



# 1.

## JAVASCRIPT BASICS



# Øvelse 0 - opret en github-mappe til dagens øvelser

1. Opret et **tema7** repository på github
2. Lav en mappe, **tema7**, på din computer (måske under en **2-semester**-mappe, som ligger i en **KEA**-mappe?).
3. I Brackets - åbn **tema7** mappen: Klon **tema7**-repository'et til tema7 mappen.
4. Lav i **tema7** en undermappe, **undervisningsopgaver**.
5. Lav i **undervisningsopgaver** en mappe, **01-js-basics** til dagens øvelser.
6. Åbn mappen **01-js-basics** i Brackets
7. Læg et genbrugeligt, tomt html-skelet, **00-skelet.html** ind i mappen
8. Commit. Push til GitHub.

```
tema7 undervisningsopgaver
    01-js-basics
        00-skelet.html
```



# Hvad vi kommer igennem:

1. Om programmeringssprog og javascript
2. Erklæringer af variabler og konstanter
3. Indbyggede metoder til tekster og tal
4. if-statement
5. metoder til interaktion med og uden DOM:  
console.log, inputfelter og knapper
6. Funktioner
7. Øvelser (mange!)



# Programmeringssprog?

Computersprog: forskellige sprog, som bruges på computeren

Programmeringssprog:

Sprog man bruger til at få computeren til at udføre opgaver

Indeholder anvisninger/kommandoer til computeren

Mest anvendte programmeringssprog

Wikipedia, definition af programmeringssprog: [https://simple.wikipedia.org/wiki/Programming\\_language](https://simple.wikipedia.org/wiki/Programming_language)



# 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: [https://en.wikipedia.org/wiki/Computer\\_language](https://en.wikipedia.org/wiki/Computer_language)



# Programmeringssprog

Et programmeringssprog er et kunstigt sprog

I forhold til naturlige/menneskesprog er de simple og meget formelle

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

Spørgsmål på Stackoverflow: [What is the difference between syntax and semantics of programming languages:](https://stackoverflow.com/questions/17930267/what-is-the-difference-between-syntax-and-semantics-of-programming-languages)

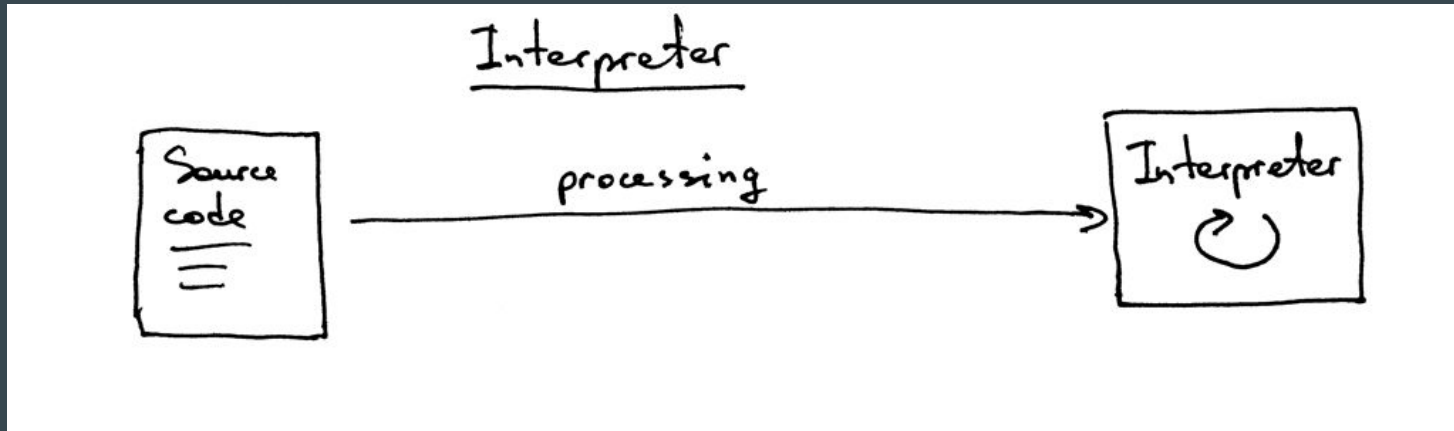
<https://stackoverflow.com/questions/17930267/what-is-the-difference-between-syntax-and-semantics-of-programming-languages>



