

# Laboration 1

## Variabler och beräkningar

### – övningar/uppgifter

1M322 Webbteknik 2, 7,5hp

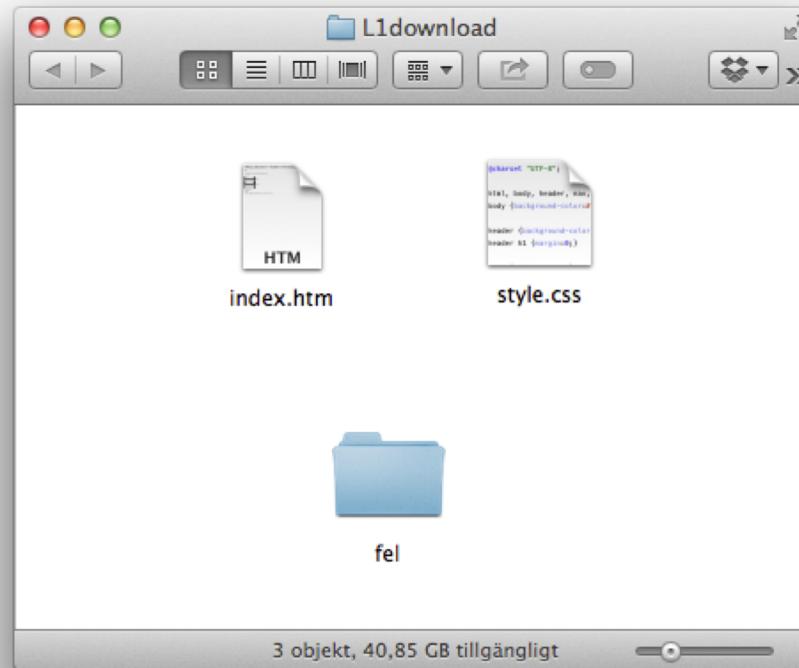
Medieteknik



# 1. Ladda ner arbetsdokument

Till övningarna i denna laboration finns det ett antal filer som du kan ladda ner i en zip-fil.  
Länk till zip-filen finns på laborationens webbsida.

Då du packat upp zip-filen, får du en mapp med två filer och ytterligare en mapp med ett antal filer.



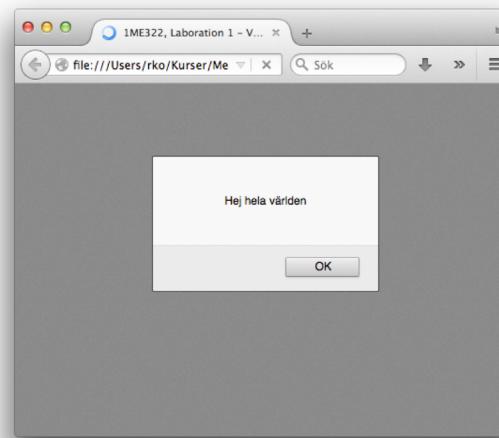
## 2. Länk till scriptfil

Du ska nu skapa en fil för JavaScript-koden och länka in den i HTML-filen.

- Öppna din webbeditor och skapa en ny fil för JavaScript.
  - Lägg in kommandot `alert("Hej hela världen");` i filen.
    - Det är det enda som just nu ska ligga i filen, för att testa den.
  - Spara filen med namnet `script.js` i samma mapp som du har HTML-filen som du laddade ner.
- 
- Öppna filen `index.htm` i webbeditorn.
  - I `head`-delen lägger du in ett `script`-element som refererar till filen `script.js`.
- 
- Öppna filen `index.htm` i din webbläsare.
    - Då filen läses in ska du få en ruta med meddelandet som du skrev i `alert`.



```
alert("Hej hela världen");
```



