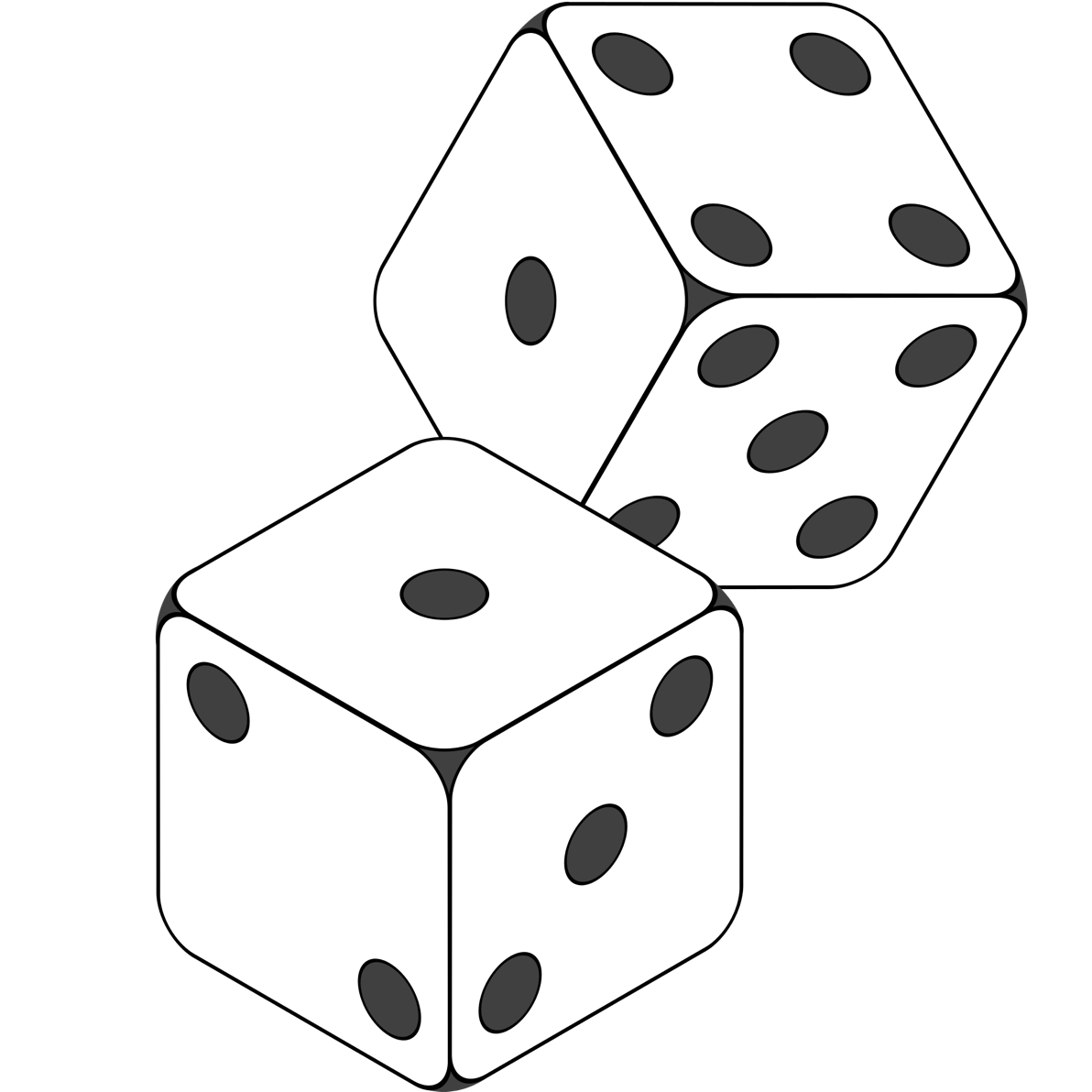


Workshop med spil i

HTML, CSS og JavaScript



Terningespillet 10.000

*Man spiller om at nå 10.000 point først.*

#### Hvordan får man point?

En ener giver 100 point

En femmer giver 50 point.

Tre ens giver antal øjne på terningen gange 100. F.eks. giver 3 4'ere 400 point.' Fire ens giver som 3 ens X 2. F.eks. giver 4 3'ere 300 X 2.

Fem og seks ens giver som 3 ens X 2 for hver ekstra terning. F.eks. giver 5 2'ere 200 X 2 X 2. 6 2'ere giver 200 X 2 X 2 X 2.

