Introduktion

Et forløb om modellering af et baskethold skrevet i JavaScript og bearbejdet gennem stepwise improvement.
Præsentationsvideo: http://kortlink.dk/ts6p

Målgruppe:

Informatik-elever på C-niveau. Enten valgfag eller som naturvidenskabeligt fag.
Forløbet kan udvides med databaser (fx SQLite) i Informatik B.

Elevernes forudsætninger:

Vi antager, at der forud for dette forløb er arbejdet med simplere programmeringsformer, hvor grundlæggende begreber er introduceret. Det kunne f.eks. være blok-programmering med skift mellem blokke og tekstkode. Herunder variable, løkker og forgreninger. Systemer der tilbyder træning heri er bl.a. Grasshopper (som er gamificerede JavaScript-øvelser på mobilen) og OzoBlockly (et udviklingsmiljø til Ozobot-robotterne, som snart tilbyder JavaScript blok- og tekstprogrammering).

Faglige mål: Hvilke faglige mål? Hvordan opfylder eleverne de faglige mål via forløbet?

Konstruktion af it-system som løsning til en problemstilling
Eleverne skal kunne

- *løse et mindre problem ved at beskrive problemet, samt designe, realisere og afprøve et it-system gennem brugerorienterede teknikker*
  - Eleverne løser mindre problemer undervejs i forløbet, der er bygget op over fire iterationer af et IT-system til at håndtere et baskethold (stepwise improvement). Der programmeres i JavaScript og koden afvikles i repl.it
- *demonstrere viden om fagets identitet og metoder*
  - Eleverne opnår viden om fagets identitet og metoder ved at de ser en 1:1 sammenhæng mellem kode og brugergrænseflade. Her ser se behandling af og interaktion med digitale data i form af spillerobjekter på et baskethold.

Repræsentation og manipulation af data
Eleverne skal kunne

- *modellere data samt redegøre for udvalgte typer af data og anvende disse i simple it-systemer eller udvidelser af disse*
  - Eleverne arbejder med repræsentation af data i form af javascript objekter i den grundlæggende model. I den udvidede model skal der gemmes i JSON format.

    - Manipulation af data sker via de forskellige funktioner som eleverne arbejder med og selv udvikler.

Programmering
Eleverne skal kunne

- identificere basale strukturer i programmeringssprog, modellere programmer og anvende programmering til udvikling af simple it-systemer
    - Det sker konkret ved at arbejde med disse strukturer i javascript. Hvis man ikke ønsker at inddrage objekter (hvilket ikke er et krav), kan programmet også skrives med en simplere datastruktur.

Kernestof: Hvad skal eleverne lære, og hvilket materiale anvendes?

Repræsentation og manipulation af data

- *abstraktion og strukturering, begrebs- og datamodeller*
    - Eleverne overvejer, hvilke former for data og funktioner der vil være relevante at lægge til systemet
- *data og datatypers repræsentation og manipulation*
    - Der arbejdes med variable, javascript objekter og funktioner

Programmering

- *funktioner*
    - eleverne programmerer forskellige funktionstyper og forholder sig til, hvad de gør
- *variable, sekvenser, løkker og forgreninger*
    - eleverne arbejder med disse begreber i arbejdet med at udvikle funktioner og forstå de eksisterende

Relevant læsebogsstof:
Mathiasen, K. *"Informatik C"*, Praxis 2017: kapitlet "Programmering" side 177-192.
Særligt siderne:
177-181: Modeller.
186-191: Variable og funktioner.

Martin Damhus, Jesper Buch, Elisabeth Husum m.fl.: "Informatik", SYSTIME 2017 (ibog).
Kapitel 3, Hovedafsnit "Programmering"

Forløbets opbygning

Som beskrevet ovenfor, er temaet for forløbet at modellere et problemområde (håndtering af at basketball hold).
For at holde styr på basketballholdet er der udviklet et system, hvor de mest basale attributter omkring spillerne er kodet ind (navn, alder, værdi). Der er også lavet nogle grundlæggende funktioner (displayTeam, findAge, findNumberOfPlayers).

Eleverne skal arbejde med systemet i forskellige faser:

1. De skal afprøve programmet (*use*) og finde ud af, hvordan det virker. Dette sker i første omgang ved at klikke på de forskellige dele af programmet og prøve dem af uden at de ser på koden. Programmet afvikles og aftestes på repl.it. Eleverne har med repl.it adgang til debugging af deres kode.