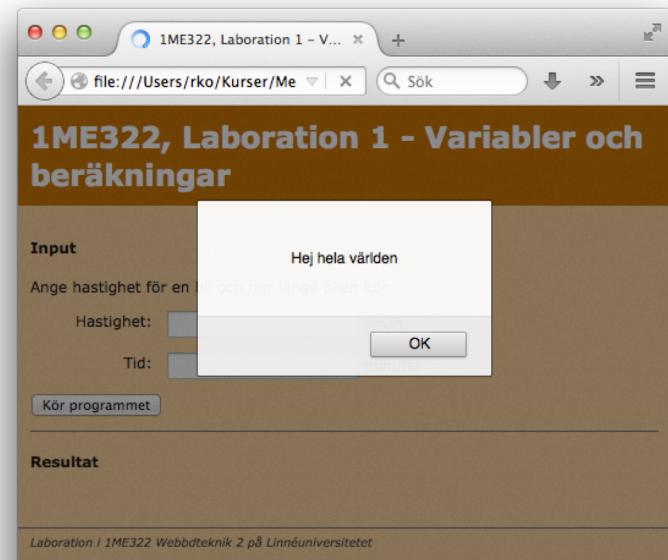
### 3a. En init-funktion

Nu ska du i scriptfilen lägga in en funktion som anropas, då webbläsaren läst in hela webbsidan.

Du kommer skapa en kod som liknar den som finns i exempel *js1-ex3-2* i föreläsning 1, fast du gör det här i två steg.

- I filen *script.js* skriver du en funktion som du kallar *init*.
- Flytta *alert*-satsen med din hälsning till funktionen, så att den ligger mellan klamrarna för funktionen.
- Efter funktionen skriver du *window.onload = init;*
  - Detta gör att *init* utförs, då webbsidan är inladdad i fönstret.
- Testa i webbläsaren.
  - Om du fortfarande har *index.htm* öppen, klickar du på knappen för att ladda om. Annars får du öppna sidan igen.
  - Meddelandet från *alert* ska nu komma, då webbsidan är inläst.

```
function init() {  
    alert("Hej hela världen");  
} // End init  
window.onload = init;
```



### 3b. En testfunktion

Nu ska du skapa ytterligare en funktion och koppla den till knappen på webbsidan.

- I *script.js* lägger du till en funktion som du kallar *testScript*.
- Flytta *alert*-satsen från *init* till *testScript*.

Du refererar här till *id*-attribut i HTML-koden. Så titta även i filen *index.htm*, och se vilka element det refereras till.

- I *init*-funktionen skriver du in de fyra rader som finns i rutan här intill, dvs samma som i *init*-funktionen i exempel *js1-ex3-2*.
- De tre variablerna deklarerar du ovanför funktionen.

- De måste ligga där, för att du sedan även ska kunna använda dem i den andra funktionen.

```
var input1Elem, input2Elem, resultElem;

function init() {
    input1Elem = document.getElementById("input1");
    input2Elem = document.getElementById("input2");
    resultElem = document.getElementById("result");
    document.getElementById("runBtn").onclick = testScript;
} // End init
window.onload = init;

function testScript() {
    alert("Hej hela världen");
} // End testScript
```

- Testa i webbläsaren.
  - Ladda om sidan, så att det nya scriptet läses in.
  - Klicka på knappen, så ska meddelandet med *alert* dyka upp.

I exempel *js1-ex3-2* ser du också att det finns kommentarer ovanför variablerna och funktionerna, som förklarar vad de är till för. Man bör alltid ta med förklarande kommentarer i sin kod, så lägg in sådana kommentarer i din kod också.

## 4. Läs input och skriv på sidan

Nu ska du skriva in kod i funktionen *testScript*, för att avläsa textfälten och skriva i *div*-elementet på sidan.

- I funktionen *testScript* tar du nu bort *alert*-satsen och lägger istället in det följande.
- Deklarera två variabler, *speed* och *time*.
- Läs in det första textfältet och spara i variabeln *speed*.  
Det andra textfältet sparar du i variabeln *time*.
- Det ska vara tal som ska skrivas i textfälten,  
så konvertera till den datatypen med *Number*.
  - Vi gör dock nu ingen kontroll av att användaren  
verkligen skriver tal. Det kommer i laboration 2.
- Skriv ut variabeln *speed* på webbsidan,  
för att testa att det fungerar att avläsa fältet och skriva på sidan.
- Testa i webbläsaren.
  - Skriv något tal i det första textfältet och klicka på knappen.  
Det du skrev i textfältet ska då skrivas på sidan under rubriken Resultat.

```
function testScript() {  
    var speed; // Hastighet i km/h  
    var time; // Tid i minuter  
  
    speed = Number(input1Elem.value);  
    time = Number(input2Elem.value);  
  
    resultElem.innerHTML= speed;  
} // End testScript
```

## 5. Uttryck

Nu ska du beräkna den sträcka man kommer med den hastighet och tid som användaren skriver i textfälten.