Enere er dog anderledes på denne måde tre enere giver 1000 fire enere giver 2000 fem enere giver 4000 Seks enere giver 10.000 point.

#### Hvordan tælles point sammen?

Hver gang man har slået, skal man mindst lægge en pointgivende terning til side.

Hvis man fx har slået (1,2,4,4,6,6) er 1 den eneste pointgiven terning, og skal derfor lægges til side. Er der flere pointgivne terninger, må man selv vælge hvor mange man lægger til side, dog altid mindst en.

Ved slag hvor der er tre eller flere ens, er disse også pointgivende, så de kan også lægges til side. Fx. (2,4,6,3,3,3) er (3,3,3) det eneste pointgivne, og derfor skal alle tre terninger lægges til side.

Ved flere ens, må man selv bestemme hvor mange der ligges til side, dog skal der altid mindst være tre terninger der lægges til siden, da der jo skal være tre for at de giver point. Fx. (1,2,2,2,5,6) kan man lægge (1) til side, (5) til side, (1,5) til side, (2,2,2) til side, (2,2,2,1) til side, (2,2,2,5) til side eller (2,2,2,1,5) til side.

Har man lagt alle 6 terninger til side som pointgivne, skal man slå igen med alle sammen. De nye slag lægges sammen med den sum man havde fra før.

Slår man på et tidspunkt et slag uden pointgivne terninger, nulstilles den sum man har opbygget, og terningerne gives videre til næste spiller.

For at kunne gemme sine point, skal man på listen. Første gang man skal på listen skal man mindst have 1000 point. Derefter kan man komme på listen med alle pointgivne slag.

Så længe man slår et pointgivende slag, og er på listen, kan man frit beslutte sig for at stoppe, og få ført sine point på listen, hvorefter turen går videre.

Når en spiller har over 10.000 point i alt, har resten af deltagerne et forsøg **til** at komme op på flest point og vinde.

#### Etternes point

1 = 100

1,1 = 200

1,1,1 = 1000

1,1,1,1 = 2000

1,1,1,1,1 = 4000

1,1,1,1,1,1= 10.000

1,2,3,4,5,6 = 1000

Man kan tage to ettere til side i første kast og så **i** næste kast kaste endnu en etter - dette giver 300 point, ikke 1000.

Kilde: https://terningspil.dk/regler-til-10000-terningspil/

**Opgave 1 - Opsætning**

For at du kan komme i gang skal du installere:

1. Visual Studio code (VSCode) er gratis og kan findes her: <https://code.visualstudio.com/>
2. Chrome browser (som du sikkert allerede har): <https://www.google.com/intl/da_ALL/chrome/>
3. Download billeder og andet materiale her: <https://goo.gl/fp7vE4>
4. Spillet kan findes her: <http://Mdam.my.eal.dk/game/10000.html>

### **Opgave 2: Brugergrænsefladen**

Der skal være plads til følgende på brugergrænsefladen:

6 terninger

Spillerens score i denne runde Spillerens score "på listen"

Knap til at foretage nyt slag med terningerne

Knap til at registrere scoren på listen og starte en ny runde.

Til dette bruger vi HTML.

I HTML bruger vi div-tag som er en usynlig kasse, hvor man kan ligge elementer i.

Vi bruger class for at kunne designe kasserne. Class henviser til en style som indeholder baggrunds-billede, farver, tekststørrelse og meget mere.

Reff: <https://www.w3schools.com/html/default.asp>

Start HTML

<html>

<header>

<style>

</style>

</header>

<body>

</body>

</html>

**Bord til terninger**

Vi bruger et div-tag der bliver lagt ind under <body> til at lave bordet med. Vi tildeler tagget en class, hvor vi kan style/designe ens div

<div class="diceTable">

**Visning af terningerne**

For at lave en terning bruger vi et tag kaldet div. Denne skal placeres inde på bordet og derfor inde i div-tagget der blev lavet til bordet

<div id="dice1" class="dice1"></div>

Der skal laves 6 terninger som hver har forskelligt id.

For at vi kan sætte billeder på bordet og terninger, så vi kan se noget på skærmen skal vi lave en class som ligger under style-tag.

Vi bruger diceTable (det er vigtigt med store og små bogstaver) for at designe bordet.