2. De skal se på koden (*analyze*) og prøve at finde ud af, hvordan programmet er bygget op og hvordan de forskellige dele fungerer. Herunder f.eks.:
    a. Hvor er data om spillerne?
    b. Hvor er funktionerne defineret og hvordan virker de?
    c. Hvordan er sammenhængen mellem kode og interface i browseren?

3. De skal løse en række opgaver som vi stiller med differentierede mål (*modify*). Et eksempel med alder og skader på de tre niveauer kunne være:
    a. Grøn: beregn samlet værdi af holdet.
    b. Gul: beregn samlet værdi af ikke skadede.
    c. Rød: Indfør ny attribut og brug den i en beregning

4. De skal omstrukturere forskellige dele af programmet (*modify*) så det bliver bedre og mere anvendeligt. Det kan gøres på flere måder, f.eks. ved:
    a. Omskrivning til bedre kode (dele funktion op i beregningsdel og udskrift)
    b. Bedre dokumentation i form af kommentarer
    c. Optimering og tydeliggørelse af funktioner og programstrukturer så det fremstår mere velorganiseret

5. De skal udvide programmet med nye funktioner som de selv mener er relevante - naturligvis med argumentation (*extend*). På denne måde skaber vi en sammenhæng mellem problemområdet og den konkrete implementering og får eleverne til at overveje, hvordan systemet kunne forbedres og udvikles.

Udvidelsesmuligheder til B-niveau

På B-niveau vil det være oplagt at arbejde med lagring og håndtering af data i et separat format.

En mulighed kunne være at arbejde med JSON som datamodel. Herved opnår man at eleverne kan arbejde med repræsentation af data uden at skulle arbejde med databaser. JSON er det mest brugte format til udveksling af data mellem forskellige IT systemer og en rigtig database er en stor udvidelse af forløbet. En mulighed for at bruge JSON i forløbet er skitseret herunder:

De skal konstruere en udvidelse til programmet som kan gemme spillerdata i JSON format (create). Oplægget er, at klubben skal sælges, og at den den nye klub har sit eget system til håndtering af spillere. Den nye klub er altså kun interesseret i data om spillerne, og de skal

leveres i JSON format, så de kan indlæses i det nye system. Eventuelt skal data indlæses et andet sted for at sikre at de kan anvendes.

Eftersom JSON primært anvendes som et udvekslingsformat er det dog mere oplagt at koble en underliggende database til programmet hvor spillerdata hentes og gemmes.

På B-niveau er der krav, om at eleverne skal implementere en database i et it-system og på denne måde kan eleverne arbejde med E/R diagrammer og normalformer. Også denne del vil kunne overføres til andre lignende problemområder.

Centrale begreber

Funktioner

Vi vil i forløbet arbejde med fire forskellige former for funktioner. Vi henter inspiration fra IT systemer, men skærer ned på kompleksiteten ved ikke at formalisere hvordan model, problemområde og anvendelsesområde spiller sammen.
Ved at opdele funktionerne i funktionstyper skærper vi elevernes bevidsthed om, hvad det er de foretager sig, og hvad de ønsker at opnå med udvidelsen af programmet.

De 4 funktionstyper er følgende:
Mathiassen Lars, m.fl: *Objekt orienteret analyse & design*, Forlaget Marko, 2001 siderne 135-139

Aflæsning	En aflæsningsfunktion kan f.eks. være at se antallet af spillere. Denne funktion er implementeret i version 0.
Opdatering	En opdateringsfunktion kan f.eks. være at tilføje en ny spiller. Denne funktion er implementeret i version 1.
Beregning	En beregningsfunktion kan f.eks. være at finde gennemsnitsalderen for alle spillere, mere avanceret for alle skadede spillere. Denne funktion er implementeret i version 2.
Signalering	En signaleringsfunktion kan f.eks. være at markere med en advarsel hvis gennemsnitsalderen er for høj, mere avanceret hvis en vis procentdel af spillerne er skadede. Denne funktion er implementeret i version 3.

Model og problemområde

Vi arbejder med modellering som det centrale begreb. Vores model består af de data og funktioner, som tilsammen udgør programmet.
I læreplanen anvendes kun problemområde og ikke anvendelsesområde. Vi bruger derfor problemområde mere bredt som et udtryk for det, der skal beskrives og modelleres.

