangskontrollen adgang med begründelsen "kodeord forkert".

Eksempel: Biometrisk autentificering

Et IT-system kan bruge biometrisk autentificering af mennesker. Det betyder, at unikke biologiske egenskaber måles som autenticitetsoplysninger, når mennesket præsenterer sin identitet. De biologiske egenskaber, der måles, kan fx være øjets irismønster eller fingeraftryk.

Kommunikation over netværk

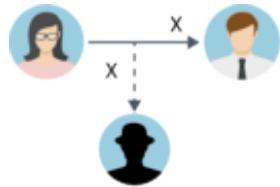
Netværkssikkerhed handler om at *kommunikere sikkert* over et netværk. Kommunikationen kan fx være udveksling af kæresterbreve mellem hemmelige elskende eller HTTP-forespørgsler til en webserver. Alt efter sammenhængen kræver sikker kommunikation mellem aktørerne A og B, at forskellige (eller alle) af følgende sikkerhedsmål er opfyldt:

fortrolighed: Uvedkommende kan ikke opsnappe kommunikationen.

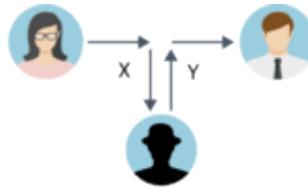
integritet:

- *data-integritet:* Indholdet af en besked fra A til B kan ikke ændres under transporten over netværket.
- *autenticitet:* En uvedkommende aktør C kan ikke udgive sig for at være A eller B.
- *uafviselighed:* Hvis A sender B en besked, kan A ikke bagefter nægte det.

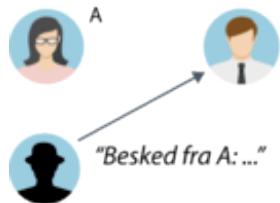
tilgængelighed: Uvedkommende kan ikke afbryde A's og B's mulighed for at kommunikere.



Aflytning:
Brud på fortrolighed



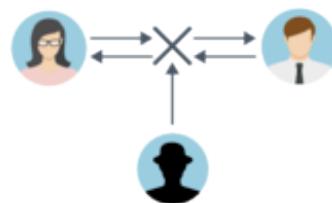
Ændring af besked:
Brud på data-integritet



Falsk afsender:
Brud på autencitet



Afsender adviserer at have sendt besked:
Brud på uafviselighed



Afbrydelse af forbindelse:
Brud på tilgængelighed

Trusler: Brud på fortrolighed, integritet og tilgængelighed.
iStockphoto.com/artvea / Systime

Der er en række måder at bryde sikkerheden på et netværk:

Packet sniffing

Pakker i netværkslaget kan blive aflyttet eller kopieret, uden at det bliver opdaget. Ondsinde aktører kan dermed opsnappe fortrolig kommunikation.

Spoofing og phishing

En ondsindet aktør udgiver sig for at være en anden, end han er. Man bruger som regel ordet spoofing om "aktivt" bedrag af denne art, fx email-henvendelser med falsk afsender, mens phishing bruges om mere "passivt" bedrag, fx at oprettholde en falsk netbanks-hjemmeside, der lokker folk til at indtaste kreditkortinformationer.

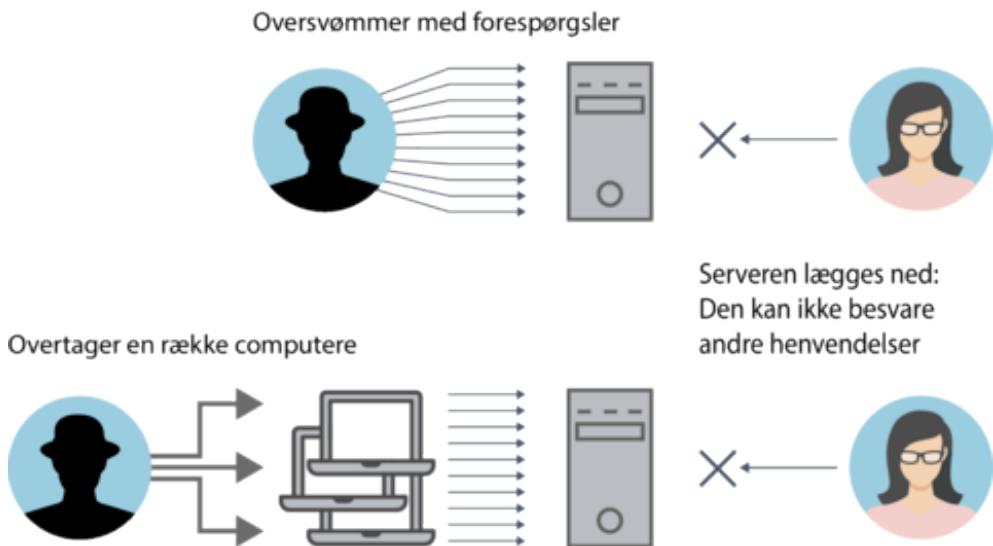
Replays

Et replay-angreb forklares bedst ved et eksempel: Lad os sige, at A er en netbankkunde og B en netbank. A henvender sig nu til netbanken og oplyser kodeord mv. Derefter foretager A de aktiviteter han skal i sin netbank. Til sidst logger A af netbanken. Hvad A og B ikke har opdaget, er at hackeren C har "optaget" A's logon-information til netbanken B – ved packet sniffing. Lidt senere "afspiller" C den del af "samtaLEN" mellem A og B, der bestod i, at A loggede på netbanken. Heraf navnet "replay" – C genafspiller dele af en kommunikation. Net-

banken B opfatter C's henvendelse som om, A henvender sig igen, og den giver adgang til A's konto. Resultatet er, at C har adgang til A's netbank.

DoS

Ondsindede aktører kan umuliggøre kommunikation ved fx at "lægge en webserver ned". Det gøres ved at genere webserveren med så mange henvendelser, at den ikke kan passe sit job over for andre klienter. Sådanne afbrydelser kaldes *Denial of Service* (DoS), fordi parter, der ønsker at kommunikere, nægtes adgang til det.



DoS og DDoS.
iStockphoto.com/fonikum/artvea / Systime

En særlig ondsindet type DoS-angreb er såkaldte *distribuerede DoS-angreb* (DDoS), hvor en hacker kaprer en række computere til at hjælpe sig med et målrettet angreb.

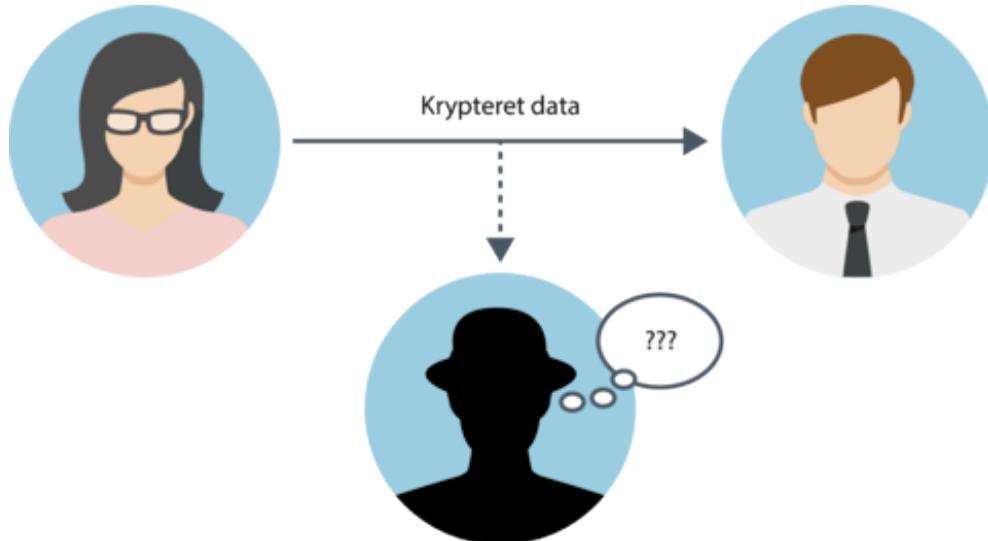
Trådløse netværk har ikke bare alle de samme sårbarheder som kabel-netværk, men også en lang række ekstra.

Trådløse netværk er *lette at opdage, aflytte og koble sig på* – modsat et kabelnetværk, som man kun koble sig på med et kabel, der er i fysisk i kontakt med en netværksenhed. En bygning med et trådløst netværk er i fare for at forbipasserende kan koble sig på.

Ondsindede trådløse netværk kan opstilles, fx nær eksisterende trådløse netværk. Hvis brugere ved en fejl benytter sig af det ondsindede netværk, fx i stedet for det netværk, de troede de brugte, kan de afsløre fortrolig information.

Kryptografi

Kryptografi er læren om at *kryptere data*. Krypteret data er tilsløret: Kun afsender og de modtagere, der ved, hvordan data kan *dekrypteres*, kan forstå indholdet. Bliver krypteret data opsnappet af en, der aflytter forbindelsen, får vedkommende ikke noget ud af det.



Kryptering.
iStockphoto.com/artvea / Systime

Eksempel: Cæsar-kryptering

Allerede den romerske kejser Julius Cæsar brugte kryptering, når han sendte taktiske ordrer med kurér mellem sine udsendte generaler. På den måde kunne hverken en spion, der udgav sig for at være kurér, eller overfaldsmænd, der bemægtigede sig den krypterede besked, afløre beskedten.

Cæsars teknik var simpel. Han valgte et tal, - f.eks. tre. Derefter erstattede han hvert bogstav i sin besked med bogstavet tre gange længere henne i alfabetet:

Før kryptering: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Efter kryptering: DEFGHIJKLMNOPQRSTUVWXYZABC

Hvis han ville kode teksten "Top secret!", ville han få den krypterede tekst "Wrs vhfuhw!". Uforståeligt – medmindre man kender krypteringsmåden.

Med kryptografi kan man opnå:

- fortrolighed,
- data-integritet,

- autenticitet af afsender- og modtager-identitet,
- uafviselighed.

Eksempel: Lønforhøjelse

Du er ansat i et firma og en dag får du en email fra chefen, der fortæller dig, at din løn stiger til 100.000 kroner om måneden. Fordi der er brugt kryptering, kan du være sikker på

- at ingen andre har læst emaileten, mens den rejste fra din chefs computer over netværket til din computer (fortrolighed)
- at emailens indhold ikke er blevet ændret, siden din chef afsendte den (data-integritet)
- at emaileten faktisk kom fra din chef (autenticitet)
- at din chef ikke kan nægte at have sendt emaileten, når han senere ombestemmer sig (uafviselighed).

Kryptering

Kryptering sikrer fortrolighed.

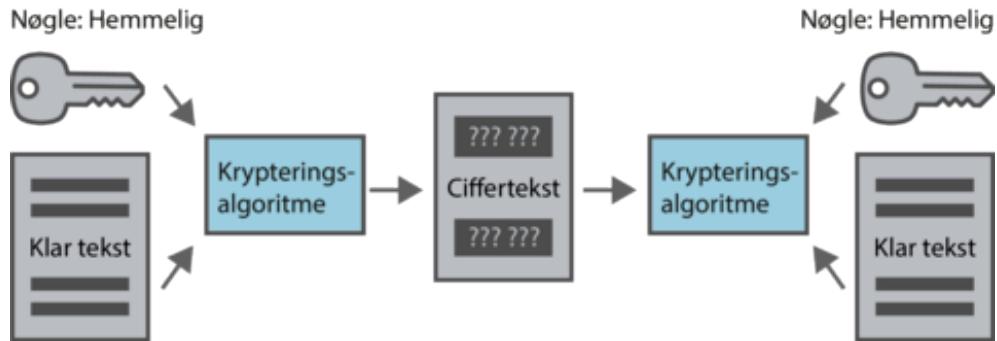
Til kryptering bruges en *krypteringsalgoritme*. Man bruger primært kryptering, der ikke afhænger af, at krypteringsalgoritmen er hemmelig. Dette kaldes for *Kerckhoffs princip*.

Hvis vi ser på eksemplet med Cæsar-kryptering fra før, kalder man beskeden "Top secret!" for klarteksen. Nøglen er tallet 3 og cifferteksten, altså den krypterede tekst, er "Wrs vhfuhw!".

Samlet set skal vi altså

- ved kryptering bruge klarteksten og nøglen til at få cifferteksen,
- ved dekryptering bruge cifferteksten og nøglen til at få klarteksten igen.