# 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

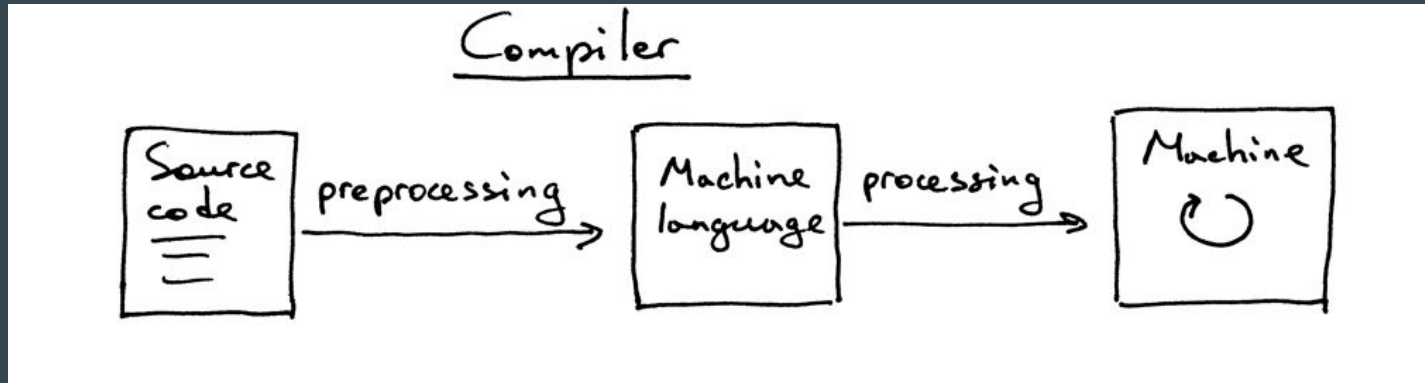




# Oversatte sprog

- 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

**Scriptprog** bruges tit blot som synonym for fortolkede sprog  
men 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

**Programmeringssprog**

**Fortolket sprog**

**Scriptprog**

**Objektorienteret sprog**

Alle browsere har indbygget en javascript-fortolker/engine

javascript kan **manipulere** webdokumentets html-elementer (**DOM**)

javascript kan også **manipulere** html-elementernes layout (**CSS**'en)

Et standardiserings-organisation, **ECMA** tager sig af beskrive javascript

Sidste versioner af [ECMA-script](#)

Wikipedia om Javascript: <https://en.wikipedia.org/wiki/JavaScript>



# javascript - et objektorienteret sprog

Javascript har mange **indbyggede objekter**

Desuden er det muligt at **skabe 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**.

F.eks:

**Math.PI** (egenskab)

**Math.random()** (metode)

• (punktum) mellem objektnavn og egenskab/metode kaldes **dot-notation**



# Opsamling: Begreber - hvad var det nu, det betød???

DOM

Opmærkningssprog

Layoutsprog

Programmeringssprog

Compilet sprog

Fortolket sprog

Script-sprog

Javascriptfortolker (hvor?)

objektorienteret sprog

indbyggede objekter

dot-notation



# 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.

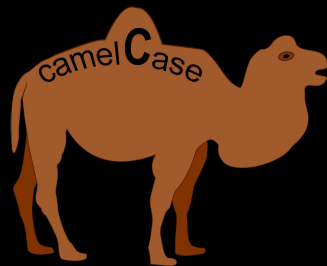
Variabler **erklæres** øverst i scriptet.

Eksempler:

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

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

Good practice:  
**camel-case** til variabelnavne





# Konstanter

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

Her bruger man **konstanter** i stedet for variabler

Erklæring af en konstant:

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

Konstanter kan ikke ændres, men kan ellers bruges ligesom variabler



# Opsamling: Ord og begreber - hvad betød de?

let

variabel erklæring

camelcase

konstant

tildeling af værdi

datatyper:

tekst/string

tal

boolean





# 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.

window.**console.log()** - skriver noget (som man angiver i parentes) i browser-consollen  
window.**alert()** - åbner et lille boks i browservinduet med den tekst, som angives i ()

