JAVASCRIPT BASICS

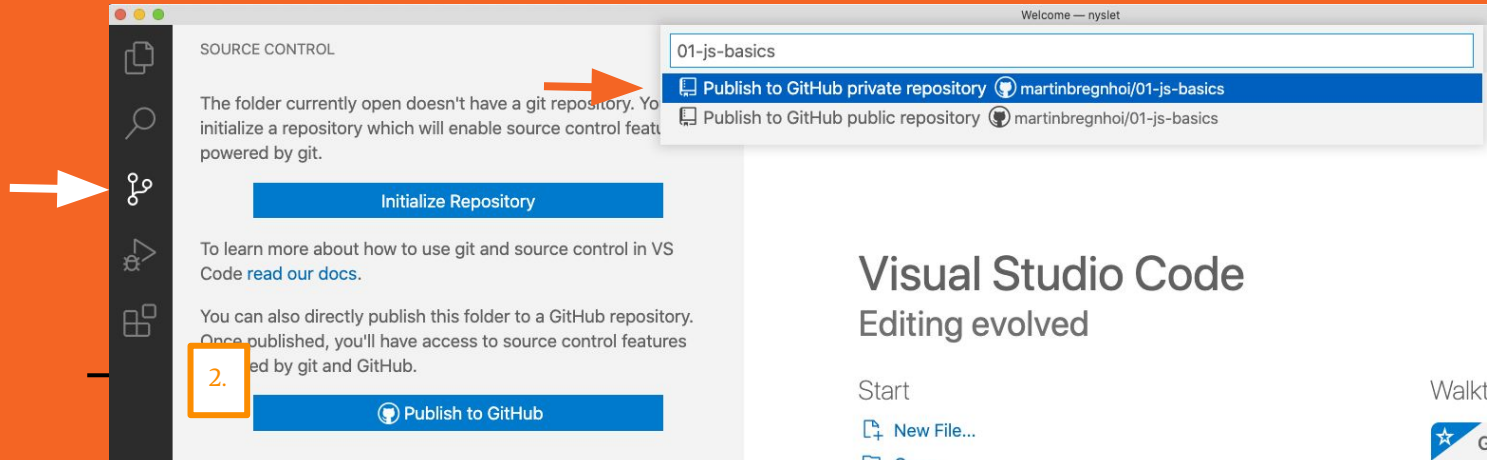
Øvelse 0 - opret en mappe til dagens øvelser, og publicer den til Github

1. Lav en mappe på din computer. Kald den f.eks. **tema7**
2. Lav en undermappe øvelserne.
3. Åbn denne undermappe i VS Code
4. Lav et html-skelet i mappen.
5. Publicer to GitHub. (se næste slide…)

Øvelse 0 fortsat - publicer til Github

I VS Code kan du nemt oprette et nyt repository på Github (hvis ellers VS Code er sat rigtigt op…) således:

1. klik på Source Control
2. klik på Publish to Github og udfyld feltet
3. Tjek github.com for at se om dit repository er oprettet.



Hvad vi kommer igennem:

1. Lidt generel teori om programmeringssprog og javascript
2. Erklæring af variabler og konstanter
3. Indbyggede metoder til tekster og tal
4. if-statements
5. Metoder til interaktion
6. Funktioner
7. Øvelser

Programmeringssprog:

- Forskellige sprog, som bruges på computeren
- Sprog man bruger til at få computeren til at udføre opgaver
- Indeholder anvisninger/kommandoer til computeren
- Mest anvendte programmeringssprog
- Wikipedia, definition af programmeringsprog

Programmeringssprog og andre computersprog

- html er **ikke** et programmeringssprog
  - det er et **opmærkningssprog**
  - hvilke elementer har vi på en webside (**DOM'**en)?
- css er **ikke** et programmeringssprog
  - det er et **layoutsprog**
  - hvordan skal elementerne på en webside præsenteres?
- **javascript** er et programmeringssprog**!**
  - besked til computeren i et program: hvad skal der ske?