Vi kan antage, at krypteringsalgoritmen Cæsar-kryptering er kendt, men nøglen er i udgangspunktet ukendt.



Kryptering: Inden afsendelse krypteres klartekst til cifertekst med nøglen. Ved modtagelse dekrypteres ciferteksten med nøglen.

iStockphoto.com/fonikum / Systime

Symmetrisk kryptering

Symmetrisk kryptering er kryptering, hvor afsender og modtager skal bruge den samme nøgle: Afsender krypterer med nøglen, modtager dekrypterer med nøglen.

Cæsar-kryptering er *symmetrisk*.

Sikkerheden ved en symmetrisk krypteringsalgoritme afhænger af, at nøglen svær at gætte. Jo mere kompleks nøgle, desto sværere er det at knække koden. Cæsars algoritme har en simpel nøgle og er dermed let at gætte. Der er kun 29 forskellige nøgler (hvorfor?), og de er hurtigt gennemprøvet.

Eksempel: DES og AES fortrolighed

DES (Data Encryption Standard) er en symmetrisk krypteringsalgoritme, der blev lancet i 1977. DES bruger et 56-bit lang *binært tal* som nøgle. Der er altså 2^{56} , eller lidt over 70 millioner millarder, forskellige DES-nøgler. Man skulle tro, at DES dermed er ubrydelig, men et effektivt kryptoanalysehold dekrypterede en DES-besked på mindre end 4 måneder i 1997. Siden voksende computerkraften, og i 1999 behøvedes kun 22 timer for at dekryptere en DES-besked.

For at imødegå ondsindede personer med masser af computerkraft til at knække DES-krypterede beskeder, lancerede NIST (en amerikansk organisation for tekniske standarder) i 2001 efterfølgeren til DES: AES (Advanced Encryption Standard).

AES bruger nøgler med bitlængde 128, 192 eller 256. En maskine, der kan knække en DES-krypteret besked på 1 sekund, skal bruge omtrent 150 billioner år på at knække en besked, der er AES-krypteret med en 128-bits nøgle.

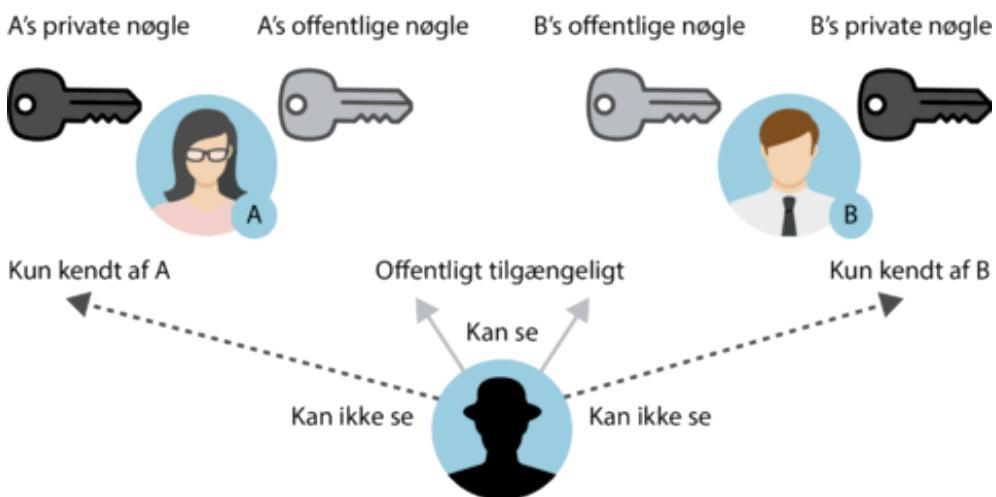
Symmetrisk kryptering forudsætter, at afsender og modtager i al hemmelighed har kunnet blive enige om, hvilken nøgle der skal bruges. Men det kræver jo (sikker) kommunikation!

Hvs man vil sende krypterede data over nettet skal afsender og modtager altså først på anden måde have aftalt og udvekslet nøglen.

Problemet løses ved *asymmetrisk kryptering* – en teknik, der blev opfundet i 1970'erne. Her behøver afsender og modtager ikke at have udvekslet en hemmelig nøgle for at kunne kommunikere sikkert.

Asymmetrisk kryptering

Kryptering, hvor afsender og modtager hver har sit eget sæt nøgler, en *privat nøgle* og en *offentlig nøgle*. Den private nøgle er hemmelig og kun kendt af ejermanden, mens alle har adgang til at se den offentlige nøgle. Data krypteret med den offentlige nøgle kan kun dekrypteres med den private nøgle.



Offentlig og privat nøgle.

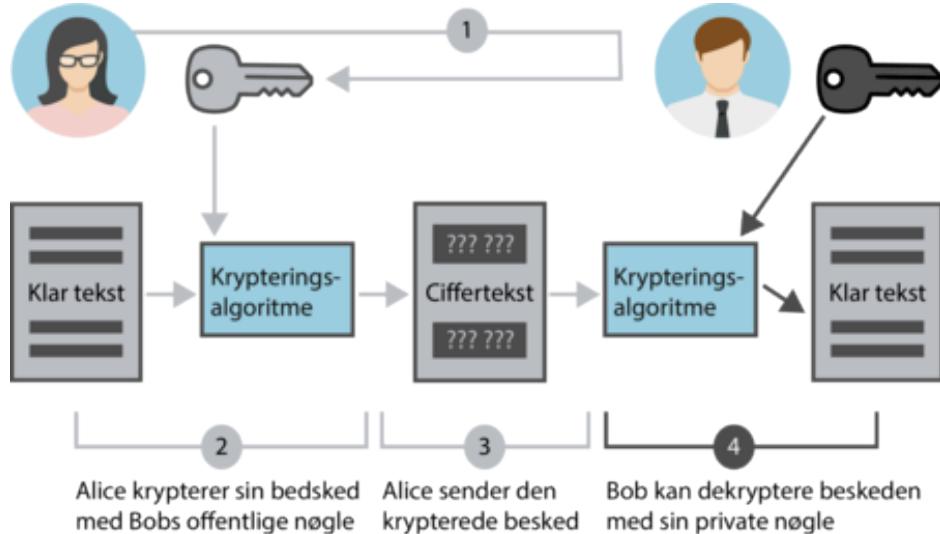
iStockphoto.com/fonikum/artvea / Systime

Da asymmetrisk kryptering hviler på brugen af offentlige nøgler, modsat symmetrisk kryptering, hvor der kun er hemmelige nøgler, kaldes asymmetrisk kryptering også for *public key-kryptering* (public betyder offentlig på engelsk).

Ved asymmetrisk kryptering, er der altså i alt 4 nøgler i spil (modsat symmetrisk kryptering, hvor der var 1 nøgle):

- A's private nøgle
- A's offentlige nøgle
- B's private nøgle
- B's offentlige nøgle

Eksempel: Asymmetrisk kryptering



Asymmetrisk kryptering med privat og offentlig nøgle.

iStockphoto.com/fonikum/artvea / Systime

Alice vil sende en besked, L, til Bob og hun ønsker at sende beskeden krypteret..

1. Hun henter derfor Bobs offentlige nøgle, Boboffentlig. Den kan alle hente, idet den er offentlig.
2. Alice krypterer sin besked, L, ved hjælp af Bobs nøgle, og får ciferteksten $\text{Bob}_{\text{offentlig}}(L)$.
3. Alice sender den krypterede besked til Bob.
4. Bob modtager $\text{Bob}_{\text{offentlig}}(L)$. Han kender, som den eneste, sin private nøgle, $\text{Bob}_{\text{privat}}$. Bob kan derfor dekryptere $\text{Bob}_{\text{offentlig}}(L)$. Det gør han ved at beregne $\text{Bob}_{\text{privat}}(\text{Bob}_{\text{offentlig}}(L)) = L$.

Hvis Bob beslutter sig for at svare, bruger han den helt tilsvarende procedure: Han henter først Alices offentlige nøgle, krypterer sit svar med den, og sender den krypterede besked til Alice. Fordi Alice kender sin private nøgle, kan hun dekryptere Bobs besked.

Analogi: Aflåste kister

Dronning Alice af Alicien vil gerne sende et hemmeligt brev til Kong Bob af Bobbistan. Desværre har dronning Alice kun en mistænkelig kurér, Igor, til sin rådighed. Igor kunne sagtens finde på selv at læse brevet undervejs eller udlevere det til fremmede.

1. Hun sender derfor Igor til Bobbistan for at hente Kong Bobs hængelås. Igor adlyder. I Bobbistan udleverer Kong Bob sin hængelås (åben). Hængelåsen er ikke i sig selv værdifuld, så Igor stjæler den ikke på hjemvejen.
2. Hjemme i Alicien putter Dronning Alice brevet ned i en kiste, spænder jernkæder om kisten og låser kæderne fast med Kong Bobs hængelås.
3. Dronning Alice sender Igor afsted med den aflåste kiste. Igor kan ikke åbne kisten og undlader derfor at stjæle den – det ville han alligevel ikke få noget ud af. Møder han mistænkelige fremmede på rejsen, vil de heller ikke kunne åbne kisten og se brevet derinde.
4. Vel fremme i Bobbistan modtager Kong Bob kisten. Efter at have sendt Igor hjemad, fremdrager Kong Bob sin nøgle, sætter den i hængelåsen – den passer! – og låser op.

Brevet er nået sikkert frem.

Fortællingen kan bruges til at forstå asymmetrisk kryptering med:

- **Hemmeligt brev** ≈ besked, Alice vil sende til Bob
- **Kong Bobs hængelås** ≈ Bobs offentlige nøgle
- **Kiste låst med Kong Bobs hængelås** ≈ besked krypteret med Bobs offentlige nøgle
- **Kong Bobs nøgle** ≈ Bobs private nøgle
- **Igor** ≈ transport over et netværk, der måske bliver aflyttet

RSA er en asymmetrisk krypteringsalgoritme (opkaldt efter opfinderne Ron Rivest, Adi Shamir og Leonard Adleman). RSA er en globalt accepteret standard for kryptering.

Et eksempel på en anvendelse af asymmetrisk kryptering er HTTPS. HTTPS er en protokol, der bruges, når man ønsker, at de data, der sendes over nettet, krypteres. Dermed er data fortrolige. Man kan se, at der anvendes HTTPS ved at kigge i URL'en i adressefeltet på browseren.



Opgave: RSA-kryptering på nettet.

Gå på nedenstående hjemmeside og prøv at lave en RSA-kryptering:

[RSA Calculator](#)

Hashing

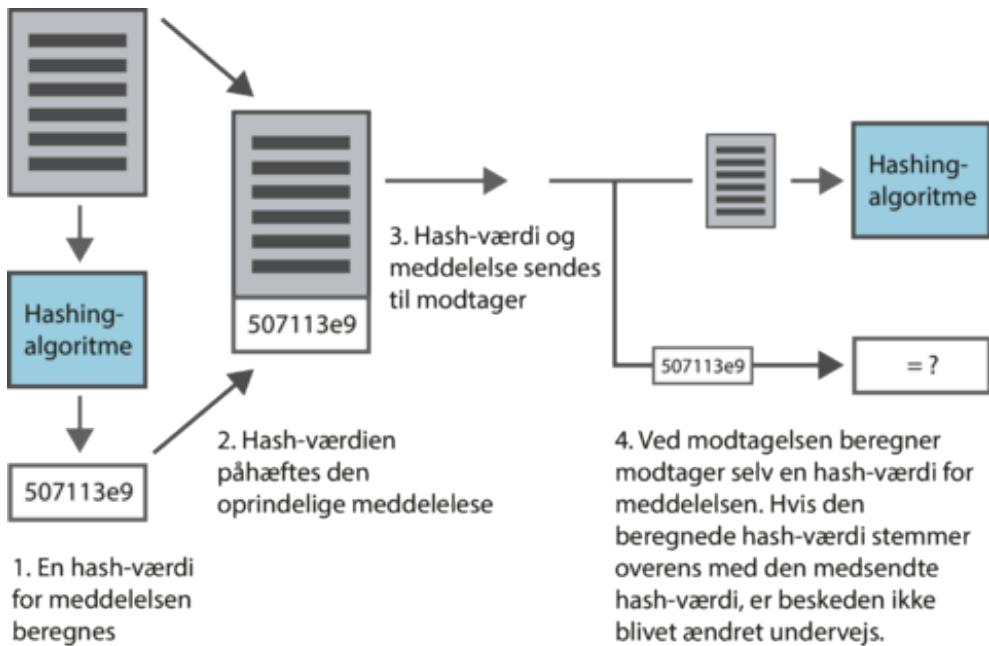
Hashing bruges til at sikre data-integritet af beskeder. Dvs. hashing sikrer at bliver indholdet i beskeden ændret, vil det blive opdaget.