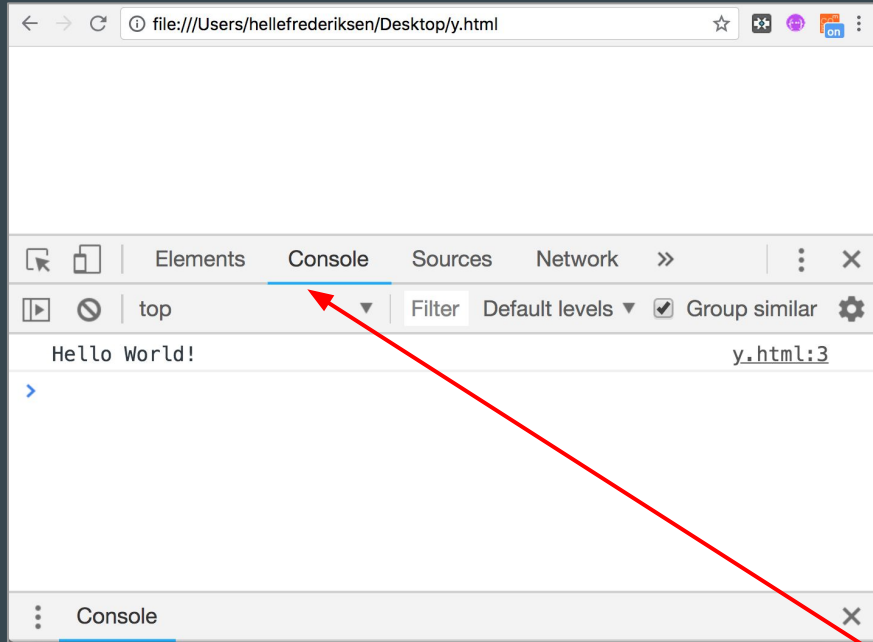
BEMÆRK! window er underforstået, og udelades.

**VI BENYTTER KUN console.log()!**

ref: [https://www.w3schools.com/jsref/obj\\_window.asp](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



# Sammenlægning af tekster

Efter erklæringen, kan variablerne ændres i **statements**

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

**+** er en operator til sammenlægning af tekster, **text concatenation**

linje 2 kan også skrives som:

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

**+=** er også en operator til **concatenation**.

Tager den værdi, variablen har i forvejen, og tilføjer højresiden



# Template literals

Når man arbejder med tekster, kan man i stedet for tekst-concatenation med +-operatoren bruge en ny syntax-mulighed i js:

## Template literals

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

Udenom hele teksten bruge `...` (accent grave)

I teksten, kan man indsætte **variabler** eller **udtryk**.

Man bruger **\$** foran og **{...}** omkring variabler eller udtryk.

**Template literals (Template strings)**, MDN web docs:

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template\\_literals](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals)



# Indbyggede metoder til tekster

I js kan man meget andet end blot at sammenlægge tekster

Tekst-variabler har en række metoder:

Eks (søg efter flere, når du får brug for det):

```
let tekst = "Eksempel på en tekst"; (eller: 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](https://www.w3schools.com/jsref/jsref_obj_string.asp)



# Tal-operatorer

## + Addition

eks:        **pris = indkobsPris + moms;**

## - Subtraction

eks:        **indkobsPris = pris - moms;**

## \* Multiplication

eks:        **totalPris = antal \* pris;**

## / Division

eks:        **pris = totalPris/antal;**

**++ Increment** ( læg 1 til )

eks:        **antal++;**

**-- Decrement** ( træk 1 fra )

eks:        **antal--;**



# Statements til beregning af tal

I statements kan man udregne tal.

Eksempel 1:

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





# Indbyggede metoder til beregninger

**Math**, som har 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](https://www.w3schools.com/js/js_math.asp)



# Øvelse 01 - udregn areal

Åbn [00-skelet.html](#), og gem den som en ny fil, **01-areal.html** i [01-js-basics](#)

Skriv i script-tagget et program, som kan udregne areal ud fra en længde og en bredde.

**Længde**, **bredde** og **areal** skal erklæres som variabler i programmet.

Resultatet skal vises i console-vinduet, og have denne form:

**Længden er 3 meter og bredden er 5 meter. Arealet er længde\*bredde**

Test programmet med forskellige værdier for længde og bredde

Commit og push til gitHub, når du er tilfreds med opgaven

tema7  
undervisningsopgaver  
01-js-basics:  
[00-skelet.html](#)  
**01-areal.html**



# Opsamling - hvad var det nu, det betød?

window-objektet

`window.console.log()`

`string`

`concatenation`

`Template literals`

`Math.round(tal)`

`Math.pow(n, m)`

`Math.random(tal)`

`tekst.length`

`tekst.toUpperCase()`

`tekst.toLowerCase()`

`isNaN(tal)`

Operatorer

tekster: `+` og `+=`

tal: `+`, `-`, `/`, `*`



# BETINGELSE / CONDITION IF-STATEMENT



# if-statement

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

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

- hvor betingelse er en boolsk værdi (sand eller falsk) eller et boolsk udtryk

eksempel:

```
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](https://www.w3schools.com/js/js_if_else.asp)



# Logiske operatører

**==** Er lig med

**!=** Er forskjellig fra

**>** Er større end

**<** Er mindre end

**>=** Er større end eller lig med

**&&** Og (mellem to betingelser)

**||** Eller (mellem to betingelser) alt+i på mac-keyboard

**!** Ikke (foran en betingelse)



# if-statement - flere eksempler\*

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

```
if(sex == "female"){  
    console.log("Kvinde");  
}
```

```
if(sex != "female"){  
    console.log("Mand");  
}
```

```
if(sex != "female" && alder < 18){  
    console.log("Dreng");  
}
```

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

\*for eksemplets skyld betjener vi os her af en binær kønsopfattelse



# 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("før skolealder");  
} else if(alder < 15) {  
    console.log("skolealder");  
} else if(alder < 65) {  
    console.log("erhvervsaktiv alder");  
} else {  
    console.log("pensionsalder");  
}
```



# Øvelse 02 - er arealet mellem 100 og 200?

Gem **01-areal.html** i en ny kopi, **02-arealTest.html**

I programmet skal du erklære to variabler: **længde** og **bredde**.

Programmet skal udregne arealet og fortælle om arealet er for lille, ok eller for stort. Resultatet skal vises i console.

Hvis arealet er under 100, skal der stå: **Arealet er for lille**

Er arealet mellem 100 og 200, skal der stå: **Arealet er ok**

Er det større end (eller lig med) 200, skal der stå: **Arealet er for stort**

Test programmet, så du ser alle tre muligheder for respons i funktion.

Commit og push til gitHub, når du er tilfreds med opgaven

tema 7  
undervisningsopgaver  
js-basics:  
01-areal.html  
**02-areaTest.html**



# INTERAKTIVITET



# 1. Elementer der skal bruges i de interaktive øvelser

## Knap

For at lave opgaverne skal du bruge en knap. Hvad som helst kan bruges som **knap** hvis man lægger en **click-event** på den:

```
minKnap.addEventListener("click",funktionDerSkalAktiveresNårDerKlikkes);  
  
function funktionDerSkalAktiveresNårDerKlikkes() {  
  
    console.log("Jeg har klikket på knappen");  
  
}
```



## 2. Elementer der skal bruges i de interaktive øvelser

### Input felt

Du skal også bruge et inputfelt:

```
<input type="text" id="name">
```

For at finde ud af hvad der står i dette tekstfelt kan vi bruge flg. javascript:

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



## Øvelse 3 Sig goddag

Lav en ny html-fil, **03-goddag.html**, og gem den i **js-basics**

Skriv et program, som spørger brugeren om hans/hendes navn (inputFelt), og ved et tryk på en knap herefter svarer med “Goddag Martin” (console.log), hvor navnet selvfølgelig er det angivne.

Commit og push til gitHub, når du er tilfreds med opgaven

tema 7  
undervisningsopgaver  
js-basics:  
01-areal.html  
02-areaTest.html  
**03-goddag.html**



# Øvelse 4 udregn areal i interaktiv dialog med bruger

html-filen til denne øvelse, skal hedde **04-arealInteraktiv.html**

Lav en ny udgave af areal-programmet.

Programmet skal bede brugeren angive en længde og en bredde ved hjælp af inputfelter.

Ved klik på en knap skal programmet udregne arealet, og give brugeren resultatet efter formen:

**Længde:** 5 meter, **bredde:** 7 meter, **areal:** 35 meter( længde\*bredde)