Arbejdsformer

Vi ønsker, at der så vidt muligt skal indgå **"anderledes" undervisning** i alle lektioner for eksempel:

- Kinæstetisk øvelse (funktionsbegrebet, med objekter som variable)
- Quizlet - Live - begrebsindlæring
- Tarsia puslespil - begrebsindlæring
- Sortere i sandt eller falsk
  Aktiviteten er ikke nævnt et specifikt sted i lektionsplanen, men det er en god elevaktiverende aktivitet at bringe i spil, hvis der en dag bliver et kvarter til overs. Eleverne skal producere udsagn, som makkeren skal vurdere sandhedsværdien af
- Skattejagt Woop App kan bruges til en anderledes summativ evaluering.
  Skattejagten går ud på, at man sætter nogle poster på forskellige GPS-koordinater, hvor man kan stille et multiple choise spørgsmål. Når man svarer rigtigt, ser man på kortet på sin mobil, hvor den næste post er. Man kan sende eleverne ud i hold ét efter ét og tage tid, hvem der kommer hurtigst tilbage igen. Spørgsmålene kunne fx være begreberne fra Quizletten fra lektion 6.
  Det er også muligt, at lade eleverne selv lave skattejagter til hinanden. Så kan eleverne selv finde på spørgsmål og svar, som selvfølgelig skal godkendes af læreren. I denne video kan man se, hvordan man opretter skattejagter i app'en:
  kortlink.dk/trdf

Vi vil gerne inddrage **refleksive øvelser** - arbejdsark som et google-form med efterfølgende **JITT** (**J**ust **I**n **T**ime **T**eaching) herunder **formativ evaluering** og **refleksionsspørgsmål**

- hvilke oplysninger har man som ejer brug for når man har et baskethold (analyse af problemområdet) - brainstorm/induktiv øvelse)
- Hvad kendetegner en basketspiller? (hvilke attributter er vigtige? F.eks. alder, køn, højde, værdi, scoreprocent?)
- mønstergenkendelse - hvilke dele af af koden ligner hinanden (for-løkker, datastrukturer etc.)

Mere om JiTT:
https://science-gym.dk/pilot/jitt/jitt.pdf
http://webphysics.iupui.edu/JITT/ccjitt.html
Endelig har vi lavet en **skabelon** til **evaluering af forløbet** og vil anvende **Woop App** som summativ evaluering.

Didaktiske principper:

I forløbet har vi først og fremmest lagt vægt på princippet "*use-modify-create*" eller en variation heraf, nemlig "*use-analyze-modify-extend*". Arbejdet med JavaScript kræver noget erfaring, før det bliver naturligt for eleverne, at skabe selvstændigt. Meget tid vil gå med finde og debugge syntaks, for at skabe et script fra bunden, hvorfor vi har valgt, at eleverne i dette forløb skal være skabende ved at udvide et eksisterende script.

Vi bruger *stepwise improvement* til de større modifikationer eller udvidelser af koden. Generelt arbejder eleverne i par, og mødes med andre elever eller par for at få ideer og sparring på deres arbejde, hvorefter de kan forbedre deres kode. Man skal her være opmærksom på, at direkte afskrift af kode ikke er hensigtsmæssig.
Til at introducere konkret syntaks og teknikker bruger vi *Worked Examples*, hovedsageligt med video, som er særdeles velegnet til at vise og forklare en proces.
Efter et forløb med blok-programmering kan eleverne gennemskue tekstbaseret programmering. Da vi giver dem koden, skal eleverne ikke skrive fra bunden og skal i første omgang delvis kopiere kode fra den, de får.
Eleverne vil støde på JavaScript mange steder, da nettet er bygget op om det, og koden kan ses i alle browsere.
Til eksamen vil man kunne udlevere noget tilsvarende, hvor eleverne skal ændre casen til den, der er spurgt om.

Forløbets opbygning

Forløbet er bygget op over 5 iterationer over hver sin version af koden. Bud på specifikationer ses i filen: http://kortlink.dk/tue5