Hash-funktion

Til hashing bruges en *hash-funktion*. Hash-funktionen er en algoritme, der tager en besked (kort eller lang) som input og leverer en hash-værdi af en bestemt størrelse som output. En hash-funktion beregner altid samme output-hash-værdi for den samme input-besked. En hash-funktion er *en-vejs*: Ud fra en hash-værdi kan den oprindelige besked ikke genskabes.

Eksempel: Hashing

Alexander vil sende beskeden "Du er sød! Alexander" til Malene over et usikkert netværk. For at sikre sig, at Malene ikke modtager en forkert besked, hasher Alexander beskeden med en hash-funktion og beregner hash-værdien "507113e9". Alexander sender nu både besked og hash-værdi til Malene. Beskeden når frem til Malene, og hun beregner hash-værdien af den modtagne besked. Hvis hun får hash-værdien "507113e9", kan hun sammenligne med Alexanders medsendte hash-værdi og konstatere, at beskeden er nået uændret frem. Hvis hun derimod får en anden hash-værdi, må beskeden være blevet ændret under sin rejse på netværket.



Hashing.

Hashing bruger ingen nøgler. Hashing-algoritmen er heller ikke hemmelig – i overensstemmelse med Kerckhoffs princip. Fidusen med hashing er, at det er en en-vejs-proces.

Opsnapper en ondsindet aktør en hash-værdi, fx "aac16ea6", *kan han ikke genskabe den besked*, der gav hash-værdien.

Der er altså en afgørende forskel på, hvordan kryptering med nøgler fungerer, og så hashing. Med den rette nøgle kan man dekryptere en krypteret besked. Ingen kan nogensinde "dekryptere" en hash-værdi og få den originale besked tilbage.

Kollisionsfri hash-funktioner

En god hash-funktion er *kollisionsfri* – dvs. at forelagt en besked og dens hash-værdi er det *umuligt* at finde en anden besked med samme hashværdi.

Digital signatur

En *digital signatur* er en elektronisk underskrift. Den sikrer, at den sendte meddelelse ikke er ændret, at underskriveren er den han siger han er og at underskriveren ikke kan nægte, at han har sendt meddelelsen. Dvs en digital signatur sikrer dataintegritet, autencitet og uafviselighed.

Eksempel: Brug af digital signatur

Digitale signaturer bruges overalt, hvor der indgås kontrakter eller økonomiske aftaler over netværk, fx

- mellem personer og banker
- mellem banker og banker
- mellem virksomheder

Inden vi ser på, *hvordan* man laver en digital signatur, kigger vi på et eksempel, der illustrerer, *hvorfor* digitale signaturer er nødvendige. Hashing-teknikker alene er nemlig ikke nok til at sikre besked-autenticitet:

Eksempel: Hashing uden autentificering

Alexander sender (besked,hash-værdi)-parret

("Du er sød! Alexander", "507113e9")

til Malene over et usikkert netværk. Erika, der er jaloux, opsnapper både besked og hash-værdi. Hun ændrer beskeden til "Jeg hader dig, Malene. Alexander" og beregner en ny hash-værdi, "029a159e", svarende til den nye besked. Erika sender nu (besked,hash-værdi)-parret

("Jeg hader dig, Malene. Alexander", "029a159e")

til Malene. Malene modtager beskeden, beregner hash-værdien "029a159e" og tror nu – fordi hash-værdierne stemmer overens – at beskeden ikke er blevet ændret undervejs.

Vi har altså ikke sikret fortrolighed.

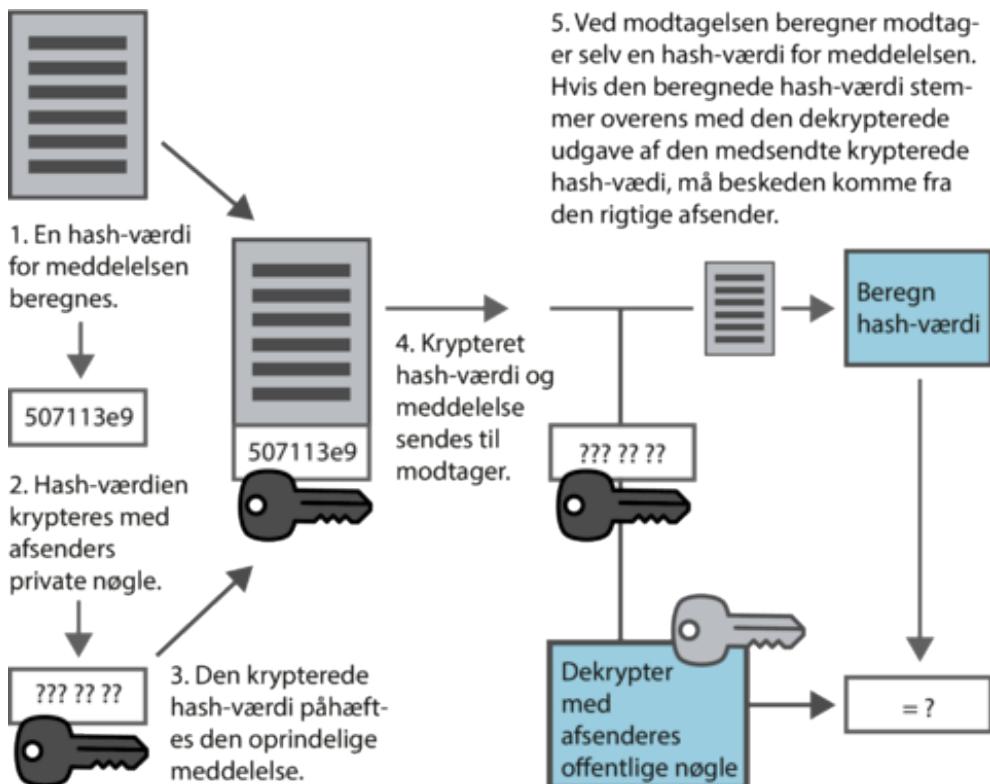
Vil man sikre sig, at beskedens afsender er den rigtige, må *digitale signaturer* tages i brug.

En digital signatur implementeres med en kombination af asymmetrisk kryptografi og hashing:

Digital signatur

En digital signatur på en besked er en hash-værdi af beskeden, krypteret med afsenderens private nøgle.

Ideen er, at hvis man kan dekryptere hash-værdien med en eller andens offentlige nøgle, så må den "en eller anden" lige præcis være afsenderen. Signeringen sikrer på den måde autentificering og uafviselighed. Hashingen sikrer data-integritet.



Digital signatur.
iStockphoto.com/fonikum/ Systime

Vi illustrerer ideen med et eksempel.

Eksempel: Digital signatur

Alexander vil sende beskeden "Du er sød! Alexander" til Malene. For at sikre dataintegritet og for at Malene ved, at beskeden er fra ham, gør Alexander følgende:

("Du er sød! Alexander", "bdb74befc887188")

til Malene.

1. Alexander beregner hash-værdien "507113e9" af beskeden "Du er sød! Alexander".
2. Han krypterer hash-værdien "507113e9" med sin private nøgle og får den krypterede hash-værdi "bdb74befc887188". Kun Alexanders offentlige nøgle kan dekryptere den krypterede hash-værdi.
3. Alexander sender (besked, krypteret hash-værdi)-parret
4. Malene beregner først hash-værdien af den modtagne besked – hun får hash-værdien "507113e9", fordi en hash-funktion altid giver den samme hash-værdi for den samme input-besked.
5. Malene dekrypterer herefter den modtagne krypterede hash-værdi "bdb74befc887188" med Alexanders offentlige nøgle. Det giver hende værdien "507113e9".
6. Fordi den modtagne og den beregnede hash-værdi er ens, er beskeden ikke blevet ændret undervejs. Fordi Alexanders offentlige nøgle kunne bruges til at dekryptere med, må beskeden være fra ham.

Kryptografisk opsummering

Vi opsummerer alle de mange kryptografiske begreber.

| | Fortrolighed | Data-integritet | Autenticitet | Uafviselighed |
|--|--------------|-----------------|--------------|---------------|
| Kryptering sikrer | X | - | - | - |
| Hashing sikrer | - | X | - | - |
| Digital signatur sikrer | - | X | X | X |
| Kryptering og digital signatur sikrer | X | X | X | X |

Ikke alle kryptografiske teknikker sikrer alle IT-sikkerhedsmål. Med de rette kombinationer af kryptografiske teknikker kan man opnå de sikkerhedsmål, en given beskedudveksling kræver.

Antivirus-software og Firewalls

To generelle måder at beskyttes sine data på er at bruge antivirus-software og firewalls.

Et antivirus-program kan opdage malware, inden det når at inficere en computer i et IT-system – eller kurere systemet, hvis det er blevet inficeret. Malwaren uskadeliggøres eller, bedre, fjernes.

Sådanne programmer har to primære metoder til at opdage malware:

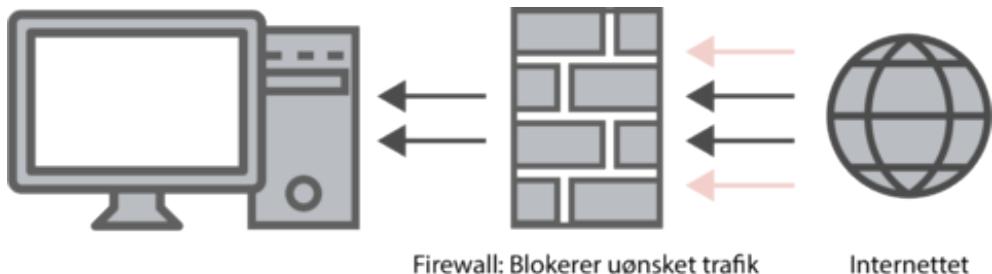
- *Scan efter malware:* Systemet scannes jævnligt efter mønstre, der matcher virusdefinitioner i en opdateret oversigt over kendt malware.
- *Overvåg systemet:* Hvis mistænkelige begivenheder indtræffer - hvis fx filstørrelser ændres uden grund, eller programmer holder op med at virke.

Hvis en mistænkelig fil eller mistænkelig program-opførsel opdages, sørger antivirus-programmet for at uskadeliggøre filen eller programmet – eller alarmere brugeren.

En anden måde at beskytte sine data på er at forhindre aktører udefra i overhovedet at få adgang. Ligesom man i middelalderen beskyttede borge med en voldgrav og en vindebro, beskytter man nu om dage IT-systemer med *firewalls*.

Firewall

En firewall er en kombination af hardware og software, der afgrænsler et internet-opkoblet IT-system fra resten af internettet. Firewallen tillader noget trafik at passere og blokerer for anden trafik.



Firewall.

iStockphoto.com/fonikum / Systime

Ideen med en firewall er, at den kun lukker "velkommen" trafik ind i IT-systemet – det kaldes *filtrering*, fordi trafikken sluses gennem et filter. En firewall frasorterer en stor del af den uønskede trafik, men kan aldrig give fuldstændig beskyttelse.

Firewalls kan filtrere på flere måder:

- *pakkefiltrering*: Firewall'en inspicerer de enkelte ankommende pakker. Hvis en pakke kommer fra en tvivlsom adresse, har et ukendt format, genkendes som skadelig, eller vil kontakte en port, der ikke ellers bruges, kasseres pakken.
- *beskedfiltrering*: Det kan være nødvendigt at filtrere trafik på basis af fx enkeltbrugeres identiteter.

Opgaver til it-sikkerhed



Opgave: Kodeord

Jonas skal vælge et kodeord til sin netbank. Han vil vælge et af følgende:

1. makrel
2. hotfyr19
3. _aA(!0
4. J9&.a!2wFop4%
5. 5j3*G~iyIIG6nC

- Hvilket kodeord er sikrest? Brug en online-kodeordstjekker til at afgøre styrken af hvert enkelt kodeord.
- Er kodeordstjekkere enige om, hvor stærke kodeordene er?
- Hvilen faktor er mest afgørende for et sikkert kodeord?



Opgave: Fortrolighed og integritet

- Forklar forskellen på fortrolighed og integritet. Kan man have fortrolighed uden integritet? Integritet uden fortrolighed? Forklar.