.diceTable{

height:620px;

width: 950px;

display: block;

background: url("table.jpg") -40px -200px;

float:left;

}

1. Download billeder til bord og terninger her: <https://goo.gl/fp7vE4>
2. Punktum i starten af navnet, betyder at man kan bruge den som en class i HTML.
3. display: block; betyder at den skal vises som en kasse der fylder det der er sat.
4. background henviser til billedet og herefter kommer der lidt koordinater der flytter baggrunden
5. float: left få efterfølgende elementer til placere sig ved siden af kassen i stedet for under kassen

.dice1{

background: url("dices.png") -0px -65px;

width: 65px;

height: 65px;

display:block;

background-repeat: no-repeat;

float: left;

margin:1px;

}

Eksempel på hvordan style til terning 1 ser ud.

1. Terningerne skal også vises. På ovenstående eksempel kan du se hvordan man viser terning med værdien 1 (et øje). For at få de andre værdier skal man indstille baggrund til et andet område.

Background: url(”dices.png”) -0px -65px de sidste to værdier betyder at man fra venstre starter I 0 og fra toppen starter man I -65. Det betyder at vores start-koordinat for det øverste venstre hjørne er flyttet ind på billedet med terninnger så det passer med starten af terningen med værdien 1.

Da alle terninger ligger på linje vil top værdien altid være -65, men for at få terning med værdien 2 skal jeg starte længere til venstre, så jeres opgave er at lave en visning af alle terninger ved siden af hinanden

1. Placer terninger i midten af bordet ved at lave en div uden om alle terninger og placere den vha. class.
2. position:absolute;
3. left:250px;
4. top:250px;

Position absolute gør at man kan placere elementer oven på andre elementer

Left og top placere elementet

**Scoreboard**

<div class="blackboard" >

<span class="title">Nuværende runde</span><br/>

<input type="text" disabled="true" id="currentRound"/>

**Knapper**

<button>Kast terningerne</button>

Placeres under scoreboardet (div-tag).

#### Tip

1. Du kan hente billederne af terningerne og bordet <https://goo.gl/fp7vE4>
2. Giv alle elementerne et ID, så du kan referere til dem senere.
3. Husk at HTML-filen skal hedde index.html og placer hele spillet i en mappe.

Prøv at bygge brugergrænsefladen i html og se, hvordan app'en ser ud i en browser (google chrome).

## **Opgave 3: Terningeslag**

Slå tilfældige slag med de seks terninger.

Når brugeren åbner app'en, skal de seks terninger vise tilfældige værdier. For hver værdi skal du vise det tilhørende billede.

Vis skal bruge JavaScript for at udfører denne opgave og den bedste måde at lave det på er i en fil for sig selv. Det giver os mulighed for at finde eventuelle fejl lettere end hvis vi lavede det i samme fil som HTML.

#### Tip

Du lavet et tilfældigt tal mellem 1 og 6 i Javascript således: Math.floor(Math.random() \* 6) + 1

Brug en for-løkke til at lave et tilfældigt terningeslag med hver af terningerne.

Sørg for at app'en husker hvilken værdi hver terning viser (det skal bruges senere, når vi skal beregne brugerens point). Det kan du eksempelvis gøre ved at have et Javascript array med seks indgange, hvor hver indgang repræsenterer en ternings værdi.

1. Opret en ny fil og kald den game10000.js. placer filen i samme folder som game10000.html
2. I HTML filen lav følgende tag inde i <header> for kunne bruge JS-filen når man kører spillet.

<script src="game10000.js"></script>

Reff: <https://www.w3schools.com/tags/att_script_src.asp>

1. For at teste at det virker kan man lave en alert.

alert("wohoo");

1. Når du åbner din html-side skal den komme med en besked der siger ”wohoo”. Ellers er der en fejl. Hvis den virker, så spørg om sidemanden kan få det til at virke og hjælp hvis der er behov for det.
2. Nu skal vi have fat i en af terningerne. Det kan vi gøre ved at pege på terningen med en funktion der via id på terningen kan hente objektet. Funktionen hedder ”**document.getElementById**(”Dette\_Er\_Mit\_Id”). Document henviser til hmtl.

var Dice = document.getElementById("dice1");

1. ”dice1” er id på terning placeret som den 1. For at ændre terning nummer 1 til værdien 2 skal den have en anden **class.** Det an vi skifte ved at bruge className.

