

JavaScript-øvelser

Lidt forklaring, demo og øvelser

Indhold

Lidt forklaring, demo og øvelser	1
Indhold.....	1
Opstart af nyt website til øvelserne/opgaverne.....	1
Fakta om JavaScript	2
Forskel på HTML 4.01 og HTML5.....	3
Placering af JavaScript i html-dokumentet.....	3
---- Øvelse 1 ---- Placering af JavaScript – test det selv eller gå videre til Øvelse 2	5
Variabler	6
Typesvagt.....	7
---- Øvelse 2 ---- Demo af "typesvagt" JavaScript	8
---- Øvelse 3 ---- Variabler	8
Funktioner	9
Forskel på en funktion i C# og JavaScript	11
---- Øvelse 4 ---- Funktioner	12
Funktioner med parameter	13
---- Øvelse 5 – parameter 'this' og event onclick og style-objektet	14
---- Øvelse 6 ---- parameter 'this' og onfocus/onblur og style-objektet.....	15
value og innerHTML.....	15
---- Øvelse 7 ---- onclick og innerHTML-property	16
Image - src	16
---- Øvelse 8 ---- image src	17
Forstå rækkefølgen.....	17
---- Øvelse 9 ----.....	17

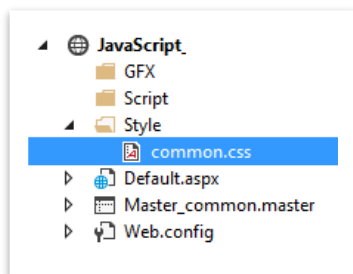
Opstart af nyt website til øvelserne/opgaverne

- **Opret et nyt .NET website – kald det evt. JavaScriptUndervisning eller noget – HER SKAL DU SAMLE ALLE ØVELSERNE I JAVASCRIPT, JQUERY mv. Det skal afleveres en af de sidste dage.**
- **Brug gerne en MasterPage** til at lave en menu med link til alle dine opgaver/eksempler ... så du nemmere kan finde rundt i postyret

- Prefix gerne dine JavaScript opgaver med **JS_** og jQuery med **JQ_** og Angular med **AN_** ... og tilføj et nummer (01, 02, 03 osv.) - så lægger din opgaver og øvelser sig pænt samlet i mappen

De første opgaver herunder handler om JavaScript – og dine løsninger skal gemmes i mappen "**JavaScript**".

Mappestrukturen kunne se sådan her ud til at begynde med:



Fakta om JavaScript

- JavaScript blev brugt første gang i **1995**. Det blev udviklet for bl.a. at kunne **validere input fra brugere** – hos brugeren/klienten (i stedet for på serveren), på et tidspunkt, hvor det var en meget langsommelig proces at klare valideringen frem og tilbage mellem klient og server, fordi det hele foregik via et 28.8 kbps telefonmodem!
- JavaScript er på en måde både et **scriptingsprog** og et **programmeringssprog**. Det er tekst i et tekstdokument (med file-extension .js) og bliver i dag kompileret af browseren hos klienten – altså på brugerens pc (frem for den server, hvor websitet er hostet). Tidligere blev JS ikke kompileret – men blot fortolket/oversat af browseren - og var derfor alene et scriptingsprog ... tjek evt. artiklen her: [A Guide to JavaScript Engines for Idiots](#)
- JavaScript hed oprindeligt "**Livescript**", men blev omdøbt til JavaScript, da programmeringssproget Java var nyt og meget populært ... dermed red Livescript med på Java-bølgen. "Vi er også en slags Java" ☺ ... hvilket ikke passer - JavaScript har stort set kun semikolon og tuborgklammer til fælles med Java.
- **JavaScript** er IKKE det samme som **Java** – Java er et rigtigt programmeringssprog, som altid har skulle kompileres (oversættes til noget andet – fx html, css mv.) af en server, hvor websitet er hostet, **INDEN** det sendes ud til brugeren/klienten. JavaScript sendes uændret direkte til klienten, hvor klientens browseren så står for at kompilere/udføre JavaScriptets instruktioner.
- **Stort set alle browsere** har en indbygget JavaScript-motor, så JavaScriptet kan blive udført. Men der kan være en del forskel på, hvordan de fungerer, og hvor hurtige de er:

JavaScript Performance

(Smaller is better)

Browser, Headless Browser, or Runtime	JavaScript Engine
Mozilla	Spidermonkey
Chrome	V8
Safari**	JavaScriptCore*
IE and Edge	Chakra
PhantomJS	JavaScriptCore
HTMLUnit	Rhino
TrifleJS	V8
Node.js***	V8
io.js***	V8

Browser	Seconds
Chrome 10.0.648.205	2.801
Chrome 13.0.754.0 canary	0.543
Firefox 4.0.1	0.956
IE 9.0.8112.16421 64	1.159
IE 10.0.1000.16394	0.562
Opera 11.10	1.106
Safari 5.0.5 (7533.21.10)	0.984

Klik på billederne for at se siderne, hvor de er taget fra. Og Google selv – der kan være sket en del, siden artiklerne blevet skrevet!