Wikipedia: forskellige typer computersprog:

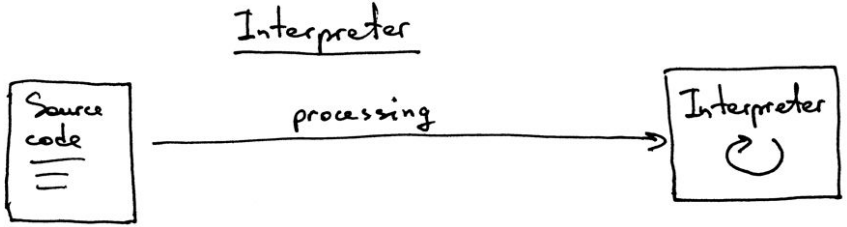
Programmeringssprog

- Et programmeringssprog er et kunstigt sprog
  - Simple og meget formelle I forhold til naturlige/menneskesprog
- Et computersprog er beskrevet i form af nogle regler
  - Syntax: hvilke regler har vi for hvordan ordene kan sættes sammen
  - Semantik: Hvilken betydning har sætningerne
  - Regler for syntax og semantik beskrives i referencemanualer

Difference between syntax and semantics of programming languages:

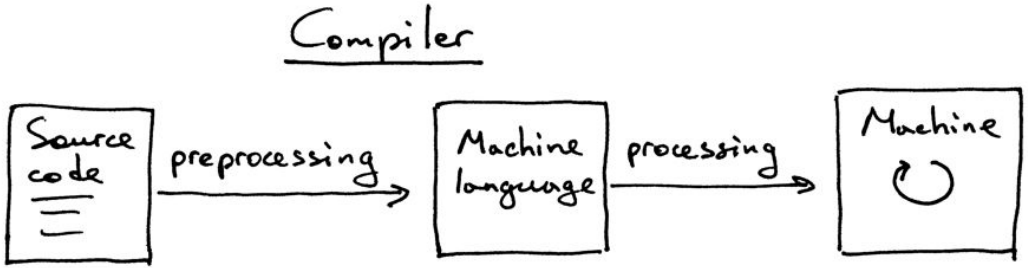
Fortolkede sprog

- Programmet skrives og gives til en fortolker/ interpreter.
- Fortolkeren oversætter sætningerne en ad gangen til maskinsprog
- Computeren udfører sætningerne efterhånden, som den får dem.
- Eksempler: javascript, php, python, ruby, perl



Oversatte sprog

- Programmet skrives og gives til en oversætter/compiler.
- Compileren oversætter det hele til maskinsprog
- Computeren udfører det oversatte program.
- Eksempler: Java, C#, Swift, SASS



Script-sprog

- Scriptsprog bruges tit som synonym for fortolkede sprog
- Det kan også betyde et programmeringssprog til småprogrammer (scripts), som fungerer sammen med et andet computersprog
- F.eks javascript, som fungerer sammen med html og css i en browser

Om javascript

- Er et **programmeringssprog**
- Er et **fortolket** sprog
- Er et **scriptsprog**
- Er et **objektorienteret** sprog
- Alle browsere har en indbygget javascript-**fortolker** (engine)
- Javascript kan manipulere web-dokumentets html-elementer (**DOM**)
- Javascript kan manipulere html-elementernes layout (**CSS**)
- Et standardiserings-organisation, **ECMA** tager sig af beskrive javascript
- Liste med alle versioner af ECMA-script
- Wikipedia om Javascript

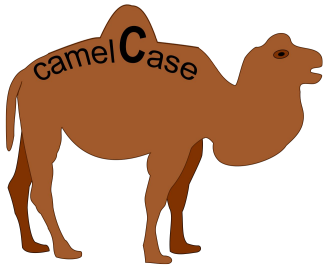
javascript - et objektorienteret sprog

- Javascript har mange indbyggede objekter
- Det er også muligt at skabe egne objekter i javascript
- Et eksempel på et indbygget objekt i javascript er Math
- Math er et indbygget objekt, som har en lang række egenskaber og metoder (vi brugte nogle af dem på tema 4):
  - Math.PI (egenskab)
  - Math.random() (metode)
- Dot-notation **.** (punktum) mellem objektnavn og egenskab/metode

ERKLÆRINGER AF VARIABLER I JAVASCRIPT

Variabel-erklæringer

- En variabel er et “navn”, som man kan tildele en værdi.
- En variabels værdi kan senere i programmet ændres til noget andet.
- Variabler erklæres typisk øverst i scriptet.