Opgave: Autorisation

- Forklar begreberne identifikation, autentificering, autenticitet, verifikation, autorisation.
- Hvad er forskellen på autentificering og identifikation?



Opgave: Krav til digital signatur

- Hvilke tre krav skal en digital signatur opfylde?



Opgave: Trusler mod en virksomhed

Hvilken gruppe udgør generelt den største IT-sikkerhedstrussel for en virksomhed:

- Virksomhedens ansatte?
- Hackere?



Opgave: Autentificering

- Hvilke oplysninger udgør identifikation, autenticitetsoplysning og verifikation, når du vil hæve penge med dit hævekort i en hæveautomat?



Opgave: Beskyt din profil

En måde at beskytte sine data på er at gemme sig bag et alias – altså et dæknavn.

- Hvorfor kan et alias beskytte dig og dine data?
- Find et sted på nettet hvor du kan få genereret et falsk alias og afprøv det.
(et forslag er fakenamegenerator.com)



Opgave: Søgemaskiner der ikke tracker

Når du søger på nettet, bliver din færden tracket, altså sporet, så andre kan se, hvor du har været.

- Find en søgemaskine, der lover ikke at tracke din færden.
(Et eksempel kan være duckduckgo.com/)
- Find et værktøj til din browser, der fortæller hvem der tracker din færden.
(Et eksempel kan være ghostery.com/)



Opgave: It-sikkerhedsmål vs. kryptografi

- Forklar, hvordan krypteringsteknikker kan anvendes til at opnå følgende sikkerhedsmål:
 - fortrolighed
 - data-integritet
 - autenticitet af afsender- og modtager-identitet
 - uafviselighed
- Forklar, hvorfor kryptering ikke kan bruges til at opnå sikkerhedsmålet "tilgængelighed".



Opgave: HTTPS

Hvor bruges protokollen HTTPS?

- Find tre steder på nettet hvor der bruges HTTPS.
- Hvorfor er der brugt kryptering her?
- Hvad er det for data man ønsker at beskytte?



Opgave: SQL injection

- Undersøg på internettet, hvad en *SQL injection* er. Identificér aktører, trusler og sårbarheder.
- Hvilke modmidler findes der?



Opgave: Honeypots

- Inden for IT-sikkerhed, hvad er en så honeypot? Hvad bruges en honeypot til?
- Find en beskrivelse af et honeypot-setup på internettet.



Opgave: Hoaxes

Hoax er engelsk og betyder "fupnummer". I computersammenhænge er en hoax en falsk alarm, fx omkring virus og malware. Sådanne falske alarmer spredes som regel af uvidende brugere via internettet.

- Find, vha. internettet, 2 eksempler på hoaxes.
- Forklar, hvilke IT-sikkerhedsmæssige problemer en falsk alarm skaber.



Opgave: Kreditkortsvindel

- Undersøg, hvordan betaling med kreditkort på nettet foregår. Redegør for sikkerheden i betaling med kreditkort på nettet.
- Forklar, hvordan kreditkortsvindlere bryder sikkerheden omkring elektroniske kreditkortbetalinger.



Opgave: Kerckhoffs princip

- Begrund Kerckhoffs princip.
- Bruges kryptering med hemmelige algoritmer nogen steder? Hvis ja, hvor? Hvorfor tror du, at man har valgt at bryde Kerckhoffs princip?

Klient-server arkitektur

Når vi bruger vores mobiltelefoner, pc'er og lignende, foregår meget af arbejdet med hjælp fra internettet.



Opgave: mobiltelefon uden internet

Sluk for internetforbilledelsen på din mobiltelefon, og find 5 apps, der ikke fungerer uden netadgang. Find derefter 5 apps, der fungerer fint uden internetadgang.

Hvad er det, der gør forskellen?

Når man bruger et it-system, er det ikke altid, at alt materialet ligger på den enhed, som man bruger. Somme tider ligger noget på enheden og noget på en server.

I sådan en situation er en klient et brugerprogram, der kører på fx en mobiltelefon eller en PC, mens en server er en internet-tilkoblet maskine, der altid er tændt og klar til at modtage forespørgsler fra klienter.



Opgave: Klient og server

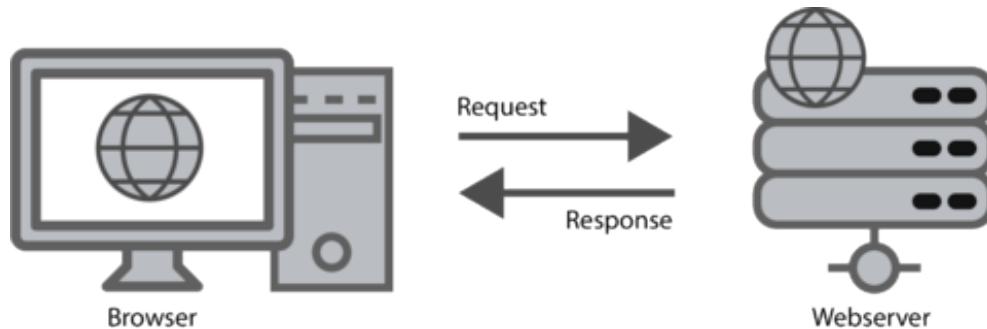
Hvilke af nedenstående it-systemer har brug for adgang til en server?

- Word på pc
- browser på pc
- spil på pc
- gps i bil
- ur i bil

Lad os se på et konkret eksempel på et it-system, der både bruger en klient og en server:

Vi vil åbne hjemmesiden med adressen systime.dk på en pc. Browseren på pc'en er klienten. Hjemmesiden ligger på en server. Vi indtaster adressen i browseren, der så kontakter den hjemmeside-server, som svarer til adressen.

Når browseren forespørger indholdet af systime.dk, sørger serveren for at udlevere indholdet. Denne enkle to-trins-kommunikation består af forespørgsel (*request*) og svar (*response*).



Request og response.

iStockphoto.com/fonikum / Systime

Klient-Server arkitektur

En klient-server arkitektur er en model med to aktører: *klient* og *server*. Klienten forespørger tjenester hos serveren. Serveren svarer ved at levere de forespurgte services. Klienten er aktiv – den, der tager initiativ til en forespørgsel. Serveren er passiv – den svarer blot på forespørgsler.

Servertyper

Ofte er servere reserveret til at arbejde med en bestemt kategori af indhold.

- F.eks. tilbyder en webserver adgang til webindhold. Dens klienter er typisk browsere.
- Spilserveren tilbyder adgang til at spille et online-spil. Dens klienter er spilprogrammer.
- Printserveren tilbyder adgang til en eller flere printere. Dens klienter er tit tekstdrivers, der beder om at få printet dokumenter.
- Filserveren tilbyder adgang til en samling filer. Dens klienter er særlige programmer, der kan hente filerne til brugerens computer.

Andre arkitekture

Klient-server arkitekturen er en meget simpel arkitektur, der er god til små it-systemer.

Store it-systemer som f.eks. store onlinespil, har en ekstra inddeling af serveren, så der bliver tale om en trelags arkitektur.

Kommunikation



iStockphoto.com/LindaYolanda

Hvad sker der, når vi kommunikerer? Og hvordan kan man beskrive en kommunikation?

Der er en afsender, der afsender et signal og en modtager, der modtager signalet. Signalet skal have et indhold.

Hvis der er tale om f.eks. en reklame, er det ikke nok at modtageren modtager signalet, han skal også reagere på signalet.



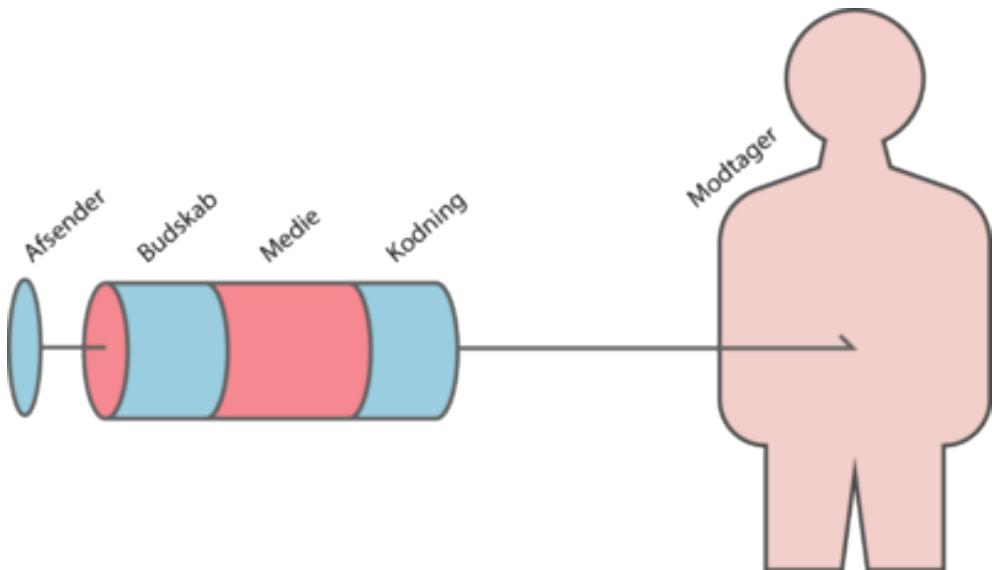
Der er forskellige modeller

Vi skal se på forskellige kommunikationsmodeller. Umiddelbart er de dog ikke voldsomt forskellige – de indeholder alle en *afsender*, et *signal* og en eller flere *modtagere*. Men den betydning, man tillægger de forskellige elementer, er ret forskellig.

I de første modeller anså man afsenderen for det vigtigste element, fordi det er den pågældende, der er den aktive. Men man er efterhånden blevet mere opmærksom på, at det ikke er nok blot at udsende signaler. Og for nye medieforskere er det nærmest modtageren, der er den vigtigste, fordi det er ham/hende, der – bevidst eller ubevist – afgør, hvilken skæbne signalet får.

De første kommunikationsforskere så også kun på selve kommunikationen, mens nyere forskere også ser på den sammenhæng, som kommunikationen foregår i – herunder den eventuelle *støj*, som på forskellig måde kan påvirke kommunikationen. Som vi skal se, kan støj være mange forskellige ting.

Laswells kommunikationsmodel



Laswells kommunikationsmodel.

Den første kommunikationsmodel blev lavet i 1930'erne af amerikaneren *Harold D. Laswell*. Det var en meget enkel model, med nøje sammenhæng mellem afsenderens budskab og kommunikationens effekt.

Laswell udviklede senere (i 1948) denne model til den såkaldte *Kanylemodel*, også kaldet *De fem H'er* (Hvem siger Hvad i Hvilkens kanal til Hvem og med Hvilkens effekt). Laswell mente, at hvis blot afsenderen gjorde sit arbejde godt nok, kunne han/hun nærmest 'sprøjte' sit budskab direkte ind i modtageren.

Laswells model kan virke naiv i dag, hvor vi er vant til reklamer – og også vant til at *overse* dem – men da Laswell udarbejdede sin kommunikationsteori, fylde reklamer meget mindre end i dag. Det var derfor ikke så underligt, at Laswell kunne få den tanke, at blot man valgte det rigtige medie – og den rigtige placering i mediet – så skulle budskabet nok trænge ind.

Laswells største fejtagelse var nok, at han brugte sine analyser af propaganda under Første Verdenskrig til at opstille en *generel* teori for kommunikation. En krigssituation er en meget speciel situation, og den til enhver tid siddende regering og præsident har i den situation en enorm magt over befolkningen.

Reklamer i dag

Man kan også i dag se, at store effektive politiske kampanjer kan flytte mange stemmer – især i lande uden lange demokratiske traditioner, men efterhånden gennemskuer vælgerne og forbrugerne de forskellige reklametricks, og så virker de ikke længere. I dag er reklamebranchen generelt gået bort fra tanken om, at bare man skruer sine reklamer godt nok sammen, så virker de efter hensigten.

Imidlertid har reklamebranchen ikke andre muligheder end at målrette budskaberne bedst muligt, så de fortsætter med at sprøjte. Det virker da også nogenlunde, bl.a. fordi reklamerne gentages mange gange. Men stadig flere forbrugere bliver ikke kun immune; de bliver ligefrem reklame-*hadere*. Der var således i 2017 omkring 48 % af de danske husstande, der havde sat mærket med 'Reklamer, Nej tak' på deres postkasse.