- **JScript** var Microsofts udgave af JavaScript i Netscape-tiden – JScript er således lavet primært til Internet Explorer.
- **ECMAScript** er den standard, som senere blev aftalt for alle "JavaScript-dialekterne" – og som bl.a. det nuværende JavaScript er baseret på og skal overholde.
- Tjek fx udviklingen af ECMAScript versioner, som viser noget om, hvor hurtigt det går: [ECMAScript Versions ...](#)
- JavaScript var tæt på at forsvinde, men levede en diskret tilværelse i de forskellige browsere i en årrække. JavaScripts mulighed for brug af AJAX – muligheden for at lave dynamiske kald uden reload af siden – og jQuery, pustede liv i JavaScript
- I de seneste år er brugen af JavaScript nærmest eksploderet – og der er blevet udviklet mange JavaScript libraries og frameworks mv. – se fx en oversigt her: [List of JavaScript Libraries](#)
- De forskellige JavaScript-engines – fx browseres - må gerne tilføje egen funktionalitet, så længe de overholder den grundlæggende ECMA-standard, men dermed opstår der altså funktioner, som er forskellig i de enkelte browsere.
- **Actionscript** (til Flash) er også et ECMA-scriptsprog – det samme er det sprog, som du kan scripte til Photoshop og øvrige Adobe-produkter (de kan ovenikøbet gemmes som .js-filer)
- JavaScript "består" af 3 elementer:
 - **ECMAScript** – som er grundsubstansen i alle JavaScript-varianter (fx også til Photoshop og Flash)
 - **DOM** ([Document Object Model](#)) – gælder kun browsere
 - **BOM** ([Browser Object Model](#)) – gælder kun browsere
- DOM- og BOM-delen – som er vigtige dele af JavaScript til brug for en browser - er ikke en del af ECMA-kravene.

Forskel brug af JavaScript i HTML 4.01 og HTML5

"type"-attributten SKAL med i HTML 4, men er frivillig i HTML5 (default er "text/JavaScript"):

```
<script type="text/JavaScript">
  // Noget kode ....
</script>
```

I nyere browsere burde man også kunne bruge `type="application/JavaScript"` – men ikke alle browsere understøtter denne mere korrekte udlægning. Så hold dig til `type="text/JavaScript"`

Placering af JavaScript i html-dokumentet

JavaScript kan placeres i både `<head>` og `<body>`. Det ligner til forveksling det, du kender fra CSS – hvor du har 3 muligheder for placering af stylet:

1. På selve tag'et – element-styling: `<h1 style="color: red;">Overskrift</h1>`

2. Som inline-styling i head:

```
<style>
  h1 { color: red; }
</style>
```

3. Ved at linke til et eksternt/external stylesheet).

```
<title>Øvelse</title>
<link href="Style/common.css" rel="stylesheet" />
<meta charset="utf-8" />
```

Her er JavaScriptet placeret på to forskellige måder i `<head>` (som du også kender det fra CSS):

```

<head>
  <title>Placering af JavaScript</title>

  <script type="text/javascript">
    alert("1. Script-tag i head");
  </script>

  <script type="text/javascript" src="script1.js"></script>

</head>

```

Og her er det placeret i `<body>` (hvor den ene version – den på selve tag'et – ligner det, du kender fra css):

```

<body>

  <input type="button" onclick="javascript:alert('3. Javascript direkte på et html-tag');" value="Klik her ..." />

  <script type="text/javascript">
    alert("4. Script-tag i body");
  </script>

</body>

```

1. Script-tag i head

Ok til test - eller hvis du bare har en lille stump JS, som kun skal bruges til denne side

2. Script-tag i head med link (src) til ekstern JavaScript-fil

Sådan! Så kan du genbruge filen på andre sider og holde HTML'en ren for JS

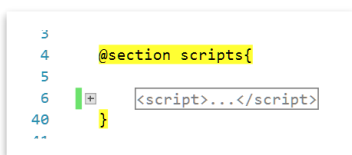
3. JavaScript direkte på et html-tag

Kun til nød eller hvis du har en særlig god grund - undgå js på elementerne ... det er noget værre rod at finde rundt i, hvis du bruger det meget og har mange sider!

4. Script-tag i body (lige før </body>)

Er blevet temmelig populært – og HTML-delen af siden afvikles en smule hurtigere, fordi scriptet ikke forsinkes indlæsning af øvrigt indhold (images mv.). Det har også den fordel, at scriptet først bliver læst, når sidens indhold er indlæst, så de tags/elementer, scriptet arbejder med ER læst ind - hvilket IKKE er tilfældet, hvis du placerer scriptet i HEAD. Omvendt kan det virke lidt rodet, at scriptet ligger i body – så man som udvikler skal scrolle ned i bunden af siden, for at finde ud af, om der er noget script på siden. OG ... så kan der være situationer, hvor scriptet SKAL placeres i head, fordi fx en eller anden funktion på en knap SKAL være der, straks knappen er synlig - hvilket ikke altid er tilfældet, hvis scriptet ligger i bunden og fx forsinkes af indlæsning af et stort image.