```
tema7
undervisningsopgaver
js-basics:
  01-areal.html
  02-areaTest.html
  03-goddag.html
  04-arealInteraktiv.html
```





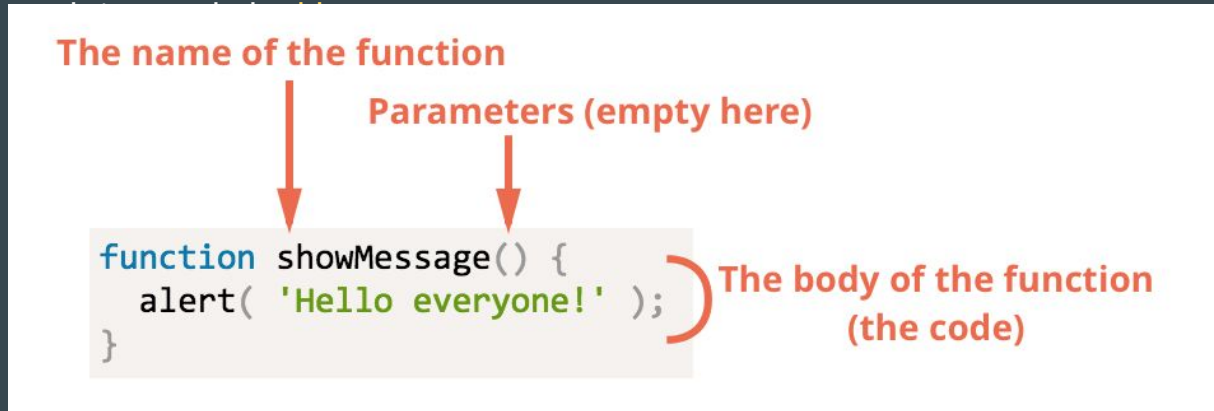
# FUNKTIONER



# Funktioner - hvorfor

Vi kikker lige på funktioner igen, vi har jo kikket på dem højere oppe i forbindels med eventlistenere.

En funktion er et stykke isoleret kode.



Når en funktion er erklæret, sker der ingenting.

Først når den kaldes, udføres funktionen:

`showMessage();`

Når vi henviser til en funktion i en eventlistener

`addEventListener("click",showMessage);`

kalder vi den ikke, der er ikke nogen `()` bagefter, for hvis der var det ville funktionen blive kaldt når eventListeneren bliver indlæst og ikke når vi klikker på knappen



# Funktioner - hvorfor?

- Man kan undgå at gentage kode ( DRY - Don't Repeat Yourself)
- Programmer bliver modulopbygget og mere overskueligt
- Funktioner kan også tit genbruges i andre programmer

Derfor forsøger man altid at oprette funktioner, hvor man kan.



# Funktion med parameteroverførsel

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

Erklæring af funktionen

Funktionen vil vise en værdi i consolen i inspectoren.

txt er det variabelnavn funktionen bruger om værdien, kaldes også et parameter

```
visBesked("Goddag");
```

Funktionen kaldes med værdien "Goddag" som parameter.

I funktionen får variabelen txt denne værdi, og den logges i consolen.

```
let besked = "Farvel";
```

```
visBesked(besked);
```

Funktionen kaldes med variabelen **besked** som parameter. **besked** har værdien "Farvel" fra linjen før.

funktionen får variabelens værdi som parameter

I funktionen får variabelen txt denne værdi, og den logges i consolen.



# Funktion med flere parametre

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

```
gangTal( 3 ,4 );
```

Erklæring af funktionen

tal1 og tal2 er de to parametre. Resultat indeholder de to parametre ganget med hinanden.

Funktionen vil vise værdien af resultat i consolen i inspectoren.

Funktionen kaldes med værdierne 3 og 4 adskilt af et komma. Der kunne sagtens være flere værdier, de skal bare være adskilt af komma.

I funktionen får variabelen *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}`;  
}
```

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

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

I console.log-sætningen kaldes funktionen med variabelen besked som parameter.  
besked har værdien “Kamilla underviser i morgen”  
i funktionen kaldes værdien for message

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

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



# Opsamling

funktion

parameteroverførsel

funktion, som returnerer en værdi



# Øvelse 5 Arealer og funktioner

Filens navn: **05-arealFunktion.html**

Lav en funktion, **areal**, som med længde og bredde som parametre, kan udregne et areal, og returnere resultatet.

Programmet skal:

- Bede brugeren om længde og bredde med input elementer

Commit og push til gitHub, når du er tilfreds med opgaven

tema7

undervisningsopgaver

js-basics:

01-areal.html

02-areaTest.html

03-goddag.html

04-arealInteraktiv.html

05-arealFunktion.html





# Øvelse 6 Arealudregning med fejlmeddelelse

Filens navn: **06-arealFejl.html**

I det forgående program, arealFunktioner.html:

Test, hvad der sker, hvis brugeren angiver tekst i stedet for tal som længde eller bredde

Prøv at ændre på programmet, så det tager højde for fejldata og giver brugeren en fejlmelding

Commit og push til gitHub, når du er tilfreds med opgaven

Husk at man kan teste om noget er et tal med `isNaN()`. NaN betyder: Not a Number

```
tema7
  undervisningsopgaver
    js-basics
      01-areal.html
      02-areaTest.html
      03-goddag.html
      04-arealInteraktiv.html
      05-arealFunktion.html
      06-arealFejl.html
```



# EFTERMIDDAGSOPGAVER



# Øvelse 7 Sig godmorgen

html-filen til denne øvelse, skal hedde `03-godmorgen.html`

Skriv et program, som i consolen siger:

Godmorgen mellem kl. 5 og kl 10,

Goddag mellem kl 10 og 18,

Godaften mellem 18 og 24 og

Godnat mellem 24 og 5.