Eksempler:

```
let forNavn = "Martin"; // tekst
let alder = 58; // tal
let enlig = false; // boolean
let køn; // ingen værdi endnu
```

Variablerne forNavn, alder og enlig har fået tildelt værdier af tre forskellige datatyper: *tekst, tal* og *boolean*.

Konstanter

Tit har man brug for navne på værdier, som **ikke** skal ændres.

Her bruger man konstanter (const) i stedet for variabler, f.eks:

```
const moms = 0.25; // momsen er konstant og skal ikke ændres i programmet
```

Konstanter kan ikke ændre værdi, når de først er erklæret, men bruges ellers præcis ligesom variabler

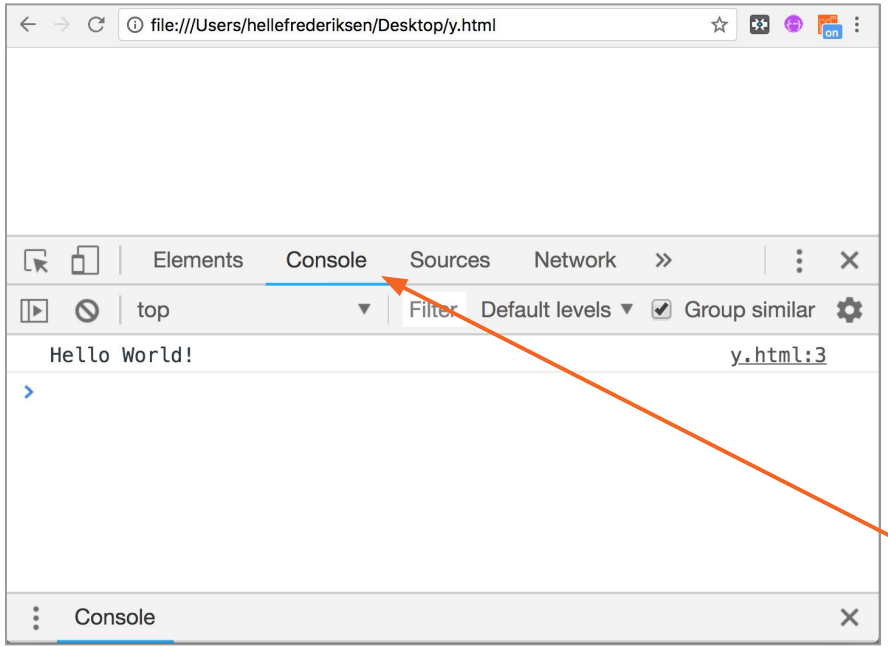
OPERATORER OG WINDOW-METODER

Window-objektet

- Browserens vigtigste indbyggede objekt er **window-objektet**
- **window** er browserens vindue - med eller uden html-elementer (DOM)
- window-objektet har en lang række **egenskaber/properties** og metoder, som kan tilgås fra javascript, f.eks:
- window.**console.log()** - skriver det som man angiver i parentesen i konsollen
- window.**alert()** - åbner en dialogboks i browservinduet med en tekst
- Bemærk at *window* er underforstået, og udelades derfor oftest.
- **BENYT KUN** console.log()**!**

  https://www.w3schools.com/jsref/obj_window.asp

window.**console.log()**



Indbygget metode, som kan udskrive en værdi i browserens console-vindue

Eksempel:

```
let greeting = "Hello World";
console.log(greeting);
```

I Chrome (og de fleste andre browsere):

Højreklik i browservinduet -

Vælg inspect/undersøg

Vælg fanebladet *Console*

Sammensætning af tekster

Efter erklæringen, kan variablerne ændres i statements

```
let minTekst = "Her er en tekst";
minTekst = minTekst + " , som fortsætter her!";
```

+ er en operator til sammensætning af tekster (text concatenation)

+= er en anden operator til concatenation, som tager den værdi variablen har i forvejen, og tilføjer højresiden.
Linje to kan derfor også skrives som:

```
minTekst += " , som fortsætter her!";
```