OBS! Jeg er fra en tid, hvor vi lærte, at script-tag'et ALTID skal placeres i head. Så det gør jeg stadig. Og om mit script så forsinket html-dokumentet i at blive indlæst handler i høj grad om, hvor meget JavaScript-kode der er. Jeg har kun små stumper kode, så derfor har det ingen væsentlig betydning, at det ligger i `<head>`. Havde jeg meget store scripts, ville jeg nok lægge det i `body`. I øvrigt er det i MVC nemt at lægge JS i bunden af siden, selvom det bliver vist i toppen af siden – ved at placere en `@RenderSection("scripts", required: false)` i bunden af `_Layout`-siden. Og så placere scripts-section i toppen af siderne med JavaScript:



Om dit script skal ligge i head eller body, må du selv afgøre – google efter fordele og ulemper (pros and cons). Uanset hvad, så hold dit script samlet ét sted, så du ikke forvirrer dig selv unødigt her i begyndelsen – eller hvis du senere skal tilbage og kigge på koderne!

Læs om fordele – ulemper ved placering af scriptet i head eller body ... fx her:

<http://stackoverflow.com/questions/2451417/whats-pros-and-cons-putting-JavaScript-in-head-and-putting-just-before-the-body>

<http://stackoverflow.com/questions/1013112/where-should-i-declare-JavaScript-files-used-in-my-page-in-head-head-or-ne>

Læs evt. mere om placering af scriptet her:

http://www.w3schools.com/js/js_where.asp

Eller den mere tekniske her: <http://www.w3.org/TR/html401/interact/scripts.html>

---- Øvelse 1 ---- Placering af JavaScript – test det selv eller gå videre til Øvelse 2

Der er som nævnt 4 steder, du kan placere dit JavaScript – som vist herover. I opgaven her handler det alene om, at du prøver at placere den samme stump kode på 4 forskellige måder.

Lav opgave 1 og læg mærke til i hvilken RÆKKEFØLGE de 4 scripts bliver kaldt frem:

Opret en JavaScript-side – kald den fx script1.js – indsæt noget kode a la det her:

```
alert("2. Script fra ekstern js-fil");
```

Opret en html- eller chtml-side og indsæt noget kode a la det her:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transi
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Placering af JavaScript</title>

  <!-- PLACERING "INLINE" I SCRIPT-TAG I HEAD -->
  <script type="text/javascript">
    alert("1. Script i head");
  </script>

  <!-- PLACERING I HEAD - LINK TIL EKSTERN JS-FIL med koden alert("2. Script fra ekstern js-fil"); -->
  <script type="text/javascript" src="script1.js"></script>

</head>
<body>

  <!-- I BODY DIREKTE PÅ ET ELEMENT/TAG -->
  <input type="button" onclick="javascript:alert('3. Script på et html-tag');" value="Klik her ..." />

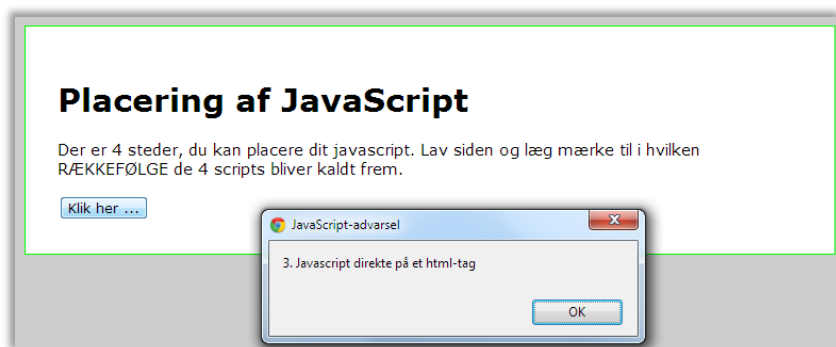
  <!-- I BODY SOM "INLINE" SCRIPT -->
  <script type="text/javascript">
    alert("4. Script i body");
  </script>

</body>
</html>

```

Læg mærke til de fire måder at placere scriptet på ...

Kør siden (åbn den i en browser) og bemærk i hvilken rækkefølge scriptet (alert'en) bliver afviklet.



Variabler

Variabler er – som i C# - en måde at opbevare værdier på. Variabler er en pegepind ind i computerens RAMs hukommelse. **Variabler består af:**

- Et passende navn – fx "dyr"
- Og kan så tilføjes/sættes lig med en værdi – fx "hest"

Tænk evt. på en variabel som en slags kasse du kan lave til at opbevare noget indhold (en værdi) i. Kassen (variablen) skal have et navn (så du kan adskille den fra andre kasser), og herefter kan du putte næsten hvad som helst i den. Senere kan du hive det (den værdi) frem, som du "har puttet i kassen", evt. lave om på værdien og putte den tilbage i kassen. Eller putte noget helt nyt i kassen. Eller fjerne/slette kassen.

Du erklærer (laver/navngiver) variabler således:

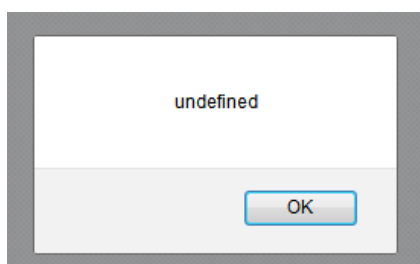
```
var etnavn;
var ettal;
var endato;
```

Eller sådan her:

```
var etnavn, ettal, endato;
```

Variablerne i eksemplet herover er "tomme kasser" – "kasserne" er gjort klar og har et navn (**etnavn**, **ettal**, **endato**). Men der er ikke puttet noget (en værdi) ned i kasserne endnu.

Hvis du laver en **alert** på indholdet af en af de tomme variabler – fx **alert(endato)**; - for at se dens indhold, vil du få svaret "**undefined**", som betyder, at variablen findes, men værdien er "udefineret" eller ukendt.



Hvis du skal putte en værdi i "kassen"/variablen, kan du gøre det sådan her – hvor variablerne først bliver erklæret/lavet og bagefter får en værdi.