Holdningsbearbejdende kampagner

Men normalt bruger man udtrykket 'holdningsbearbejdende' om de kampagner og reklamer, der prøver at ændre vores holdning – og *adfærd* – i en retning som det offentlige finder passende. Det kan fx være at lade være med at køre spritkørsel, holde op med at ryge eller huske kun at dyrke sikker sex.

Disse kampagner er meget omdiskuterede, og det er sjældent, at man umiddelbart kan se nogen større effekt af dem. Men nogle af dem har dog virket på længere sigt, det gælder bl.a. netop holdningen til spritkørsel og rygning. Men det er ikke den enkelte reklame eller kampagne, der har virket; det er snarere, at de gennem en lang periode har påvirket holdningen i befolkningen.

Eksempel på kampagne – Stop før 5

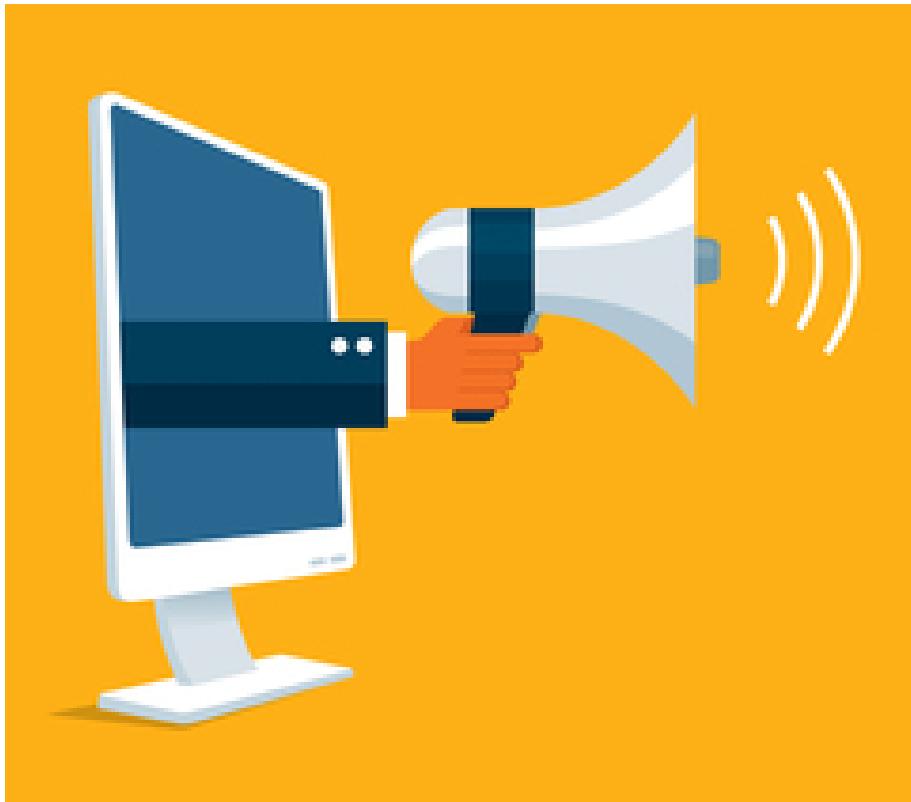
Du kan se et eksempel på sundhedsstyrelsens alkoholkampagne [Stopfør5](#).



Opgave: Reklamer og kampagner

1. Ser du reklamer på internettet – eller overser du dem?
2. Diskutér, hvilke fordele og hvilke ulemper der er ved reklamer på internettet.
3. Hvad mener du om de forskellige holdningsbearbejdende kampagner, der kommer fra det offentlige? Er der nogen af dem, der har påvirket dig?

Støj og kommunikation



iStockphoto.com/sorbetto

Der kan være flere grunde til, at en meddelelse ikke får den effekt, afsenderen har tænkt. Det kan være, at afsenderen har valgt et forkert medie. Det kan også være, at modtageren af en eller anden grund ikke forholder sig – evt. ikke *wil* forholde sig – til meddelelsen. Det kan også være, at meddelelsen forstyrres af støj.

Kommunikation har altid været generet af støj; der kan være støj i en radio eller en telefon, men støj er ikke kun noget, der knaser. Alt, hvad der forstyrrer en kommunikation, er støj.

Ofte kan man se, hvordan især uerfarne brugere har valgt at bruge alt for mange effekter i deres kommunikation. Det kan fx være præsentationer, der er fyldt med bevægelige figurer – og måske også lyd – eller det kan være tekster, hvor der bliver brugt alt for mange forskellige skrifter og skriftstørrelser. Det giver et amatøragtigt indtryk og kan desuden let føre til, at modtagerne bliver mere optaget af effekterne end af indholdet.

Støj fra reklamer – og omvendt

Computeren har ført til en ny slags støj – nemlig den støj der kan opstå når to elementer der placeret tæt ved hinanden, kolliderer. Det kan fx være en reklame for et kaffe-produkt,

der er placeret ved siden af en artikel, som handler om helbredsrisikoen ved at drikke for meget kaffe.

Du kan se et eksempel på en sådan kollision [her](#)



Opgave: Kollision

Find eksempler på kollision af reklamer og tekster på internettet.



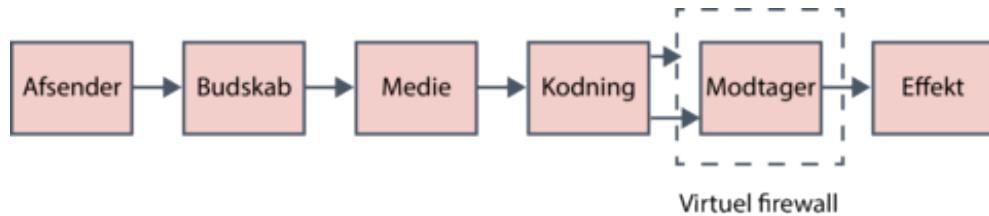
Opgave: Støj

Overvej, hvor du af og til oplever støj i forbindelse med kommunikation. Og husk, at støj kan være alt muligt der forstyrrer modtagelsen.

1. Kan du komme med eksempler på støj fra afsenderen?
2. Kan du komme med eksempler på støj der skyldes mediet?
3. Kan du komme med eksempler på støj der skyldes modtageren?
4. Kan du komme med andre eksempler på støj?

Moderne kommunikations-modeller

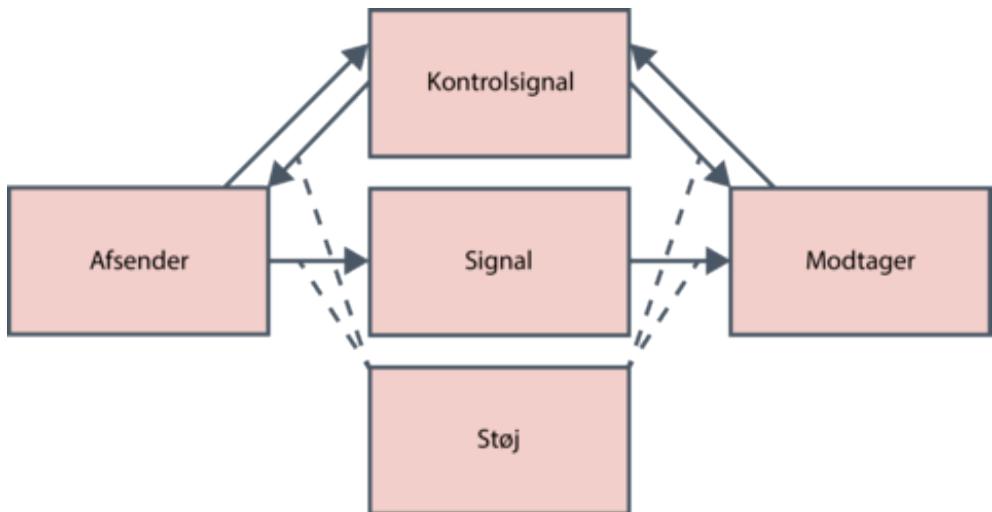
Vor tids modtagere er udstyret med en 'virtuel firewall', der hindrer at uønsket kommunikation kun vanskeligt kommer igennem. I nedenstående model kan man se, er grundlaget stadig afsender > signal > modtager, men modtageren er ikke bare et led i en kæde.



En moderne kommunikationsmodel.

Det er ikke sikkert, at modtageren/målgruppen er modtagelig for kommunikation. Og det er ikke sikkert, at ens signaler bliver forstået på helt samme måde, som man ønsker det. Hvis der er tale om massekommunikation, er det ekstra svært. Det gælder også ved film og litteratur. Det er langt fra givet, at læserne forstår en roman på samme måde som forfatteren. De enkelte læsere forstår den desuden ofte forskelligt – eller i hvert fald ikke på helt samme måde.

I dag er man klar over, at en kommunikationsproces ofte er ret kompliceret – og man ved, at det kan være svært at forudsige, hvordan et budskab bliver modtaget og forstået.

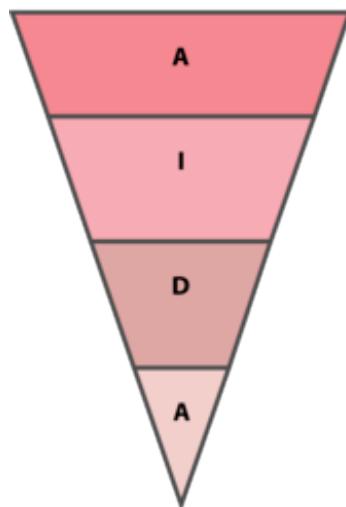


Kommunikationsmodel suppleret med støj og kontroksignaler.

I ovenstående model er indsattet støj og kontroksignaler. Det sidste er både signaler, som afsenderen udsender for at sikre, at modtageren forstår budskabet, og signaler som modtageren afsender for at sikre, at han/hun forstår budskabet. Det er fx 'Kan du høre, hvad jeg siger?' og 'Mener du...'. Også endelsen 'Ikke' fungerer egentlig som et sådant kontroksignal.

AIDA og HYSO

AIDA-modellen



A = Attention / "Opmærksomhed"

Modtagerens opmærksomhed fanges gennem et iøjnefaldende blikfang eller en fængslende tekst.

I = Interest / "Interesse"

Fastholder vi interessen, skyldes det ofte, at budskabet har relevans for os her og nu.

D = Desire / "Lyst og ønske"

Interessen skærpes, og nysgerrigheden får os til at læse videre i annoncen for at få mere at vide om produktet.

A = Action / "Handling"

Nu opfordres der til handling. Modtageren er nået til det punkt, at han vil vide, hvor produktet kan købes eller hvor han kan få yderligere informationer om det.

AIDA-modellen.

Hvis man arbejder med reklamer, ønsker man at en annonce både skal indeholde noget, der fanger opmærksomheden og noget, der kan skabe handling – samt noget der kan sikre overgangen til den ønskede handling. Dette kan beskrives med AIDA-modellen.

AIDA er en forkortelse for Attention – Interest – Desire – Action:

- Attention = opmærksomhed. Det som fanger øjet. Det kan være både tekst eller billede. Det er som regel placeret i reklamens øverste halvdel, ofte i venstre side, hvor det vandrette og det lodrette gyldne snit skærer hinanden.

Nogle eksempler kan være teksterne 'Alle taler om det' eller 'Op til 30 % rabat!' eller et portræt med øjenkontakt til læserne. Det kan også være en særlig farve, fx bruges gul ofte som blikfang.

- Interest = interesse. Det som får læseren til at fastholde interessen. Det kan også være både tekst eller billede.

Nogle eksempler kan være tekster som 'Nu kan du gøre noget ved det', 'Vind en luksusrejse med alt betalt' eller måden øjenkontakten etableres på – sendes der fx særlige sexuelle signaler?

- Desire = ønske/begær. Det som får læseren til at ønske – begære – produktet. Det er oftest tekst, der med masser af plus-ord beskriver varen på en nærmest uimodståelig måde.

Nogle eksempler kan være en tekst, der fortæller om, hvor fedt det er at dyrke en bestemt form for motion eller dans, eller en tekst, der fortæller om alt, hvad luksusrejsen omfatter. Det kan også være baggrundsbilleder, der viser en stemning, fx af nostalgi, fællesskab, fest og farver eller af en spændende fremtidsvision.

- Action = handling. Det som får læseren til at handle – at købe produktet. Det er oftest tekst, fx oplysninger om hvor varen kan købes. Meget brugt er også et billede af varen – ofte placeret i nederst højre hjørne, 'farvel'-eller 'bladre-hjørnet'.