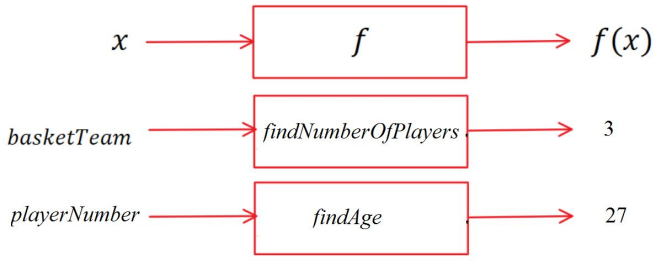
version af kode der udleveres	**udleveres (use/analyze)**	**modify/extend**	**Funktionstype / fokus**
Version 0	Aflæs alder. Aflæs antal spillere. Vises i Alert	Aflæs værdi af spiller. Vises i alert.	aflæs
Version 1	Tilføj spiller med .push og i koden	Alle: Tilføj spiller med i koden. Grøn: Analysere hvordan en ny attribut implementeres. Rød: Tilføj attribut og derefter ny spiller med .push og i koden.	opdater
Version 2	Find gennemsnitsalder	Grøn: beregn samlet værdi Gul: beregn samlet værdi af ikke skadede Rød: Indfør ny attribut og brug den i en beregning (fx find %del af ikke-skadede)	beregn
Version 3	Giv signal, hvis gennemsnitsalderen er for høj (når der lægges en ny spiller ind)	Grøn: Giv signal, hvis antallet af skadede er for højt (når der lægges en ny spiller ind). Gul: Er værdien af de ikke-skadede høj nok? Rød: Eleverne skal selv tilføje en	signalering(/refaktorering)

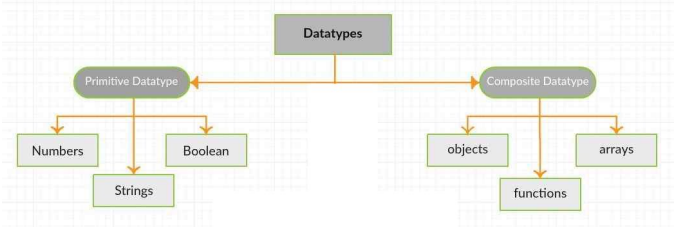
		funktion efter eget valg.	
Version 4	Tilfør style	Flyt om på elementer Lav ændringer af knapperne Juster overskriften Ændr skriftype og farve	CSS

Forløbsplan

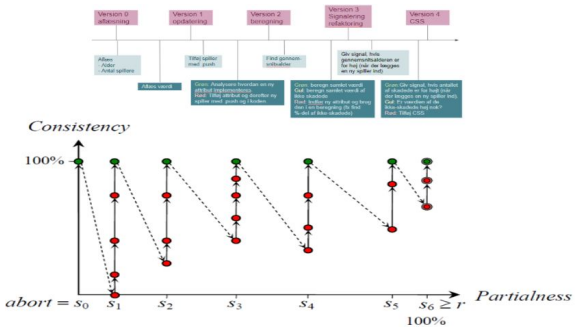
Skitse over forløbsplanens opbygning på lektionsniveau.
Hver lektion er normeret til 90 minutter.

Lektion nr.	**Fagligt indhold**	**Elevmateriale**
1	**Brainstorm** på "Baskethold". Eleverne arbejder med hvert punkt, først individuelt/i par og derefter opsamling i plenum. Forslag til generelle begreber og konkrete fænomener i modelleringen af et baskethold og sammenhænge mellem disse. **Introduktion** til JS. Hvor og hvorfor anvendes JS? Video: https://www.youtube.com/watch?v=gT0Lh1eYk78 **Opret bruger** i repl.it: https://repl.it/ **Use**: Se instruktions-videoen:kortlink.dk/tqg8 (Introduktion til repl). Åbn linket til JavaScript-koden DreamTeam version 0: Opret din egen egen kopi af programmet ved at klikke på "Fork". Afprøv koden ved at klikke på "run". Skriv en kort beskrivelse (500 anslag) af programmets formål og funktionalitet (hvad kan det?). Det kaldes en *specifikation*. Gem specifikationen i din logbog. Bud på specifikationer findes i linket: http://kortlink.dk/tue5 Vis 2. slide i denne ppt: kortlink.dk/tpxq (overblik over modellens data) **Afslutning**: (kan også foretages løbende i OneNote Class Notebook) ExitTicket fx i Google Analyse. Link til elever: kortlink.dk/tpxt Hvad har du fået med fra timen i dag?	Lektie til modulet: Mathiasen K. "Informatik C" side 177-181. Afsnit om modeller. Arbejdsark: kortlink.dk/tqtp