```
var etnavn, ettal, endato;

etnavn = "Jørgen";
ettal = 42;
endato = new Date();
```

Værdien behøver ikke blive puttet i variablen med det samme – ofte er det først senere i scriptet, når en funktion bliver kaldt, eller hvis nogle betingelser er opfyldt, at værdien bliver puttet i variablen.

Det kan også ske senere i scriptet, at variabelens værdi bliver overskrevet af eller lavet om til en ny/anden værdi.

Du kan også erklære variablerne OG putte værdier i dem på én gang:

```
var etnavn = "Vagn";
var ettal = 42;
var endato = new Date();
```

Typesvagt

Bemærk, at JavaScript er et **type-svagt sprog**. Dvs. at du ganske vist giver "kasserne"/variablerne et navn, men du fortæller ikke noget om, hvilken type indhold man må putte i "kassen"/variablen. Fx kan det her lade sig gøre i JavaScript:

```
var etnavn = "Vagn";    // string
ettal = 42;            // number
etnavn = new Date();    // object (dato-objekt)
```

Her er variablen **etnavn** først af datatypen **string** (fordi der bliver puttet tekst/string i variablen). Herefter bliver den til et **number** (fordi string-værdien bliver overskrevet med en tal-værdi). Og til sidst bliver variablen til et **objekt** (fordi variablen bliver udfyldt med et dato-objekt).

---- Øvelse 2 ---- Demo af "typesvagt" JavaScript

Opret en *html-* eller *cshtml*-side – og test følgende kode:

```
head runat="server">
  <title>Øvelse 2</title>

  <script>

    var test = "En test";
    console.log("Nu er variabelens datatype: " + typeof(test));

    test = 42;
    console.log(("Nu er variabelens datatype: " + typeof(test)));

    test = new Date();
    console.log(("Nu er variabelens datatype: " + typeof(test)));

  </script>

head>
body>
  <form id="form1" runat="server">
    <div id="wrapper">

      <h1>Der er intet at vise på denne side!</h1>

      <p>Åbn browserens Udviklerværktøj - ofte noget med F12</p>
      <p>- og kig i konsollen/console. Du skal muligvis genindlæse (F5) siden først.</p>

    </div>
  </form>
```

C# er IKKE typesvagt – her erklærer du (eller bør i hvert fald gøre) jo variable med en type fra starten af fx:

```
int ettal = 13;
string etord = "hej";
```

Og dermed kan der KUN puttes integers (hel-tal) i variabelen `ettal`. Og tekst/string i variabelen `etord`. Alt andet vil give en fejl.

I JavaScript bliver alle variable erklæret med `var` (se eksemplerne herover), og næsten alt kan puttes i uden at du får en fejl. Og som du så i øvelsen herover, så kan den samme variabel oveni købet skifte mellem datatyper:

```
var etnavn = "Vagn";
// ... en masse kode, og så ...
etnavn = 42;
// ... igen noget kode ...
etnavn = new Date();
```

Det gør det selvfølgelig dejlig nemt, når du ikke skal spekulere over, hvilken datatype, din variabel skal rumme – men det betyder også en stor risiko for fejl senere i koden, som du først opdager "run time" (når koden afvikles). I C# får du allerede besked om fejlen, mens du udvikler ("compile time"), så du kan rette fejlen med det samme.

---- Øvelse 3 ---- Variabler

Opret en *html-* eller *cshtml*-fil og indsæt noget kode a la det her:


```

<script type="text/javascript">

    var tal1 = 10;
    var tal2 = 42;

    alert(tal1 + tal2);

    tal2 = "25";
    alert(tal1 + tal2);

    tal2 = "UPS!";
    alert(tal1 + tal2);

</script>

```

Test koden (åbn siden i browseren) – du skulle gerne få 3 **alert**-bokse med forskelligt resultat.

Læg mærke til ... svar selv på følgende (du behøver ikke skrive svaret ned):

1. Hvad sker der lige med regnestykket undervejs? Og hvad kan det forårsage af problemer senere?

Prøv evt. også at teste denne her:

```
alert('55' < '1000');
```

55 er jo mindre end 1000 – dermed burde svaret være "true" (en "sandhed").

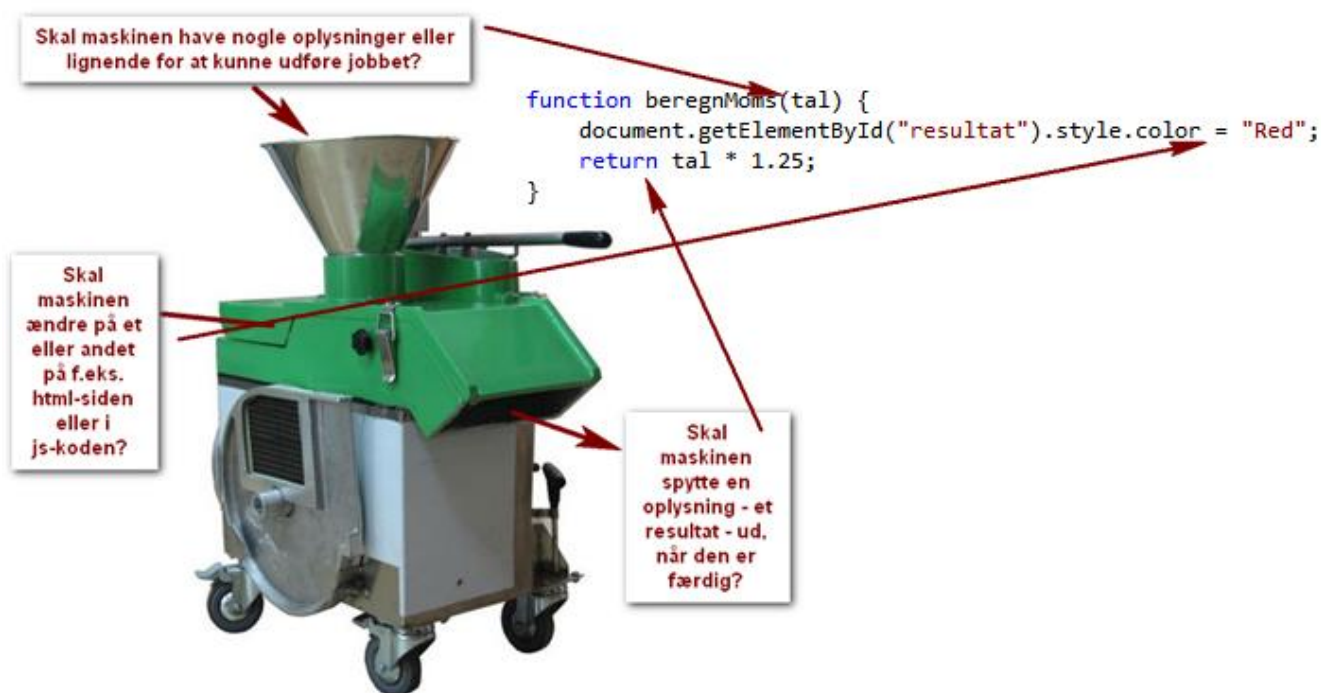
Men hvad sker der, når du afvikler koden – og hvorfor mon?

Funktioner

Funktioner kender du allerede fra C#. Funktioner i JavaScript minder meget om det, du kender derfra. Tænk fx på en funktion som en slags "kode-maskine", som udfører et eller andet stykke arbejde. Skal du oprette en funktion, skal du overveje tre ting:

1. Skal "maskinen" have et '**input**' – altså en oplysning for at kunne udføre sin opgave?
2. Skal "maskinen" spytte noget ud ('**output**'), når den er færdig?
3. Skal maskinen *ændre på noget* - fx ændre i noget tekst, eller ændre på noget styling eller andet?

Det kunne lidt primitivt illustreres sådan her:



OBS! Mindst én af de tre ting bør ske i funktionen – ellers er det en funktion uden ... funktion.

PS! Du bør også overveje, om der skal være nogle **betingelser** i din funktion. Skal maskinen/funktionen fx kun gøre noget, hvis det er december måned:

```
if (idag.getMonth() == 11)
```

Eller skal den kun gøre noget, hvis der er indtastet noget i et input-felt:

```
if (fornavn.value.length > 0)
```

Eller skal den gøre noget forskelligt afhængig af, hvad der er indtastet/valgt af brugeren? Osv.

Her er et eksempel på en simpel funktion. Du behøver ikke taste eksemplet – men kig på det og prøv i tankerne at beskrive/forstå, hvad funktionen gør:

```
function visResultat() {
  var idag = new Date();
  if (idag.getMonth() == 11) {
    document.getElementById("txt").style.backgroundColor = "#F00";
    document.getElementById("txt").innerHTML = "Så blev det endelig december!!!";
  }
}
```

1. Skal denne funktion have et input (et argument) for at virke? Hvad?
2. Returnerer – output'er – funktionen noget? Hvad?
3. Udfører funktionen et eller andet? Hvad?

En funktion bliver kaldt (sat i gang) ved at man et eller andet sted i sin kode kalder på den - altså som i ex herover skriver:

```
visResultat();
```

... så går funktionen i gang med sit job.

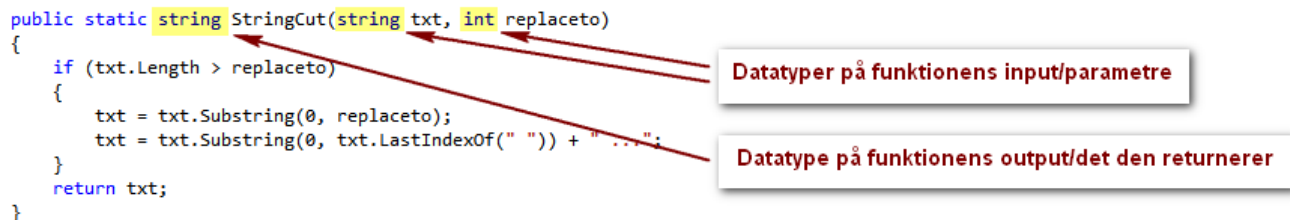
Bemærk, at der er lidt forskel på JavaScript-funktioner og dem, du kender fra **C#**. I **C#** angiver du datatyperne for parametrene ("input") – altså hvilke datatyper (tekst, tal osv.), der må puttes ind i funktionen. Og du angiver også,

hvilken datatype funktionen evt. returnerer, når den er færdig. I C# er det altså ikke muligt, at putte en forkert datatype ind i funktionen – og man kan ikke få en forkert/uventet datatype retur:

Forskel på en funktion i C# og JavaScript

Herunder er den samme funktion vist, som den vil se ud i C# og i JavaScript. Læg mærke til forskellene:

Funktionen StringCut som den vil se ud i C#

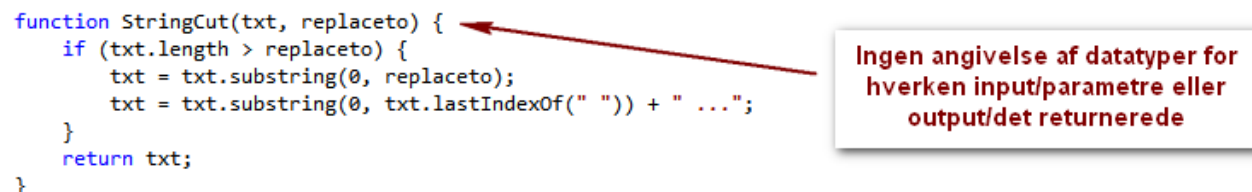


```
public static string StringCut(string txt, int replaceto)
{
    if (txt.Length > replaceto)
    {
        txt = txt.Substring(0, replaceto);
        txt = txt.Substring(0, txt.LastIndexOf(" ") + " ...";
    }
    return txt;
}
```

Datatypes på funktionens input/parametre

Datatype på funktionens output/det den returnerer

Funktionen StringCut som den vil se ud i JavaScript



```
function StringCut(txt, replaceto) {
    if (txt.length > replaceto) {
        txt = txt.substring(0, replaceto);
        txt = txt.substring(0, txt.lastIndexOf(" ") + " ...";
    }
    return txt;
}
```

Ingen angivelse af datatypes for hverken input/parametre eller output/det returnerede

I JavaScript-funktionen herover er det ikke muligt at se, om de to parametre/argumenter (txt og replaceto) skal være tekst, tal, datoer eller noget helt andet.

---- Øvelse 4 ---- Funktioner 1

Lav en nyt html- eller cshtml-fil og forsøg at løse opgaven ved at få koden herunder til at virke:

```

<script>
  //console.log("Hej Verden");

  function GoerNoget(hilsen, farve) {

    document.getElementById("overskrift").style.color = farve;
    document.getElementById("overskrift").innerHTML = "<i>Der er klikket på knappen</i>";

    document.querySelector("p").style.color = farve;
    document.querySelector("p").innerText = "Dette er en hilsen: " + hilsen;
  }

</script>

</head>
<body>

  <div id="wrapper">

    <h1 id="overskrift">Her er en test</h1>
    <p>Consectetur adipiscing elit fusce vel.</p>

    <input type="button" onclick="GoerNoget('Hej med dig', 'Blue');" value="Klik her" />
    <br />
    <input type="button" onclick="GoerNoget('Halløjsa', 'Green');" value="Klik her" />

  </div>

```

1. Når der bliver klikket på den ene knap
 - skifter teksten farve til **blå** – og der sendes **en hilsen med som bliver vist**
2. Når der bliver klikket på den anden knap
 - skifter teksten farve til **grøn** – og der sendes **en anden hilsen med som bliver vist**

---- Øvelse 5 ---- Funktioner 2

Lav en nyt html- eller cshtml-fil og forsøg at løse opgaven her:

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4      <title>2. Funktioner</title>
5      <link type="text/css" rel="stylesheet" href="common.css" />
6      <script type="text/javascript">
7
8          function visResultat() {
9              // Kode her!
10          }
11
12      </script>
13  </head>
14  <body>
15
16      <div id="wrapper">
17
18          <h1>En funktion</h1>
19          <p>Lørdag og søndag = rød baggrundsfarve.</p>
20          <p>Alle andre dage = grøn baggrundsfarve.</p>
21
22          <p id="txt">Hvilken baggrundsfarve får denne tekst mon?</p>
23
24          <input type="button" value="Klik her for at finde ud af det" />
25
26      </div>
27
28  </body>
29  </html>

```

Du skal sørge for, at funktionen "**visResultat**" udfører følgende job:

3. Hvis det er **lørdag** eller **søndag**, skal teksten "*Hvilken baggrundsfarve ...*" have en **rød** baggrundsfarve
4. Hvis det er **en anden dag end lørdag og søndag**, skal baggrundsfarven være **grøn**
5. Og det hele skal ske, når man klikker på knappen (= den event som sætter funktionen i gang)

HJÆLP!!!

- Kig på eksemplet med den "simple funktion" tidligere i dokumentet her!
- Læs mere om if-sætninger: http://w3schools.com/js/js_if_else.asp - du får jo brug for at spørge 'if det er søndag eller lørdag så ...'
- Mere om "sammenligning" (større end, mindre end, eller, og osv.): http://w3schools.com/js/js_comparisons.asp
- Mere om events (klik på knap og den slags): http://w3schools.com/js/js_html_dom_events.asp

Udfordring: Hvis det var nemt, og du blev hurtig færdig, så arbejd videre med en kopi af din opgaveløsning. Udvid den evt. så hver dag giver en forskellig farve ... eller du skriver noget om, hvilken ugedag det er osv.