Opgave: Reklame med AIDA 1

The advertisement features a bride and groom laughing at a wedding reception. The bride is wearing a white dress and a necklace, and the groom is in a tuxedo. They are seated at a table with a white tablecloth, surrounded by various items and professionals from different trades. Labels with arrows point to each professional:

- Bygningsmaler (Building painter)
- Tømrer (Carpenter)
- VVS-monter (VVS installer)
- Frisør (Barber)
- Optometrist (Optometrist)
- Ædelsmed (Jeweler)
- Tjener (Waiter)
- Skrædder (Tailor)
- Gastronom (Gastronom)
- Mediegrafiker (Media designer)
- Percelænsmaler (Porcelain painter)
- Ratorietekniker (Plumber)
- Vækslhuspartner (Business partner)
- Konditor (Confectioner)
- smedker (Smith)
- Tømrer

**DER STÅR EN
HÅNDVÆRKER
BAG..!**

Danmarks Tekniske Skoler

KLIK IND PÅ WWW.BLIVHÅNDVÆRKER.NU – HVIS DU ER TIL SYNLIGE RESULTATER!

Vil du være en vigtig del af særlige begivenheder..?
Drommer du om at skabe synlige resultater..? Vil du være med, hvor det sker, når hverdagen skal fungere..? Og vil du ha' en uddannelse, der er fremtid i..? – så har vi 105 uddannelser, der gør dig til en afgørende brikk i de begivenheder, vi alle husker..!
Klik ind på [www.blivhåndværker.nu](http://WWW.BLIVHÅNDVÆRKER.NU) og se, hvordan du kommer videre. Med også andre, der allerede nu skaber synlige resultater!

**DER STÅR EN
HÅNDVÆRKER
BAG..!**

Ejh. Danmarks Tekniske Skoler

Chili, november 2003.

Finde de fire AIDA-elementer i reklamen ovenfor.



Opgave: Reklame med AIDA 2

(kundekontakt)

Ugen nu, måske har jeg mistet unga.
Jeg taler jo denne sprug og kender
måske behovene. Det hældes en mån-
drift, jeg virkelig kan hjælpe
noget med min rådighed.

Lise Jensen
1. afd. kundekontakt
Mærkeværdi Afdeling

Hjem gør dig udfordring i hverdagen?

Har du lyst til et arbejde med økonomi og markedsføring, så er den 21-årige finans-
værtinnaelse måske noget for dig. Mærkeværdien ydmyger både team på en skole, og
gruppen i en af vores afdelinger. Du vil få lov fra starten, og du vil hurtigt få kundekon-
takt og salgsmæssige opgaver. Flotte venner og almindelig god som-rådgiver i banen,
fantastisk viden om finansværtinnaelse og gode kommunikationshåndhævelser. Lad os have fra dig, hvis du
har en idé, til et efter spørgsmål om værktøj.

Læs videre på www.danskebank.dk/finansværtinnaelse_og_mediens_ansigter

Danske Bank

Venligst stillet til rådighed af Danske Bank
Finde de fire AIDA-elementer i reklamen ovenfor.

Placér det gyldne snit og forklar hvordan det er brugt.

HYSO

Ved siden af AIDA-modellen kan man også bruge *HYSO* – Hey, You, See, So. Den siger delvis det samme som AIDA, men bruges mere bredt, bl.a. når man skriver artikler, sange og meget andet.

- *Hey* står for for det samme som attention, altså noget med at fange målgruppens opmærksomhed
- *You* minder om interest; her skal man vise målgruppen, at den pågældende artikel eller reklame mv. er relevant for netop de pågældende. Man kan oversætte You til identifikation
- *See* - hvor man forklarer, hvad emnet går ud på. Her ender vi ikke med desire, men med forståelse
- *So* - hvor sagens konsekvenser og perspektiver skitseres.

Som man kan se, ligner AIDA og HYSO hinanden i, at de begge skal fange målgruppen og gøre den interesseret. Men hvor AIDA lægger op til forførelse og overtalelse, lægger HYSO mere op til forståelse og erkendelse.

Mens AIDA således er mest oplagt at bruge i almindelige kommercielle reklamer og kampagner, er HYSO mere relevant i adfærdsregulerende kampagner – altså kampagner, der handler om at få målgruppen til fx at holde op med at ryge, ned sætte alkoholforbruget, dyrke motion eller spise sundere.



Opgave: HYSO

1. Find en reklame på nettet hvor HYSO er en bedre analysemodel end AIDA.
2. Foretag en analyse ned HYSO-modellen.

Model-View-Controller-arkitektur

Når man udvikler et større IT-system, er det en god ide at opdele det i flere dele. Dels er det lettere at være flere om at programmere de enkelte dele, og dels er det lettere senere at ændre i én del, uden at det påvirker de andre dele.

En måde at lave en opdeling på er Model-View-Controller-arkitekturen; kaldet MVC. Denne model bruges især i forbindelse med design af brugerinterface.

Modellen består af de tre dele:

- Model
- View
- Controller.

Hver del har et ansvarsområde. Der er forskellige måder at fordele opgaverne mellem de tre dele, men som regel er fordelingen af arbejdsopgaver som følger:

Model – View – Controller

Model

Model-delen håndterer data og de krav, der er til den enkelte type data. Det er også model-delen, der håndterer forretningslogik. Det er her man kan se, om der er tale om data til en bank, en forretning eller lignende.

View

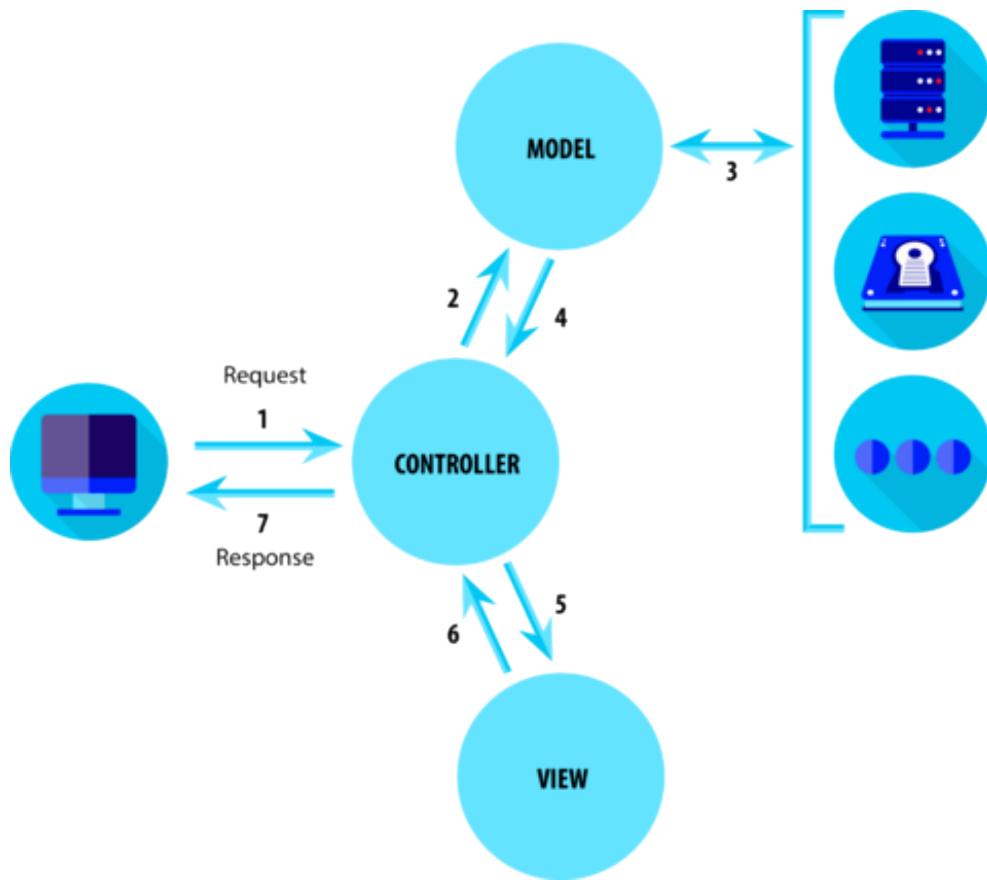
View-delen bestemmer det format, brugeren skal se data i. Formatet kan for eksempel afhænge af, hvilket skin en bruger har valgt, eller hvilken platform brugeren benytter.

Controller

Controller-delen modtager forespørgsler (requests) fra brugeren og udfører dem vha. passende ressourcer. Controller-delen tilbagesender passende svar (response) på forespørgslen.

Lad os se på et eksempel. Vi vil gerne udvikle en onlinebutik, der sælger tøj. Brugeren kan foretage forskellige handlinger. Hun kan se på varerne i onlinebutikken, og hun kan købe tøj. Butiksejeren er en anden type bruger, og han kan lægge varer på lager og fjerne dem igen.

Der sker følgende, hvis en bruger klikker på damesko for at se, hvad butikken har. Hun sender dermed en forespørgsel.



Model-View-Controller-modellen.

iStockphoto.com/KUO CHUN HUNG / Systime.

Controller-delen modtager forespørgslen (1).

Controller-delen undersøger forespørgslen og kalder model-delen (2) for at bede om de relevante data – nemlig alle damesko.

Model-delen henvender sig til databasen (3) for at få del i relevante data og sender disse til controller-delen (4).

Controller-delen henvender sig til View-delen (5) og modtager det format, data skal vises i (6).

Herefter sender controller-delen svar til bruger (7) i det relevante format. Hvis bruger for eksempel sidder med en smartphone, er der tale om et andet format, end hvis bruger sidder med en pc.

Bemærk at Response/Request er måden, man kommunikerer på i Klient og server-arkitekturen. I nogle versioner af MVC sendes Request direkte fra View-delen til brugeren.



Øvelse: Online skak

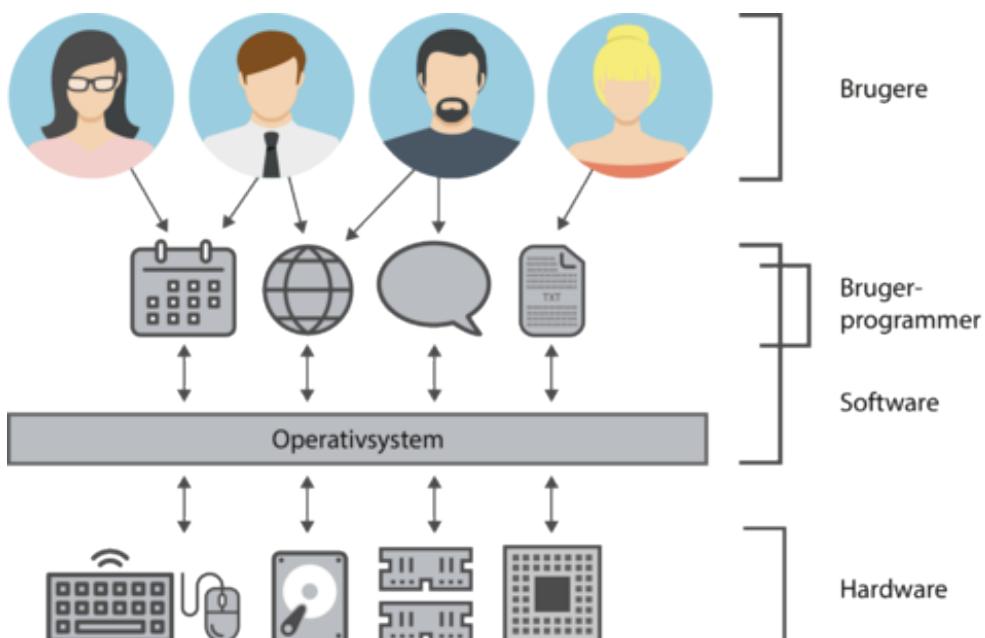
Man ønsker at lave spillet skak som et online-spil.

1. Hvordan kan man opdele et IT-system efter MVC-arkitekturen?
2. Overvej bl.a:
 - Hvor skal reglerne være?
 - Hvor skal en eventuel highscore-liste placeres?
3. Giv eksempler på handlinger, de tre dele kan udføre.

Operativsystem og processer

Når vi i dette afsnit taler om computere, er det i bredeste forstand. Det dækker pc'ere, mobiltelefoner, smarture, og andre enheder.