2	Modulet begynder med at afprøve programmet DreamTeam version 0 fra sidst igen. **Analyze**: I par: Skriv kommentarer og identificer mønstre (patterns) i programmet. Stilladseres af spørgsmål i arbejdsark. Arbejdsarket indeholder bl.a. et link til et **WE** kortlink.dk/tqg9 (hvordan kommenteres i JS kode?). Feedback på analyser i matrix. Bland grupperne, og lad eleverne forklare hinandens kommentarer. Fortsat arbejde på at forbedre analyse.(10 minutter) Fælles opsamling: Væsentlige pointer fremhæves. Fx sekvensering af program, ingen linjer kan undværes i koden, hvorfor er det nødvendigt at skrive forklarende kommentarer i kode? Grænseflade, funktionalitet og repræsentation kan inddrages (Nowack side 16). Centrale mønstre fremhæver (løkker, forgreninger, funktioner). Analyser gemmes i logbog. **Funktioner**. Kinæstetisk øvelse: Elever markeres som spillere med skilte med de valgte attributter (navn, alder, værdi, ….). En papkasse (funktion) skal tælle spillerne - eller skrive alderen op på alle spillere og lægge sammen. Evt. perspektivere til matematiske funktioner, der tager tal som uafhængig variabel. Overvej forskelle mellem funktioner i matematik og i programmering.  **Stepwise-improvement**: Vis 2-4 slide i denne ppt: kortlink.dk/tpxq (overblik over modellens data) **Afslutning**: ExitTicket fx i Google Analyse. kortlink.dk/tpxq Hvad har du fået med fra timen i dag?	Lektie til modulet: Mathiasen K. "Informatik C" side 186-191. Afsnit om variable og om funktioner. Link til arbejdsark til elever: kortlink.dk/tpyq
3	**Arbejdsark som google form:** Mulighed for formativ evaluering: Eleverne skal se et **WE** kortlink.dk/tqga (hvordan skrives *findAge()* funktionen? hvordan oprettes en knap?) **Analyze**: Sammenhængen mellem **html-koden** og **User Interface** i browseren. Link til koden DreamTeam version 0: **Modify**: ændre teksterne på knapperne til dansk/engelsk. Opsamling i plenum. Fokus på JavaScript-del vs. html-del. Hvilken del gør hvad? Hvordan identificeres de i koden? **Extend**: Eleverne skal udvide koden med en ny funktion med tilhørende knap, der **aflæser** en værdien af en spiller. **Peer feedback:** Eleverne mødes under sekvensen og sparrer med hinanden. **Opsummering** og refleksioner er rummet i formularen	arbejdsark med formular: kortlink.dk/tpyb

4	**Use**: Eleverne skal afprøve en DreamTeam version 1, uden at kigge på koden. Eleverne skal se et **WE** kortlink.dk/tqgc (hvordan tilføjes en ny spiller i koden?) **Analyze**: Eleverne skal analysere, hvordan nye spillere gemmes via knappen "add player". **Modify**: Nye spillere kan findes på: http://www.basketligaen.dk/da/top/statistikker/spillere/ Eleverne skal tilføje en eller flere spillere i koden. Præsentation om **datatyper**: tekst (string), tal, boolean. Desuden array og object. Navngivning af variable.  Konkrete eksempler på datatyper fra koden: kortlink.dk/tpyp **Quiz om datatyper** kortlink.dk/tpyh Arbejde med quizlet ud fra ark om **differentieret undervisning**: Afslut med **Quizlet live** **Analyze/Extend (differentieret): opdater** med ny boolsk attribut. Læreroplæg: Fortæl en historie om holdet. Hvilke attributter kan være relevante at tilføje? Grøn: Analysere hvordan en ny attribut implementeres. Skal også ses i et **WE**: kortlink.dk/tqgc (hvordan implementeres en ny attribut?). DreamTeam version 2 udleveres. Rød: Eleverne skal selv tilføje en ny boolsk attribut til koden: injured: true/false. Kan være lærerstyret. Test koden. DreamTeam version 2 udleveres. Vis 2-5 slide i denne ppt: kortlink.dk/tpxg (overblik over modellens data) **Afslutning**: ExitTicket fx i Google Analyse. kortlink.dk/tpxv Hvad har du fået med fra timen i dag?	link til arbejdsark. http://kortlink.dk/tqfp Arbejdsark Quizlet differentieret:kortlink.dk/tpyq
5	Eleverne skal se et **WE** kortlink.dk/tqgf (hvordan skrives en funktion, der beregner? Hvordan vises resultatet) **Use**: Eleverne skal afprøve DreamTeam version 2 og skrive en (ny) specifiikation til programmet. Link til eleverne: Version 2 **Analyze:** Eleverne skal analysere koden mht. den nye funktion, der **beregner** gennemsnitsalderenen, *findAverageAge()*. Kort opsamling i plenum. Fokus på løkke. **Extend**: Eleverne skal udvide koden med en beregningsfunktion: grøn: beregn samlet værdi af holdet. gul: beregn samlet værdi af ikke skadede. rød: Indfør ny attribut og brug den i en beregning (fx find %del af ikke-skadede). Eleverne tester deres funktion, opdaterer specifikationen og præsenterer den for en peer. Eleverne skal opsamle deres viden om JavaScript-kommandoer i et **Tarsia-puslespil**. Link til zip-fil med puslespillet: kortlink.dk/tuem **Afslutning**: ExitTicket fx i Google Analyse. kortlink.dk/tpxx Hvad har du fået med fra timen i dag?	link til arbejdsark: kortlink.dk/tpyk
6	Sekvensen indledes med et **WE** kortlink.dk/tqgg (Hvordan skrives en if-sætning?	link til arbejdsark:

	hvordan signalleres?) **Use:** Afprøv version 3 af koden, hvor der **signalleres**, hvis gennemsnitsalderen bliver for høj. Version 3 **Analyze**: Eleverne skal forklare hvordan koden virker. Hvad er der sket med funktionen *findAverageAge()* og hvorfor? Fælles opsamling: Pointer: Opdeling af *findAverageAge()* = refaktorering af koden. Link til **Quizlet** om **Kommandoer i JavaScript** kortlink.dk/tpym **Extend:** Eleverne skal udvide koden med en ny signallering: grøn: Giv signal, hvis antallet af skadede er for højt (når der lægges en ny spiller ind). gul: Giv signal, hvis værdien af de ikke-skadede for lav (når der lægges en ny spiller ind). rød: Undersøg selv hvordan CSS fungerer og lav eller modificer en CSS-fil til projektet. En færdig CSS-fil kan udleveres i version 4. Eleverne tester deres kode, og præsenterer den for en peer. **Afslutning**: ExitTicket fx i Google Analyse. kortlink.dk/tpxy Hvad har du fået med fra timen i dag?	http://kortlink.dk/tqfn
7	Eleverne skal tilføje en funktion efter eget valg. Teste den og præsentere den for en peer. For de hurtige er der mulighed for at arbejde med CSS: link til video: kortlink.dk/trqg link til kode: https://repl.it/@raskrawen/version4 **Afslutning**: ExitTicket fx i Google Analyse. kortlink.dk/tpxz Hvad har du fået med fra timen i dag? **Afslutning på forløbet:** Opsamling i plenum: Hvordan har koden udviklet sig. Fokus: Stepwise-improvement - se evt. Ppt fra start lektion 8 **Selv- og formativ evaluering**: Link til elever: http://kortlink.dk/trpx Skabelon til lærer: http://kortlink.dk/trpy Som en lidt 'sjov' og anderledes **summativ evaluering** kan eleverne sendes ud på en skattejagt med deres mobiler. Vi har tænkt på den app, der hedder Woop, som er udviklet af spejderne. I denne video kan man se, hvordan man kan anvende app'en og lave en skattejagt: kortlink.dk/trdf En Idé kunne være, at oprette 9 poster omkring skolen, hvor begreberne fra Quizletten bliver testet. Altså at begreberne står som spørgsmål og definitionerne som svarmuligheder.	link til arbejdsark: kortlink.dk/trqh

8-9	Efter Lektion 7 skal vi opsamle og forklare, at det de har været igennem i de første 7 lektioner er at opdele opgaven med at modellere et baskethold vha. 5 trin. Repetition af funktionsbegrebet med henblik på at lave algoritme til beregning af pris. Link til pptx: kortlink.dk/tufr  Kasse til beregning af pris:  **En spiller skal sælges.** Find ud af hvilke funktioner der skal implementeres og hvad der skal justeres for at der kan sælges en spiller, og basketholdet bliver opdateret. Hvilke dele indgår i en salgspris og hvordan kan det implementeres i en algoritme.	link til arbejdsark: kortlink.dk/tuev
10-11	Lav dit eget system: Overfør DreamTeam til et andet system. Sammenlign fx DreamTeam Version 0 med Film Version 0. Redegør undervejs for hvilke ændringer der skal til i koden mht. attributter og funktioner.	Link til arbejdsark: kortlink.dk/tuez