Funktioner med parameter

En funktion kan – som tidligere nævnt – modtage en eller flere oplysninger, som funktionen så bruger til et eller andet.

I det følgende eksempel modtager funktionen en **parameter**/et argument, som er en oplysning om det element (**this**), som kalder på funktionen.

En parameter er som en variabel og skal have et navn ... herunder får parameteren navnet **denne**, så når scriptet skriver **denne**, er der i virkeligheden tale om **this** – altså det element, som kaldte funktionen.

```
<script type="text/javascript">

    function visResultat(denne) {
        denne.value = "Tak fordi du klikkede ...";
    }

</script>
<head>
<body>

<div id="wrapper">

    <h1>En funktion - som modtager en parameter</h1>
    <p>Denne funktion modtager en parameter (input), som bruges i funktionen.</p>

    <input type="button" onclick="visResultat(this)" value="Klik her ..." />
    <br />
    <input type="button" onclick="visResultat(this)" value="Klik her ..." />

</div>
```

Bemærk, at man – når man arbejder med input-felter – både kan hente og ændre input-feltets værdi ... det gøres med **.value**

---- Øvelse 6 – parameter 'this' og event onclick og style-objektet

Opret en fil og test eksemplet herover.

Test at det virker, inden du går videre. Udvid og eksperimenter meget gerne, når du har fået det til at virke.

Udfordring: Hvis det var nemt, og du blev hurtig færdig, så arbejd videre med en kopi af din opgaveløsning. Sørg f.eks. for, at når der klikkes på den ene knap, så går teksten på den anden knap tilbage til "Klik her ...".

Her er et andet eksempel:

```

<script type="text/javascript">

    function goerNoget1(denne) {
        denne.style.backgroundColor = "#F0D8DB";
        denne.value = "";
    }

    function goerNoget2(denne) {
        denne.style.backgroundColor = "#FFF";
    }

</script>
</head>
</body>

<div id="wrapper">

    <h1>En funktion - som modtager en parameter</h1>
    <p>Denne funktion modtager en parameter, som bruges i funktionen.</p>

    <input type="text" onfocus="goerNoget1(this)" onblur="goerNoget2(this)" value="Skriv dit navn" />

</div>

```

Herover har elementet – et inputfelt – to forskellige events/hændelser:

- Når der fokus (**onfocus**) på elementet – kaldes funktionen **goerNoget1**
- Når der ikke længere er fokus (**onblur**) – kaldes funktionen **goerNoget2**

---- Øvelse 7 ---- parameter 'this' og onfocus/onblur og style-objektet

Opret en fil og test eksemplet herover.

Test at det virker, inden du går videre. Udvid og eksperimenter meget gerne, når du har fået det til at virke.

Udfordring: Hvis det var nemt, og du blev hurtig færdig, så arbejd videre med en kopi af din opgaveløsning og forsøg dig evt. med denne: Det er muligt helt at undgå script i **<body>** ... så der f.eks. ikke er en **onfocus** og **onblur** på **<input>** ... men hvordan? Tænk over det og eksperimenter lidt. Det er ikke helt nemt, men prøv alligevel – det giver en god viden om, hvordan JavaScript "tænker" – også selvom du ikke får løst udfordringen.

value og innerHTML

I den foregående opgave så du, at man kan ændre på attributten **value** – som er en attribut - i et input-felt – f.eks. sådan her:

```
document.getElementById("input1").value = "Skriv dit navn i det her felt";
```

Man kan også ændre på selve "html-koden" med **innerHTML** - f.eks. teksten i et **<p>** :

```
document.getElementById("etptag").innerHTML = "Det her er en ny tekst";
```

Herover er det et element med en **id="etptag"**, som får en ny tekst. Man kan også sende html-kode med i **innerHTML**:

```
document.getElementById("etptag").innerHTML = "<b>Det her</b> er en ny <i>tekst</i>";
```

---- Øvelse 8 ---- onclick og innerHTML-property

Lav en ny html- eller cshtml-fil og tast noget html-kode a la det her:

```
<script type="text/javascript">

    // Noget kode her!!!

</script>
<head>
<body>

<div id="wrapper">

    <h1>En funktion</h1>

    Skriv et tal: <input type="text" value="" id="inp1" />
    <br />
    Skriv et tal: <input type="text" value="" id="inp2" />
    <br /><br />
    <input type="button" onclick="visResultat()" value="Klik her for at lægge de to tal sammen" />

    <h4 id="resultat"></h4>
```

Du skal nu sørge for, at funktionen "visResultat" udfører følgende job:

1. De to tal, der indtastes, skal plusses
2. Resultatet af regnestykket skal vises i <h4>
3. Og det hele skal ske, når man klikker på knappen

Du opdager nok undervejs, at der **opstår lidt problemer med datatyperne** – få hjælp herunder!

HJÆLP!!

- Du vil sikkert opdage, at du får brug for at vide det her: http://www.w3schools.com/jsref/jsref_parseint.asp
- Mere om events (klik på knap og den slags): http://w3schools.com/js/js_htmldom_events.asp
- Mere om operatorer ... plus, minus og den slags: http://www.w3schools.com/js/js_operators.asp

Udfordring: Hvis det var nemt, og du blev hurtig færdig, så arbejd videre med en kopi af din opgaveløsning. Udvid evt. så brugeren både kan plusse og minusse – og måske gange og dividere. Måske kunne du også, at det ikke er muligt at indtaste andet end tal?