Template litterals

- Er et alternativ til **+-** operatoren (se forrige slide), når tekst skal sammensættes (konkateneres)
- `…` (accent grave) sættes rundt om hele teksten
- I teksten kan man så indsætte variabler eller udtryk ved at sætte $ foran og {...} omkring disse, f.eks:

```
let minTekst = "her er en tekst";
minTekst = `${minTekst}, som fortsætter her`;
```

Template literals (Template strings), MDN web docs

Indbyggede metoder til tekster

- I js kan man meget andet end blot at sammenlægge tekster
- Tekst-variabler har en række metoder
- Eksempler (søg efter flere, når du får brug for det):

```
let tekst = "Eksempel på en tekst";

let len = tekst.length; // tekstens længde (her 20)

tekst = tekst.toUpperCase(); //tekst er nu “EKSEMPEL PÅ EN TEKST”

tekst = tekst.toLowerCase(); //tekst er nu “eksempel på en tekst”
```

JavaScript Stings, W3schools: https://www.w3schools.com/jsref/jsref_obj_string.asp

Tal-operatorer

+ Addition
F.eks: **pris = indkobsPris + moms;**

- Subtraction
F.eks: **engrosPris = pris - moms;**

*** Multiplication**
F.eks: **totalPris = antal * pris;**

/ Division
F.eks: **pris = totalPris/antal;**

++ Increment (læg 1 til)
F.eks: **antal++;**

-- Decrement (træk 1 fra)
F.eks: **antal--;**

Statements til beregning af tal

- I **statements** kan man udregne tal, for eksempel:

```
let pris = 100;
const moms = 0.25;
pris = pris + pris*moms;
```

Indbyggede metoder til beregninger

Math, som har som tidligere nævnt en række nyttige metoder til tal:

```
let tal = 3.5;
tal = Math.round(tal); // tal er nu afrundet til (4)
tal = Math.pow(tal,2); // tal opløftes til 2. potens (16)
tal = Math.random(); // tal er et tilfældigt tal mellem 0 og 1 (fx. 0.843219827740112)
tal = Math.round(Math.random()*10); // tal er nu et heltal mellem 0 og 10 begge inklusive
erTal = isNaN(tal); // er sand, hvis tal ikke er et tal, her er den false (NaN - Not a Number)
```

Math, W3schools: https://www.w3schools.com/js/js_math.asp

BETINGELSE/CONDITION
IF-STATEMENT

if-statement

Når et eller flere statements **KUN** skal udføres, hvis en **betingelse** er opfyldt:

```
if(betingelse/condition){
        statement1;
        statement2;
}
```

betingelse er en boolsk værdi (sand eller falsk) eller et boolsk udtryk, f.eks:

```
if(alder < 18) {
        console.log("Barn");
};
```

< er en **logisk operator**
alder < 18 er en **betingelse/condition**

if else else if: w3school: https://www.w3schools.com/js/js_if_else.asp

Logiske operatorer

`==`	Er lig med
`===`	Strict equality (samme datatype OG indhold)
`!=`	Er forskellig fra
`>`	Er større end
`<`	Er mindre end
`>=`	Er større end eller lig med
`&&`	Og (mellem to betingelser, som **begge** skal være sande)
`||`	Eller (mellem to betingelser, hvor mindst **en** skal være sand) alt+i på mac-keyboard
`!`	Ikke (negation - foran en betingelse, som *ikke* må være sand)

if-statement - flere eksempler

```
if(alder >=18) {
        console.log("Voksen");
}

if(drik == "snaps"){
        console.log("indeholder 40% alkohol");
}

if(drik !="sukkerfri"){
        console.log("indeholder sukker");
}

if(drik !="alkoholfri" && alder<18){
        console.log("må ikke serveres");
}

if(alder<18 || alder>67){
        console.log("Ikke erhvervsaktiv alder");
}
```

if-else-statement

Et eller flere statements skal KUN udføres, hvis en betingelse er opfyldt - og ELLERS skal nogle andre udføres:

```
if( betingelse ){
        statement1;
else {
        statement2;
}
```