Et *operativsystem* (også kaldet et *stresystem* – forkortes til OS) er en slags mellemmand mellem brugere, brugerprogrammer og computerens hardware-dele. Figuren herunder illustrerer dette forhold.



Operativsystem abstrakt set.

iStockphoto.com/fonikum/IconicBestiary/Alex Belomlinsky / Systime

Figuren viser også, hvorfor man undertiden kalder kombinationen af operativsystem og hardware for en platform: Brugerprogrammer "kører ovenpå" platformen.

Brugerprogrammer er fx internetbrowsere, tekstbehandlingsprogrammer og spil.

Operativsystemet sikrer, at

- computeren kan køre flere programmer samtidigt,
- computeren er stabil og sikker,
- computeren kan deles mellem flere forskellige brugere med forskellige rettigheder og indstillinger.

Et operativsystem hjælper også programmører ved at gøre computeren let at skrive programmer til. Operativsystemet fungerer nemlig som grænseflade mellem programmer og hardware. Programmer, der kun kommunikerer med operativsystemet, er uafhængige af den underliggende hardware. Det betyder, at et program ikke skal skrives om, selvom det flyttes til en maskine med en anden hardware-sammensætning (og samme operativsystem).

Eksempel: Operativsystemer

Der er mange operativsystemer, fx: serien af Windows-systemer, Solaris, Android, iOS, Bada, Linux, Unix, MacOS, DOS, FreeBSD.

Operativsystemet er et kontrolprogram, der kontrollerer al kommunikation mellem programmer. Operativsystemet er ideelt set også det eneste program, der "snakker med hardwaren". Et program, der ønsker at snakke med et andet program eller med et stykke hardware, spørger operativsystemet om lov, og det er operativsystemet, der bestemmer, hvornår og hvordan kommunikationen kan foregå. Samlet kan man sige, at operativsystemet står for at administrere computersystemets mange systemressourcer:

| Systemressource | | Bruges af et program til fx |
|-----------------------------|--|--|
| Hardware-res-sourcer | CPU-tid (clock-cykler) | At blive udført |
| | Lager (RAM) | Gemme mellemregninger midlertidigt |
| | Lager (Hddisk) | Gemme data over længere tid, fx som filer |
| | Anden hardware (I/O-enheder) | Vise ting på skærmen Modtage bruger-indtastninger Sende ting afsted over internettet (via netkortet) |
| Software-res-sourcer | Mulighed for at kommunikere med et andet program | Samarbejde med andre programmer |

Systemressourcer.

Administrationen består i at alloker og deallokere systemressourcerne.

Operativsystem

Et operativsystem er et program installeret på en computer. Operativsystemet administrerer systemressourcer – blandt andet adgang til og brug af hardware. Programmer kommunikerer med hinanden og med hardwaren via operativsystemet.

Man skelner mellem programmer og processer. Ved et program forstår simpelt hen den mængde kode, der udgør programmet. Ved en proces forstår man et program, der er under afvikling.

Programmer er passive, mens processer er aktive. En proces rummer, udover programkode, information om sin tilstand, bl.a. om hvor den er nået til i udførelsen af sin kode.



Program: Data
(i RAM eller på disk)



Proces: Programkode + tilstand
(i RAM)

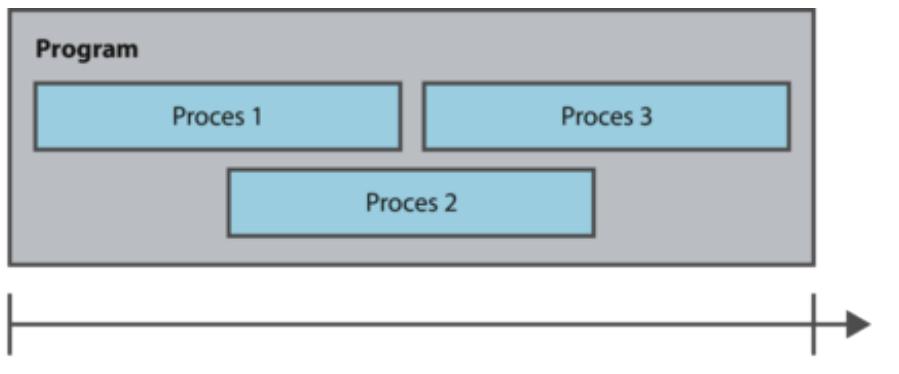
"Jeg har udført disse linjer kode..."

"Mine variable er lig med..."

"jeg har lige fået besked om, at
der er blevet trykket 'a'..."

Program og proces.

En proces kan sagtens indlæse andre processer i hukommelsen. Et kørende program kan derfor undertiden bestå af flere processer.



.Program med flere processer

Et program kan opfattes som en samling af processer, der skal udføres, eventuelt i en bestemt rækkefølge.

To processer kan også repræsentere to kopier (*instanser*) af samme program, fx hvis en bruger starter to instanser af fx et regneark.

For at koordinere deres fælles arbejde skal processer kunne "snakke sammen".

Formål med proces-kommunikation

Korrekt rækkefølge

Opgaver skal udføres i den rigtige rækkefølge i forhold til hinanden.

Formål med proces-kommunikation

| | |
|--------------------|--|
| Informationsdeling | Arbejde udført af en proces skal kunne overdrages til en anden (fx resultatet af en beregning). |
| Effektivitet | Opgaver, der fordeles som (evt. uafhængige) delopgaver på flere kommunikende processer, kan udføres hurtigere. |
| Overskuelighed | Program kan opdeles på en overskuelig måde i logisk sammenhængende processer med hver sin delopgave. |



OPGAVE: Skeduleringsalgoritmer

Man opstiller sædvanligvis følgende mål for skeduleringsalgoritmer:

- *CPU-udnyttelse*: effektiv udnyttelse af CPU'en
 - *Prioritering*: at vigtige processer får forrang fremfor mindre vigtige processer
 - *Minimal udførelsestid*: fra en proces indlæses, til den afsluttes, skal der gå kortest mulig tid
 - *Minimal klartid*: en proces skal være i tilstanden ready så kort tid som muligt
 - *Minimal ventetid*: en proces skal være i tilstanden waiting så kort tid som muligt
 - *Minimal svartid*: fra en proces indlæses, til den leverer de første resultater, skal der gå kortest muligt tid
1. Forklar, hvorfor målene sjældent kan opfyldes samtidig.
 2. Diskuter forskellen på de 4 måder at måle hastighed for procesafvikling [3, 4, 5 og 6].
 3. Giv et eksempel på en proces, hvor målet 6 er vigtigere end målet 3.
 4. Hvorfor kan man ikke sikre målet 5 alene ved hjælp af skeduleringsalgoritmer?

Personlige data

Elektroniske registre

Indtil 1960'erne var den enkelte dansker kun registreret ret få steder; foruden folkeregistret primært registre med de centrale begivenheder i vores liv – fødsel, dåb, bryllup osv. De enkelte virksomheder havde naturligvis også registre over deres kunder, leverandører og medarbejdere, og noget tilsvarende gjaldt for offentlige virksomheder og institutioner. Fx havde skoler et register over deres elever og et andet over deres lærere.

Det offentlige havde også registre over fx skattebetalere, husejere, billejere, modtagere af folkepension og meget andet. Man havde desuden også registre over personer, man mistænkte for at være farlige for samfundet, fx kommunister og socialister. Det sidste var ganske vist ikke helt lovligt.

Alle disse registre var imidlertid isolerede – man kunne ikke uden videre flytte data fra et register til et andet. Det krævede, at man manuelt aflæste en oplysning i ét register og tilføjede den i et andet.

I løbet af 1960'erne fik man elektroniske registre, og hermed blev brugen af registre lettere og hurtigere. Det blev også lettere at sortere sine poster på forskellige måder (efter forskellige felter), men der var fortsat tale om isolerede registre.

CPR-registret

Men med Det centrale personregister (CPR-registret) fra 1968 trådte brugen af registre ind i en ny tid. Alle borgere i Danmark fik et unikt CPR-nummer sammensat af vores fødselsdag og et løbenummer. Med dette nummer blev det i principippet muligt at kombinere oplysninger fra forskellige registre.

Beskyttelse af personlige data

I 1978 vedtog Folketinget love for udnyttelsen af både de private og de offentlige registre. Her blev der fastsat grænser for, hvad der måtte registreres, og hvordan oplysningerne måtte bruges.

Nogle af de vigtigste elementer var, at man ikke måtte registrere personfølsomme oplysninger som for eksempel seksuel orientering og etnisk oprindelse, at man kun måtte registrere relevante oplysninger, og at man ikke måtte samkøre registre uden tilladelse. Og denne tilladelse var svær at få.

I 2000 blev lovgivningen strammet op og samlet i den såkaldte Persondatalov – det fulde navn er Lov om behandling af personoplysninger. En af de vigtigste ændringer var, at private virksomheder nu skal indhente tilladelse hos de registrerede, hvis de vil bruge oplysningerne til markedsføring. Samtidig fik det offentlige nemmere ved at få tilladelse til at samkøre de offentlige registre, bl.a. for at bekæmpe skattesnyd samt snyd med dagpenge og sociale ydelser.



Opgave: Registrering af personlige oplysninger

1. Hvor meget har det offentlige registreret om dig via dit CPR-nummer? Find oplysningerne via borger.dk.
2. Hvor mange private virksomheder har registreret oplysninger om dig? Du kan ikke finde et samlet oversigt, så du må prøve at tælle dig frem. Du kan gå ud fra, at du er registreret alle steder, hvor du er fast kunde – forudsat du har opgivet navn og adresse. Mange steder bliver du også registreret, første gang du opgiver navn, adresse eller e-mail. Prøv at lave en optælling.

GDPR

I 2018 blev persondataloven erstattet af persondataforordningen (GDPR). Persondataforordningen er ikke en dansk lov, men en forordning der blev vedtaget i EU. Det er Datatilsynet, der fører tilsyn med at loven overholdes i Danmark.

Hensigten med forordningen er at sikre retten til et privatliv og retten over egne data.

Alle persondata er ikke lige følsomme, og derfor er reglerne ikke så stramme for almindelige oplysninger som for personfølsomme oplysninger.



Opgave: Persondataoplysninger

Eksempler på persondata kan være race, økonomi, navn, etnisk oprindelse, religion, adresse, fødselsdato, biometriske data, gæld, seksuelle forhold og arbejdsmarked.

Fordel ovenstående oplysninger i kategorierne almindelige personoplysninger og følsomme personoplysninger.

Tjek svarene på [Datatilsynet](#).



Opgave: Grundlæggende principper for GDPR

De grundlæggende principper bag GDPR er:

- Lovlighed, rimelighed og gennemsigtighed
- Formålsbegrensning
- Dataminimering
- Rigtighed
- Opbevaringsbegrensning
- Integritet og fortrolighed

Hvad ligger der bag disse principper? Tjek svarene på [Datatilsynet](#).

GDPR giver den registrerede en række rettigheder i forhold til egne data. De vigtigste rettigheder er:

- Retten til at modtage oplysning om en behandling af sine personoplysninger (oplysningspligt)
- Retten til at få indsigt i sine personoplysninger (indsigtsret)
- Retten til at få urigtige personoplysninger berigtiget (retten til berigtigelse)
- Retten til at få sine personoplysninger slettet (retten til at blive glemt)
- Retten til at gøre indsigelse mod at personoplysninger anvendes til direkte markedsføring

- Retten til at gøre indsigt i mod automatisk individuelle afgørelser, herunder profilering
- Retten til at flytte sine personoplysninger (dataportabilitet)



Opgave: Myndigheder og virksomheder

GDPR skal overholdes af mynigheder og virksomheder. Derfor er der nogen i myndigheden eller virksomheden, der har ansvaret for at dette sker. Disse er dataansvarlige og databehandler.

Hvilke opgaver har den dataansvarlige, og hvilke opgaver har databehandleren?

Sociale medier

Sociale medier



iStockphoto.com/alexsl

Vi mødes på de sociale medier som aldrig før. Et socialt medie er en internetapplikation, der skaber fælleskaber mellem brugerne. Her kan man kommunikere og dele indhold, mens andre kan se med. Som regel har det enkelte sociale medie et bestemt formål og henvender sig til en bestemt gruppe. Brugerne lægger indhold ind og deler tekst, billeder, film o.l.

Eksempler er: Instagram, Facebook, Snapchat, Youtube, Twitter, Flickr, StumbleUpon, LinkedIn, blogs og Google+.