Dice.className = "dice2"; //setting the classname/image to correct dice value

1. Random number:

Tilfældigt tal som beskrevet før ser sådan her ud:

var rndNumber = Math.floor((Math.random() \* 6) + 1); // getting the random number 1-6

1. “dice2” (fra punkt 6) er en string. Man kan tilføje til en string ved at bruge +.

Dice.className = "dice" + rndNumber;

1. For-løkken:   
   En for-løkke er en måde at gå gennem en række af opgaver eller værdier.

Man angiver en start værdi først (værdi = 1, da det er den første terning)

Herefter kommer hvornår man skal stoppe (stop når jeg har 6 terninger) og til sidst hvordan man ”bevæger” sig i rækken (f.eks hvert eneste element eller er det hvert andet element man ser på). I dette tilfælde går vi til næste terning og det kan man skrive med i++. Det betyder det samme som i=i+1.

for(var i = 1;i<=6;i++){ //running through the dices

//TODO: write code here

}

Reff: <https://www.w3schools.com/js/js_arrays.asp>

1. Javascript Array: Oprettelse af array kan ses i nedenstående. Et array er bare en liste af elementer, hvor det første element har placering 0 det næste 1 osv.

var DicesThisRound = []; // array with dices in current round

1. For at ligge et element i listen kan man bruge DicesThisRound[0] = element (som kan være et tal eller en tekst m.m.). I nævnte eksempel ligger man et element i den første placering af listen. Hvis man vil ligger et på placering herefter ville det være: DicesThisRound[1] = element osv. Det vi ønsker som element i denne liste er værdien af terningen. Jeg bruger slice for at hente den sidste værdi/element ud af className der repræsentere værdien af terningen. Slice(4,5) kan oversættes til: gå 4 tegn frem og giv mig værdien til og med det femte element.

DicesThisRound.push(dice.className.slice(4,5));

1. For at se hvad der er i vores array, kan vi prøve at bruge et indbygget program i chrome browseren der hedder udviklingsværktøjer.
2. Vi kan skrive til programmet ved følgende kode.

console.log(DicesThisRound);

**Animation af terningen i JavaScript.**

For at det virker som om at terningen bliver kastet. Kan vi lave en lille animation som hopper imellem tilfældige visninger.

Da det skal bruges mere end en gang ønsker vi at kunne kalde animationen fra en funktion.

function MakeAnimation(Dice) //Dice is the object I want to animate

{

var id = setInterval(frame, 150); //interval is the time between each frame is showed (150/1000 seconds)

var frames = 0 // initializing the frames/images to 0

function frame(){

if (frames>8) //continue until you reach a given number of frames/images

{

clearInterval(id); //this will stop the animation

// TODO: setting the class name/image to the final dice value

}

var rndNumber = //TODO: setting the random number for the dice-animation

//**TODO**: changing image on dice to make animation

Frames++; //Adding to frames so it will stop when it reaches limit set above.

};

}

Reff: <https://www.w3schools.com/howto/howto_js_animate.asp>

For at kalde vores funktion til animationen skal vi placere et kald i vores løkke fra punkt 9.

Vi skal dog definere hvilken terning/Dice vi ønsker at animere først. For at kunne gøre det skal vi hente det element vi Id først og herefter kalde animationen med elementet som parameter.

var Dice = document.getElementById("dice"+i);

MakeAnimation(Dice);

**Opgave 4: Hold terninger**

Når brugeren trykker på en terning, skal den markeres som holdt. Rent grafisk sker der ved at gøre terningen svagere i farven. Det bliver også kaldt mere gennemsigtig.

Når brugeren trykker på terningen igen, skal markeringen fjernes.

#### Tip

Du kan skal lave en funktion der hedder SetDiceONOFFHold og som har en parameter med elementet dice

function SetDiceONOFFHold(dice)

Du kan markere terningen ved at ændre dens opacity (oversat betyder det gennemsigtighed) med følgende CSS: opacity: 0.3

dice.style.opacity = '0.3';

Hvis du vil sætte terningen aktiv igen, skal værdien ændres fra 0.3 til 1.

App'en skal huske, hvilke terninger der er markeret, så de ikke kommer med i det næste slag.

Du kan bruge et Javascript-array med seks indgange til at huske det.

var DiceOnHold = []; // Dices that is on Hold and can't be re-activated

En nem made at bare tilføje et element bagerst i en liste som et array er en funktion der hedder push.

DiceOnHold.push(dice.id)

Hvis terningen er svagere i farven, skal du gøre den normal igen, men hvis den er normal skal den blive svagere.