**obs:** denne javascript-funktion fortæller, hvilken time, man befinder sig i:

`new Date().getHours()`

Commit og push til gitHub, når du er tilfreds med opgaven



## Øvelse 8 - simpel CO2 beregning

html-filen til denne øvelse, skal hedde **08-co2\_beregning.html**

Du skal lave et program, som regner ud hvor meget CO2 du bruger på en flyrejse:

$$\text{co2UdledtIKgVedFlyrejse} = \text{timer} \times 109$$

Du skal ikke i dialog med brugeren i denne opgave - læg timer ind som en variabel.

Resultatet skal vises i console-vinduet efter denne form:

**Ved 3 timers flyrejse har du udledt 327 kg CO2**

Commit og push til gitHub, når du er tilfreds med opgaven



## Øvelse 9 - simpel CO2 beregning #2

html-filen til denne øvelse, skal hedde

**09-type\_af\_klimasynder**

Programmet skal udregne udledning af CO2 ved flyrejse på baggrund af flyrejsens længde i timer

(tag udgangspunkt i en kopi af øvelse 2).

Ud fra dette skema, skal du i consolen udskrive både timer og hvor meget CO2 der er udledt.

Fx: **Du har fløjet 3 timer og udledt 327 kg CO2.**

**Det er godt for turismen i nordeuropa.**

Commit og push til gitHub, når du er tilfreds med opgaven

(galgenhumoristisk) respons	CO2 i kg
Tæt på Thunberg	mindre end 100
Ja ja, du behøver ikke at melde dig ud af Å!	Mellem 100 og 200
Det er godt for turismen i nordeuropa.	Mellem 200 og 600
Nå ja vi kan jo altid kolonisere en anden planet ...	Over 600



# Øvelse 10 Er du en klimasynder

html-filen til denne øvelse, skal hedde `10-type_af_klimasynder_interaktiv.html`

Lav en ny udgave af CO2-programmet.

Programmet skal bede brugeren angive længden på sin flyrejse i timer.

Ved klik på en knap skal programmet så udregne CO2-udledningen og give respons i consolen

Hvis brugeren laver et forkert input, skal programmet give en fejlmelding.

Commit og push til gitHub, når du er tilfreds med opgaven



# Øvelse 11 - Gæt et tal

html-filen til denne øvelse, skal hedde `11_guess.html`

Programmet skal finde et tilfældigt tal mellem 0 og 20, og bede brugeren om at gætte tallet det.

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



# Øvelse 12 - Date-rådgivning

html-filen til denne øvelse, skal hedde **12\_date.html**

Når man skal date, bør man finde en, som hverken er for ung eller for gammel.

Der findes en regel, som hedder “half your age plus seven”.

Lav et program, som kan tage imod din egen alder og din dates alder, og fortælle dig, om reglen er overholdt - både til den ene og den anden side.

Sørg endelig for, at programmet ikke kan benyttes af mindreårige eller pædofile - begge skal være over 15!

Commit og push til gitHub, når du er tilfreds med opgaven





# Øvelse 13 - Den nye hundeadler

html-filen til denne øvelse skal hedde **13\_hvor\_gammel\_er\_vuffi.html**

I gamle dage plejede man at sige at en hunds menneskealder er 7 gange hundens alder.  
Altså Freddie er nu 7.5 år gammel: 7 gange 7.5 = 52.5 år

Men det viser sig nu at hunde ældes mere når de er unge.

Den nye regnemodel ser sådan her ud: Hvis du f. eks. har en labrador, der er 12 år gammel, så er formelen:  $16 \times (12)\ln + 31 = 71$  år gammel.

Det der  $(12)\ln$  udregner den naturlige logaritme. I javascript ville det se sådan her ud: `Math.log(12)`. I behøver ikke at forstå hvad den naturlige logaritme er, I skal bare bruge den.

Lav et program der regner en hunds menneskealder ud efter den gamle og den nye metode!

Commit og push til gitHub, når du er tilfreds med opgaven