# DOM SCRIPTING med JAVASCRIPT

# Form validering

Når forms bliver udfyldt på nettet, har vi brug for at checke om de er udfyldte korrekte.
Denne kontrol hedder **validering**.
Er indtastninger 'valide' - opfylder de kravene?

# HTML5 validering:

**HTML5** giver muliged for at browseren selv foretager lidt grundlæggende validering, men det varierer mellem browsere - og browserversioner - hvor meget de kan.

Der er både nye **input types** (*email, url, number* mv.) og nye **attributes** (*required, min, max* mv.).
Browser support varierer meget, men der kan være andre fordele for mobiler og tablets.

```
<input type="text" name="navn" required>
```
Browseren kontrollerer om feltet er udfyldt.

```
<input type="email" name="email" required>
```
Kontrollerer om feltet opfylder helt basale krav. "a@b" vil fx ofte blive accepteret, selv om der mangler ".com" eller lign.
Hvis formen har 2 email felter, hvor man skal gentage adressen, bliver de ikke sammenlignet.

*Det gode er, at smartphones og tablets vil justere skærmens tastatur til at vise "@".*

**CSS** kan også style form elementer med **pseudo-classes** som ***:required, :valid, og :invalid***

```
input:focus:invalid{ background-color:#F00; }
```
Dvs at feltet med cursoren får rød baggrund hvis den ikke opfylder kravene.

```
input:required:valid{ background-color:#0F0; }
```
Dvs at nødvendige felter som er udfyldt korrekt får grøn baggrund.

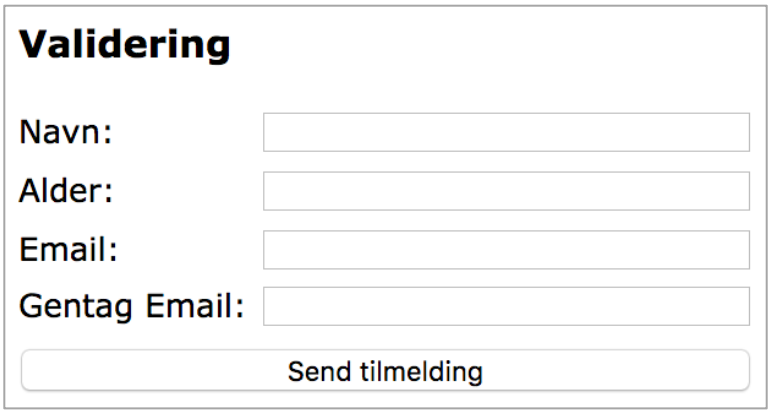
Men fordi browsernes validering er både forskelligt og mangelfuldt, skal vi validere med JavaScript.

I de følgende øvelser bruger vi ***ikke*** HTML5 validering, så man kan se effekten af vores JS kode.

# JavaScript validering:

## Øvelse:

Først skal vi have en formular
- her med 4 felter og en Send knap.




Download validering_start.zip og pak den ud.
Her er html filen med formen, en 'tak for tilmelding' svar.html, og en css fil.

Man kan enten skrive javascript enten i en ny .js fil (med link fra bunden af html filen)
eller direkte ind i html filen – i så fald skal <script> afsnittet også ligge til sidst i filen.

Bemærk at formen indeholder en tom div med ID "besked".
Her skal JavaScript fortælle brugeren hvad de mangler at udfylde.

Formen starter med:

```
<form id="form1" action="svar.html" onsubmit="return valider(this)">
```

(Normalt ville **`action`** henvise til en fil på serveren (`tilmeld.php` eller lignende), men vi vil her simulere serveren med en lille lokal svar-fil.)

Knappen skal 'returnere' (send tilbage) svaret fra en *funktion* som vil validere formen.

Der kunne være flere forms på siden. Derfor sender vi også argumentet 'this' (denne form).
*Funktionen* skal have en parameter som identificerer hvilken form den skal validere. Fx:
**`valider(f)`**.

**1. Tilføj funktionen:**

I første omgang tester vi kun for manglende udfyldelse af navnet.
Men vi skal bruge "besked" mange gange, så vi kan starte med at lægge den ind i en variabel.

```
22      var besked=document.querySelector("#besked");
23      function valider(f){
24          if(f.navn.value ==""){
25              besked.innerHTML="Angiv navn"
26              f.navn.focus();
27              return false;
28          }
29          return true;
30      }
```

**Linje 23:** Funktionen har 1 parameter - vi har valgt at kalde den 'f' - som modtager formen ('this').
**Linje 24:** Hvis formens ('f') … 'navn' feltets … *tekst* ('value') … er tom…
**Linje 25:** …skriv besked…
**Linje 26:** …sæt feltet som det aktive felt i formen…
**Linje 27:** …og send 'false' tilbage, så formularen ikke bliver afsendt.

**Linje 29:** 'return false' vil afbryde funktionen, så 'return true' sendes kun hvis 'if' betingelsen *ikke* er opfyldt.

**2. Næste check - tilsvarende for 'alder':**

```
27              return false;
28          }
29          if(f.alder.value ==""){
30              besked.innerHTML="Angiv alder";
31              f.alder.focus();
32              return false;
33          }
34          return true;
35      }
```

**3. Men vi kan også teste for om feltet 'alder' er udfyldt med tal:**

```
33          }
34          if(isNaN(f.alder.value)){
35              besked.innerHTML="Angiv alder med tal";
36              f.alder.focus();
37              return false;
38          }
39          return true;
```

**Linje 34:** `NaN` betyder I flere sprog: 'Not A Number'.
Javascript funktionen `isNaN()` giver 'true' hvis udtrykket i paranteser *ikke* er et tal.

**4. Vi kan også teste for hvilken alder - ifm. juridiske bindende aftaler mv.**

```
39          if(f.alder.value <18){
40              besked.innerHTML="Du skal være mindst 18";
41              f.alder.focus();
42              return false;
43          }
```

**5. ..og om begge email felterne er ens:**

```
64          if(f.email2.value != f.email1.value){
65              besked.innerHTML="Email-adresserne skal være ens";
66              f.email2.focus();
67              return false;
68          }
```

**6. Test for gyldig email format:**

```
var atpos = f.email1.value.indexOf("@");
var dotpos = f.email1.value.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || f.email1.value.length <= dotpos+2){
      besked.innerHTML="Skriv en gyldig e-mail adresse";
      f.email1.focus();
      return false;
}
```

`||` betyder 'or' (='eller')

`indexOf` giver tegnets position, og er 'zero-based' (d.v.s. at det første tegn er i position 0).

Dvs:
Find positionen af @ og gem den i variablen 'atpos'.
Find positionen af det sidste punktum og gem den i variablen 'dotpos'.
Check at :
- @ ikke er det første tegn (ikke er i position 0)
- det sidste punktum skal være efter @, med mindst ét tegn imellem
- det sidste punktum skal være mindst 2 tegn fra slutningen (fx .dk eller .com)

(<= betyder 'mindre end eller lig med'.
`indexOf` er zero-based, men `length` tæller fra 1.
Derfor skal `length` være *mere end* `dotpos+2` , hvis der skal være mindst 2 tegn til sidst.)