Eksempel:

```
if( alder < 18 ){
        console.log("Barn");
} else {
        console.log("Voksen");
}
```

forgrenede if-statements

Man kan konstruere en if-sætning med mange else-grene:

```
if(betingelse 1){
        statement1;
} else if(betingelse 2){
        statement2;
} else if(betingelse 3){
        statement3;
} else{
        statement4;
}
```

Eksempel på forgrenede if-statements

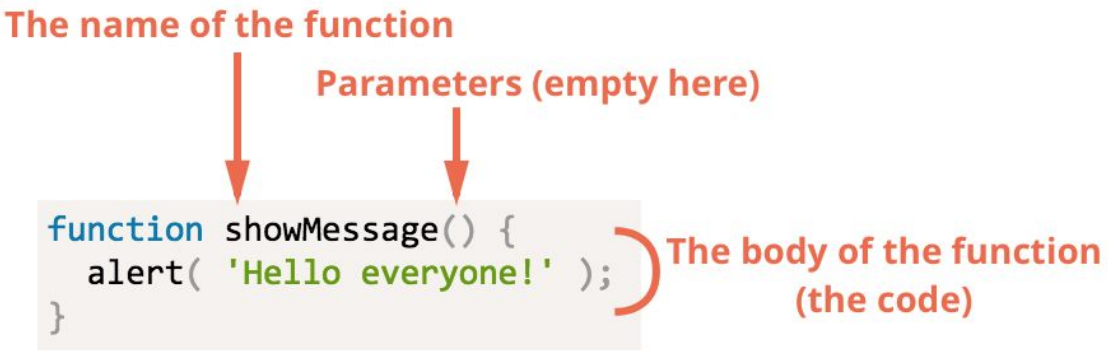
Eksempel:

```
if(alder < 6) {
        console.log("førskolealder");
} else if(alder < 16) {
        console.log("skolealder");
} else if(alder < 67) {
        console.log("erhvervsaktiv alder");
} else {
        console.log("pensionsalder");
}
```

FUNKTIONER

Funktioner

En funktion er et stykke isoleret kode.
Funktionen skal først erklæres/defineres:



Når en funktion bliver erklæret, sker der ingenting!
Først når den **kaldes**, udføres funktionen, f.eks:

```
showMessage();
```

DRY - Don’t Repeat Yourself

- Med (smarte) funktioner kan man undgå at gentage kode
- Programmet bliver modulopbygget og mere overskueligt
- Funktioner kan også ofte genbruges i andre programmer
- Derfor forsøger man **altid** at oprette funktioner, der hvor man kan

Funktion med parameteroverførsel

```
function visBesked(txt){
        console.log(txt);
}

visBesked("Goddag");


let besked = "Farvel";

visBesked(besked);
```

Erklæring af funktionen
Funktionen vil vise en værdi i konsollen i inspectoren.
txt er et variabelnavn som funktionen bruger, det kaldes også et **parameter**

Funktionen kaldes med værdien “Goddag” som argument (parameter). Inde i funktionen får variablen **txt** denne værdi, og den værdi (her “Goddag”) logges så i konsollen.

Funktionen kaldes med variablen **besked** som argument.
besked har værdien “Farvel” fra linjen før.
funktionen får variablens værdi som argument
I funktionen får variablen txt så denne værdi, og den logges så i konsollen.

Funktion med med flere parametre

```
function gangTal(tal1, tal2){
        let resultat = tal1  * tal2;
        console.log(resultat);
}
gangTal( 3, 4 );
```

Erklæring af funktionen
tal1 og **tal2** er to **parametre**.
(Funktioner kan sagtens have flere end 2 parametre)
resultat indeholder de to parametre ganget med hinanden.
Funktionen vil vise værdien af **resultat** i konsollen.

Funktionen **kaldes** med værdierne 3 og 4 adskilt af et komma.
I funktionen får variablen *tal1* værdien 3 og *tal2* værdien 4, og tallene ganges med hinanden og vises i konsollen.