Senere skal vi lave det så, man kun kan vælge terninger fra samme runde og ikke dem der måske var blevet svagere i farven fra forrige runder.

Nu skal vi have sat functionen på når man klikker på en terning og det gør vi i HTML filen.

Ved at sætte et event-kald, altså et kald der sætter en aktion i gang der hedder onclick kan vi aktivere koden inde bag ved. Her er et eksempel på terning et med kaldet på.

<div id="dice1" onclick="SetDiceONOFFHold(this)" class="dice1"></div>

Vi burger “this” til at overføre hele elementet med terningen til funktionen, så derfor vil det være det samme kald for alle terningerne. F.eks. vil terning 2 se således ud.

<div id="dice2" onclick="SetDiceONOFFHold(this)" class="dice2"></div>

### **Opgave 5: Beregn score**

Når brugeren trykker på knappen til at foretage det næste slag, skal scoren af det forrige slag beregnes.

Implementer reglerne fra første side. Hvis du synes, der er mange regler, og det tager lang tid, så start bare med et par stykker af dem.

Når scoren er beregnet, skal den lægges til den aktuelle rundes point. Det felt, der viser rundens point, skal opdateres.

#### Tip

Brug array'et med terningernes værdier til at se, hvad brugeren har slået og til at regne ud, hvor mange point han skal have.

Der skal være en variabel, som holder styr på, hvor mange point spilleren har i den aktuelle runde.

1. Først laver vi en funktion til at udregne point.

function checkForPointsForOneRound()

1. Udregning af scorer:

For at gøre det lettere at sammenligne laver vi vores array om til en string(række af tegn) på følgende måde

var DicesThisRoundString =DicesThisRound.toString();

Herefter kan vi tjekke indholdet ved at sammenligne tekster der stemmer overens med pointgivende indhold. Herunder er der vist to af de slag der kan give point.

if(DicesThisRoundString == "1,2,3,4,5,6") // straight

currentRound+=1000;

else if(DicesThisRoundString == "1")

currentRound+=100;

Du skal liste alle slag der giver point bortset fra dem hvor der er tre ens eller flere terninger med samme værdi.

1. Vi mangler nu at få alle dem med tre ens til at fungere, men det er lidt svært lige at lave en god funktion til så den kan i få i nedenstående.

function GetDiceValue(DiceValue) // get the value from dice values with 3 or more of the same > 1

{

DicesThisRound.sort();

var DicesThisRoundString =DicesThisRound.toString();

if(DicesThisRoundString.indexOf(DiceValue+","+DiceValue+","+DiceValue)>=0)

{

var RoundValue = DiceValue \* 100; // 3 dices with same numbers

if(DiceValue==1)

RoundValue =1000

// 6 dices with same value

if(DicesThisRoundString.indexOf(DiceValue+","+DiceValue+","+DiceValue+","+DiceValue+","+DiceValue+","+DiceValue)>=0)

{

RoundValue = RoundValue \* 2 \* 2 \* 2

if(DiceValue==1)

RoundValue =10000

}

// 5 dices with same value

else if (DicesThisRoundString.indexOf(DiceValue+","+DiceValue+","+DiceValue+","+DiceValue+","+DiceValue)>=0)

{

RoundValue = RoundValue \* 2 \* 2

if(DiceValue==1)

RoundValue =4000

}

// 4 dices with same value

else if (DicesThisRoundString.indexOf(DiceValue+","+DiceValue+","+DiceValue+","+DiceValue)>=0)

{

RoundValue = RoundValue \* 2

if(DiceValue==1)

RoundValue =2000

}

currentRound+=Number(RoundValue);

RemoveDicesFromThisRound(DiceValue);

}

}

Det der sker i funktionen, er at vi sortere værdierne, så det bliver lettere at finde dem der er ens, da de så vil komme lige efter hinanden.

Først undersøger vi om der overhovedet er nogen med den værdi/diceValue jeg har ind lige nu.

Hvis jeg kan bekræfte der er 3 eller flere ens vil jeg undersøge de tilfælde der kan være.

Fra reglerne kan vi se at normale tilfælde vil 3 ens give værdien af terningerne x100 og herefter med 4 ens give 3 ens x2, 5 ens = 3ens x2 x2 og 6 ens = 3 ens x2 x2 x2;

Der dog et tilfælde med 1’ere der afviger, så her har jeg lavet en værdi til dem.