Image - src

Herover har du set, at man kan ændre på inputfelters værdi med value, og på elementers indhold – f.eks. teksten i et <p> eller indholdet i en <div> - med innerHTML. På samme måde kan du ændre på source – src – på et billede sådan her:

```
document.getElementById("img1").src = "img/blomst.jpg";
```

Så næste opgave handler selvfølgelig om, at du med JavaScript skal ændre på et billede – eller skifte et billede ud - på html-siden.

---- Øvelse 9 ---- image src

Lav en ny html- eller cshtml-fil og tast noget html-kode a la det her:

```
<script type="text/javascript">

    // Måske en funktion, der starter her?
    // Noget kode inden i funktionen?
    // Og så slutter funktionen et eller andet sted ;-)

</script>
<head>
<body>

<div id="wrapper">

    <br /><br />

</div>
```

... og fyld så selv ud med det, der mangler. Der skal ske det at ...

1. Når man fører musen henover de små billeder, skal billedet i #imgcontainer erstattes med en stor version af det lille billede, som musen er over.

HJÆLP!!

- Når et billede i et skal udskiftes, så er det jo i virkeligheden src="???", der ændres i.
- Du skal bruge en event ("**når musen er over billedet**"), som skal sætte det hele i gang – se forskellige muligheder her: http://w3schools.com/jsref/dom_obj_event.asp
- Og hvis du vil se nogle eksempler på, hvordan man med JavaScript kan udskifte af billedet (src'en) i et : http://www.w3schools.com/jsref/prop_img_src.asp

Udfordring: Hvis det var nemt, og du blev hurtig færdig, så arbejd videre med en kopi af din opgaveløsning. Brug din fantasi – f.eks. pile, der kan flytte rundt med billeder – og find selv på et eller andet awesome!

Style/css – og forstå rækkefølgen

JavaScript læses og tolkes af klienten/browseren OPPEFRA OG NED. Helt bogstaveligt. Det er vigtigt at huske, da det har betydning for den kode, du laver. Vi tager lige et eksempel:

---- Øvelse 10 ----

Lav en ny html- eller cshtml-fil og tast noget html-kode a la det her:

```

<head>
  <title></title>

  <script>

    document.getElementById("overskrift").style.color = "Lime";

  </script>

</head>
<body>

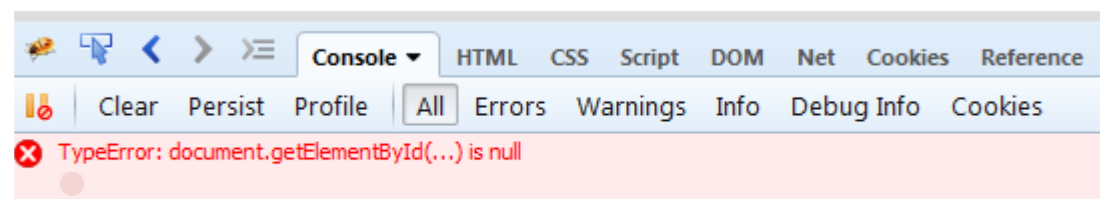
  <h1 id="overskrift">Her er en overskrift</h1>

</body>
</html>

```

Koden skulle gerne ændre stylen på det element, som har ID'en "overskrift" ... så overskriften får color "Lime".

Kør siden i browseren ... og tjek den fejl, du får i konsollen - hvad går galt?



Det der går galt er, at når linjen med koden `document.getElementById("overskrift").style.color` bliver tolket af browseren, så er resten af siden endnu ikke indlæst. Så der findes ikke et element med ID'en "overskrift" ... derfor får du en fejl.

Så vi er nødt til at sikre, at vi først forsøger at ændre skriftfarven til Lime, når overskriften er indlæst på siden.

Prøve med denne her ændring:

```

<head>
  <title></title>

  <script>

    window.onload = function () {
      document.getElementById("overskrift").style.color = "Lime";
    };

  </script>

</head>
<body>

  <h1 id="overskrift">Her er en overskrift</h1>

</body>
</html>

```

Nu forsøger vi først at skifte tekstfarven, når siden er loadet – dvs. når h1-overskriften med ID "overskrift" er indlæst. Så virker det. Test det selv.

Alternativt kunne man sikre, at overskriften først skifter farve, når der bliver klikket på et eller andet – på den måde er vi sikre på, at siden er loadet/indlæst:

```

<script>

  function skiftFarve() {
    document.getElementById("overskrift").style.color = "Lime";
  };

</script>

head>
ody>

  <h1 id="overskrift">Her er en overskrift</h1>
  <input type="button" id="knap" value="Klik her" onclick="skiftFarve();" />

```

Afslutningsvis

Bemærk, at du både kan bruge `getElementById` og `querySelector` til at fange et html-tag ud fra dets Id:

```
document.getElementById("overskrift").innerHTML = "En ny overskrift";
```

... gør det samme som denne her:

```
document.querySelector("#overskrift").innerHTML = "En ny overskrift";
```

Du kan også oprette flere styles/css-regler med