Funktioner, som returnerer en værdi

```
function visBesked(message){
        let first = "Info: ";
        return `${first} ${message}`;
}

let besked="Kamilla underviser i morgen";
console.log(visBesked(besked));

let info="Martin underviser i dag";
console.log(visBesked(info));

console.log(visBesked("Louise underviser først i næste uge"));
```

Funktionen sætter teksten “info:” foran den værdi, den modtager som parameter.
Den nye tekst returneres

I console.log kaldes funktionen med variablen besked som argument (parameter).
besked har værdien “Kamilla underviser i morgen”
inde i funktionen kaldes værdien for **message**

Øvelser

Interaktivt element

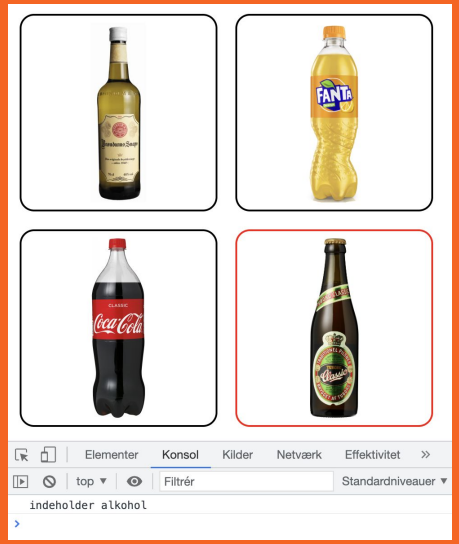
For at lave øvelse 1 skal du bruge nogle “knapper” til at klikke på.

Hvad som helst kan bruges som “knap” når der lægges en *click-event* på, som f.eks:

```
const minKnap = document.querySelector("#knapElement");

minKnap.addEventListener("click",funktionDerKaldesVedKlik);

function funktionDerKaldesVedKlik(){
    console.log("du har klikket på knappen");
}
```




```
<img src="snaps.webp" alt="snaps" />
<img src="fanta.webp" alt="fanta" />
<img src="cola.webp" alt="cola" />
<img src="øl.webp" alt="øl" />
```

Øvelse 1 - interaktivitet

- Lav en ny html-fil og gem den
- Find billeder af en øl en snaps og to sodavand og sæt disse ind i html-koden (<img src=”” alt=””>)
- Lav et program med en funktion der indeholder betingelser (if else)
- Når brugeren klikker på øllen eller snapsen, skal programmet udskrive “indeholder alkohol” i konsollen (console.log())
- Når brugeren klikker på en sodavand, skal programmet udskrive “alkoholfri” i konsollen
- **Tips**: Husk du kan bruge nøgleordet “this” i funktionen (this.alt)
- Commit og push til gitHub, når du er tilfreds med løsningen

Øvelse 2 - sig goddag

- Lav en ny html-fil til øvelsen
- Skriv et program, som i konsollen udskriver (console.log()):
  - “Godmorgen” mellem kl. 5 og kl 10,
  - “Goddag” mellem kl 10 og 18,
  - “Godaften” mellem 18 og 24 og
  - “Godnat” mellem 24 og 5.
- **Tips:** Denne javascript metode fortæller det aktuelle timetal:

  new Date().getHours()
- Commit og push til gitHub, når du er tilfreds med løsningen

Input felt

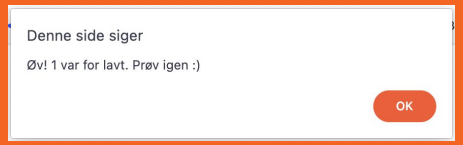
I øvelse 3 skal du bruge et input-felt:

```
<input type="number" id="tal">
```

Du kan finde ud af, hvad der står i input-feltet med flg. javaScript:

```
document.querySelector("#tal").value
```

Øvelse 3 - Gæt et tal



- Lav en ny html-fil til øvelsen
- Programmet skal generere et tilfældigt tal mellem 0 og 10 med Math.random() og bede brugeren om at gætte tallet
- Når brugeren har gættet, fortæller programmet om tallet var rigtig, eller om det var for højt eller for lavt.
- Så får brugeren lov at gætte igen, og sådan fortsætter programmet til brugeren har fundet det rigtige tal.
- Når brugeren har gættet tallet, fortæller programmet, hvor mange gæt, der blev brugt, og spørger om brugeren vil prøve igen med et nyt tal.
- Commit og push til gitHub, når du er tilfreds med opgaven