Jeg starter med at se på det mest usandsynlige tilfælde og går herefter frem mod det mest sandsynlig. Hvis ingen af mine tilfælde bliver ramt, så har jeg kun 3 ens.

Til sidst fjerne jeg de ens terninger fra listen, så den efterfølgende udregning af specialtilfælde som i har lavet tidligere kan ske.

Vi kalder denne funktion fra den function i lavede før: checkForPointsForOneRound og vi kalder den for hver værdi der kan være på en terning ved hjælp af en løkke.

function checkForPointsForOneRound() //check for points with dice value 1 and 5 and straight

{

for (var i = 1; i<=6; i++) // dices with same value check for each value 2-6, both included

{

GetDiceValue(i) // check if dices of same value gives points

}

**Opgave 6: Nyt slag**

Når brugeren har valgt de terninger, han vil holde, kan han slå igen. Han skal så naturligvis kun slå med de terninger, han ikke har markeret.

#### Tip

Brug array'et med markeringer til at holde styr på, hvilke terninger der skal slås, og hvilke der skal blive

## **Opgave 7: Markeringer ved flere slag**

Når brugeren har flere slag, er der flere ting, du skal holde øje med.

Brugeren skal ikke kunne fjerne markeringen fra terninger, som han har markeret i et tidligere slag.

#### Tip

Du kan enten bruge endnu et array til at holde øje med tidligere markeringer, eller du kan bruge en anden værdi i det eksisterende array for at skelne mellem tidligere og aktuelle markeringer.

**Opgave 8: Afslutning af runden**

Spillet skal tælle, hvor mange runder det tager spilleren at komme til 10.000. En runde kan slutte på to måder:

Hvis spilleren ikke kan lægge flere terninger til side som beskrevet i reglerne, er runden slut, og hans rundepoint nulstilles.

Hvis spilleren vælger at sætte sine point på listen, overføres de fra den aktuelle runde til listen, og en ny runde starter.

#### Tip

Hvis du ikke har lyst til at implementere alle reglerne endnu, så kan du nøjes med nogle stykker.

Husk, at man skal have mindst 1.000 point i en runde for at komme på listen første gang.

## **Opgave 10: Alt det andet...**

Spillet er egentlig færdigt nu - men det bliver kedeligt at spille, når man kun er en spiller.

Udvid spillet, så man kan være flere spillere, som spiller mod hinanden. 10.000 er et socialt spil, så alle spiller på samme telefon. Telefonen går på tur mellem spillerne på samme måde som et raflebæger ville gøre.

Hver spiller skal have sin egen liste, og det handler ikke længere om, hvor mange runder man bruger på at nå til 10.000. Man skal blot være den første til at få 10.000 point på listen.

Inden spillet sættes i gang, skal man vælge, hvor mange spillere der skal være med. Hver spiller skal have et navn.

Den sidste udvidelse er at lade spillerne tage et billede af hinanden og bruge spillernes billeder i stedet for deres navne.



Om MMD

Tidligere kunne computere kun benyttes af folk, der var specialuddannede i at give dem kommandoer. I dag er computere over alt: i smartphones, biler, fjernsyn, medicinsk udstyr og meget mere, og de er blevet så lette at betjene, at man ikke behøver at forstå, hvad der sker.

Derfor lærer mange i dag kun at bruge de digitale enheder, fordi de ikke tror, det er nødvendigt at forstå dem.

Når man ikke forstår, hvordan computeren virker, kan man ikke beherske den. Man bliver frataget sin innovationsevne og begrænset til at løse de opgaver, andre har lært den, i stedet for at mestre computeren, og få den til at løse de opgaver, man selv har.

Derfor er det nødvendigt at flere lærer at programmere.

Det handler ikke om, at alle skal være softwareudviklere. Det handler om, at programmering i dag er lige så grundlæggende en færdighed som læsning, skrivning og regning. Det er en forudsætning for, at man kan udnytte mulighederne i digitaliseringen - både privat, i uddannelsen og på arbejdsmarkedet.

Ændrer vi ikke udviklingen, vil vi om få år stå med et teknologisk A- og B-hold, hvor en lille gruppe med teknologisk indsigt bestemmer, hvilke muligheder den store gruppe skal have med deres værktøjer.

Hos MMD mener vi, at alle fortjener at kunne programmere, og vi vil gerne hjælpe jer med at lære det.