Opgave: Sociale medier

Med udgangspunkt i enten et af ovenstående sociale medier eller et du selv vælger, skal du besvare følgende spørgsmål:

- Hvad er hovedformålet med det sociale medie? Er det det samme formål som mediet oprindeligt var beregnet til?
- Hvilke typer af brugere er på det sociale medie? Hvad bruger de mediet til?
- Hvor mange er på det sociale medie?
- Find eksempler på adfærd på det sociale medie, som almindelig sund fornuft burde have forhindret.
- Find eksempler på problematisk adfærd på det sociale medie og diskutér hvilke etiske regler, der er blevet brutt.



Opgave: Virtuel spilverden

Lav ovenstående opgave med udgangspunkt i en virtuel spilverden som for eksempel spillet Counter Strike.

Er spillet et socialt medie?

Netikette

Når man kommunikerer og deler indhold på de sociale medier, er der nogle regler for hvordan man kommunikerer. Disse sikrer at alle føler sig godt tilpas og ingen føler sig generet. Disse regler kaldes netikette (en sammenbrækning af net og etikette). Når man kommunikerer og deler indhold på de sociale medier, er der nogle regler for hvordan man kommunikerer. Disse sikrer at alle føler sig godt tilpas og ingen føler sig generet. Disse regler kaldes netikette (en sammenbrækning af net og etikette).

Netiketteregler:

1. Undgå at vække vrede ved at sende provokerende eller fornærmende kommentarer.
2. Respekter andres privatliv. Undlad at lægge oplysninger, billeder eller videoer ud af andre som de ikke selv ville lægge ud.
3. Spam ikke andre ved at sende store mængder uopfordret mail.
4. Udvil sportslighed når du spiller på nettet uanset om du vinder eller taber.
5. Lad være med at genere andre ved bevist at irritere dem for at få dem til at reagere (troll).
6. Hold dig til det emne du kommenterer.
7. Undlad at bruge stødende sprog.
8. Undlad at besvare negative opslag med negative opslag. Svar positivt.
9. hvis nogen stiller et spørgsmål og du kender svaret, så yd hjælp.
10. Sig tak hvis du modtager hjælp.



Opgave: Netikette

Har du altid overholdt ovenstående 10 regler for netikette?

Beskriv et eksempel på en overtrædelse af hver af de 10 regler. Det skal enten være dig selv eller andre der har overtrådt dem.

Trelags-arkitektur

OBS

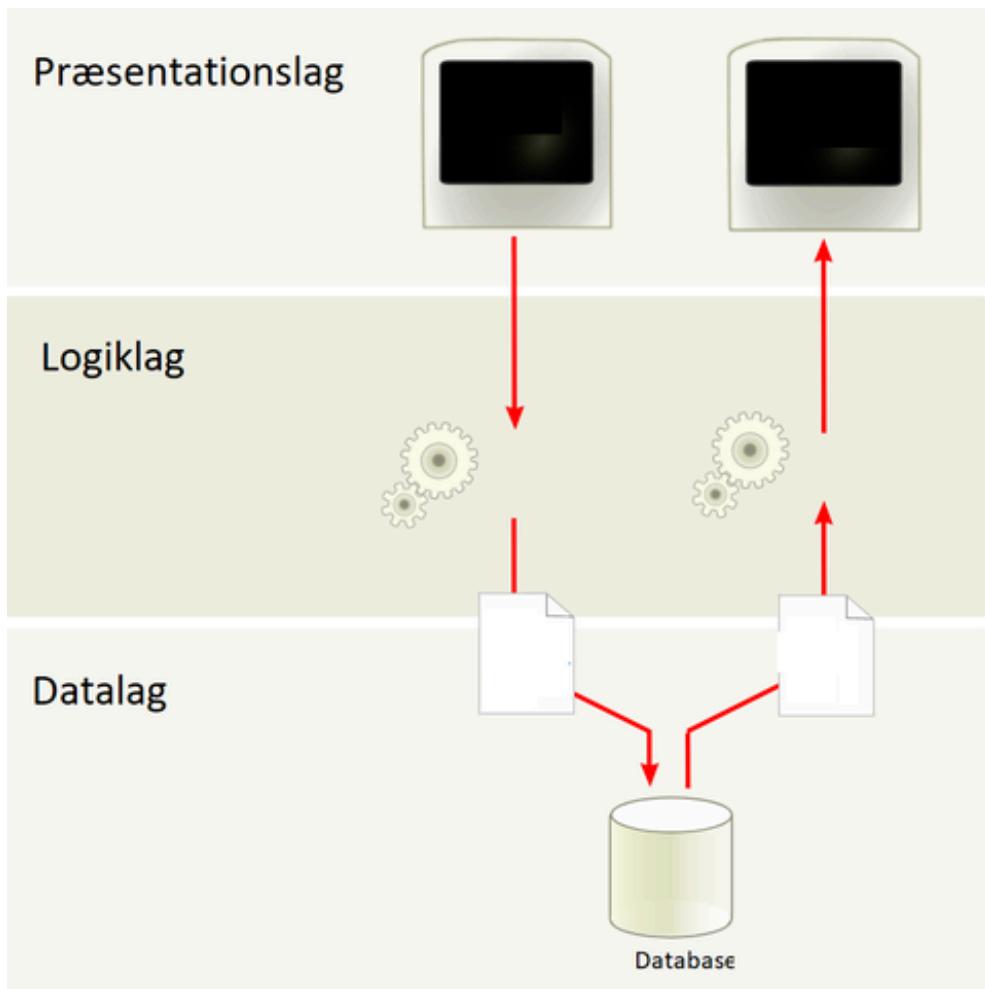
Dette er kernestof på B-niveau.

Trelags-arkitekturen er en videreudvikling af klient-server-arkitekturen og består af tre lag:

- Præsentationslaget
- Datalaget
- Logiklaget

Præsentationslaget svarer til klient-delen i klient-server-arkitekturen.

Datalaget og logiklaget er en videreopdeling af server-delen i klient-server-arkitekturen.

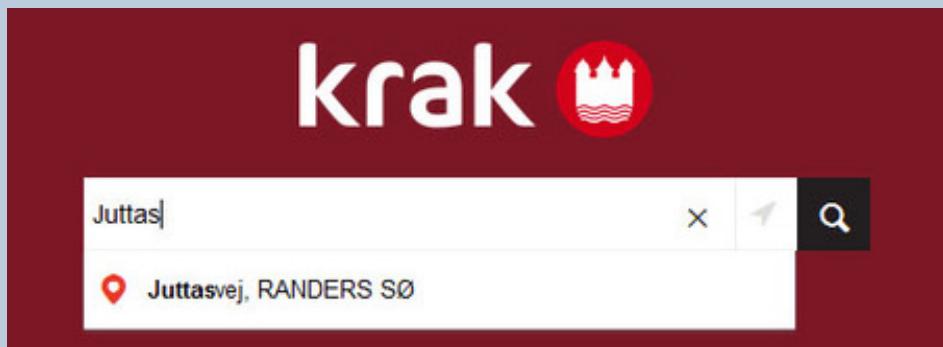


Typisk vil man lave denne videreinddeling af serveren, hvis man har mange forespørgsler på sine data.

Eksempel: Krak

På krak.dk er der mange data i form af for eksempel personer, adresser og kort. Da der er så meget aktivitet på Krak, er Krak placeret i en trelags-arkitektur.

Klienten beder om at se, hvor mange adresser der er på Juttasvej. Derfor indtastes "Juttasvej" i søgerfeltet:

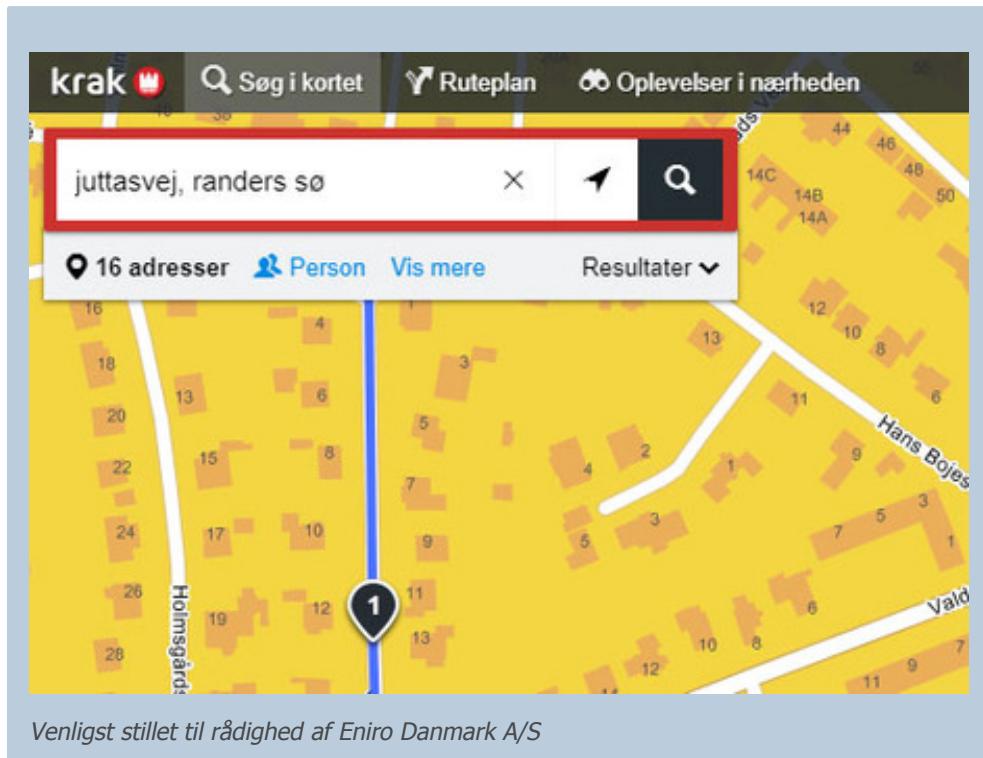


Venligst stillet til rådighed af Eniro Danmark A/S

Logiklaget modtager det indtastede og sender en forespørgsel til datalaget.

Derefter modtager logiklaget alle adresserne på Juttasvej og tæller dem.

Logiklaget sender bl.a. denne information tilbage til præsentationslaget:



Overordnet set har de tre lag følgende opgaver:

Præsentationslaget

- Input fra brugeren (tastatur, museklik, berøring o.l.)
- Præsentation af grafik, tekst, billede o.l. til brugeren
- Sende information mellem præsentationslaget og logiklaget

Logiklaget

- Foretage logiske beregninger
- Hente informationer fra præsentationslaget og på baggrund af disse hente data fra datalaget
- Lave beregninger og andre manipulationer på data fra datalaget

Datalaget

- Opbevare data
- Søge i data
- Sende information til logiklaget



Opgave: Facebook

Placér Facebook i en trelags-arkitektur.



Opgave: Store online-spil

Vælg et stort online-spil som fx Counter-Strike, og placér det i en trelags-arkitektur.



Opgave: Netbank

Placér en netbank i en trelags-arkitektur.

Elevoplæg

Nedenfor ses en række emner. De skal være overskriften på et elevoplæg. Når du har valgt et emne og kender den mængde tid, som du må bruge på dit oplæg, skal du

- opsøge så meget viden om emnet som muligt (husk at være kildekritisk)
- udvælge det væsentligste materiale og lave en struktur
- lave en præsentation med
 - en pointe pr. side
 - få farver
 - få fonte
 - gerne et billede/en illustration pr. side
- træne dit oplæg

Emner til oplæg

- Alan Turing
- Ansigtsgenkendelse
- API (Application Programming Interface)
- Autonome biler
- Big data
- Bitcoin
- Bluetooth
- Brugertracking
- Compilere
- Cookies
- Deep learning
- eBooks
- Esport
- Hvad er et godt password?
- Hvem indsamler oplysninger om dig? Disconnect og Ghostery.
- Intelligens og Eliza-test
- Internettets fysiske opbygning
- IoT - Internet of Things
- Komprimering
- Kryptering
- Kunstig intelligens
- Kvantecomputere
- Nanoroboter
- NemID - MitID
- Neurale netværk
- NFC - Near Field Communication
- Robotter der dræber
- Selvkørende biler
- Sorteringsalgoritmer
- SSH - Secure Shell
- Stregkoder
- Søgemaskiner - hvordan virker de?
- The dark Web
- UX - hvad laver en Usability experience designer?
- Virtual reality
- XP - Extreme Programming