- Deklarera variabeln *distance* i början av funktionen.

```
var distance; // Sträcka
```

- Efter raderna där du läste in *speed* och *time* lägger du in ett uttryck för att beräkna sträckan. Resultatet sparas i variabeln *distance*.

- Sträckan är hastighet gånger tid, men eftersom hastigheten är i km/h och tiden i minuter, måste du också dividera med 60, så att enheten för tiden också blir timmar.

```
// Sträcka för angiven tid  
distance = speed * time / 60;  
resultElem.innerHTML = "Sträckan blir " + distance + " km. <br><br>";
```

- Ändra utskriften till att skriva ut *distance* tillsammans med någon lämplig text.
  - Lägg till två *br*-taggar i slutet av utskriften. Du ska i kommande övningar göra flera utskrifter och då ska de hamna på en rad under denna.

- Testa i webbläsaren

- Skriv in några värden i textfälten och klicka på knappen.

**Input**

Ange hastighet för en bil och hur länge bilen kör.

Hastighet:  km/h

Tid:  minuter

**Kör programmet**

**Resultat**

Sträckan blir 67.5 km.

# 6. Fler uttryck

Nu ska du beräkna hur lång tid det tar att köra samma sträcka med en lägre hastighet.

- Sist i funktionen *testScript* lägger du till kod för att beräkna tiden.
  - Tiden blir sträckan delat med hastigheten. Det blir då i timmar, så du får multiplicera med 60, för att få det i minuter.
- Skriv ut värdet i *time* tillsammans med lämplig text.
- Använd nu operatorn `+=`.
  - Då lägger du till den nya texten, utan att ta bort den gamla.

```
// Tid, om man kör 20 km/h längsammare
time = distance / (speed - 20) * 60;
resultElem.innerHTML += "Tiden för samma sträcka, om hastigheten är 20 km/h lägre blir " + time + " minuter. <br><br>";
```

- Testa i webbläsaren.
  - Värdet för tiden blev nu ett flyttal med många decimaler, men vi struntar i det nu. Det går avrunda till heltalet med *parseInt*, men vi tar det i nästa laboration.

**Input**

Ange hastighet för en bil och hur länge bilen kör.

Hastighet:  km/h

Tid:  minuter

---

**Resultat**

Sträckan blir 67.5 km.

Tiden för samma sträcka, om hastigheten är 20 km/h lägre blir 57.85714285714286 minuter.

# 7. Beräkning av reaktionssträcka

Nu ska du beräkna hur långt man åker från det man upptäcker en fara tills man börjar bromsa, dvs reaktionssträckan. Detta ska beräknas i hur många meter man åker med den hastighet som angetts i inputfältet.

- I funktionen *testScript* lägger du till två variabler, *reactionTime* och *speedMS*.

```
var reactionTime; // Reaktionstid i sekunder  
var speedMS; // Hastighet i m/s
```

- Variabeln *reactionTime* ska innehålla hur lång tid det tar innan man reagerar. Antag att det tar tre sekunder, så lägg in värdet 3 i den variabeln.
- Variabeln *speedMS* ska vara hastigheten i m/s.
  - För att omvandla från km/h multiplicerar du med 1000 och dividerar med 3600.
- Sträckan beräknas, liksom tidigare, med formeln hastighet gånger tid.

```
// Reaktionssträcka  
reactionTime = 3;  
speedMS = speed * 1000 / 3600;  
distance = speedMS * reactionTime;  
resultElem.innerHTML += "Om reaktionstiden är " + reactionTime + " sekunder blir reaktionssträckan " + distance + " m. <br><br>";
```

- Testa i webbläsaren.
  - Prova att ändra värdet för *reactionTime* i programmet och ladda om sidan.

**Input**  
Ange hastighet för en bil och hur länge bilen kör.

Hastighet:  km/h  
Tid:  minuter

**Resultat**  
Sträckan blir 67.5 km.  
Tiden för samma sträcka, om hastigheten är 20 km/h lägre blir 57.85714285714286 minuter.  
Om reaktionstiden är 3 sekunder blir reaktionssträckan 75 m.

# 8. Accelerationssträcka

Nu ska du beräkna accelerationssträcka för tre olika bilar. Namn och tid för dessa läggs in i två "arrayer".

- Deklarera variablerna *car* och *accTime* i början av funktionen *testScript*.
  - Dessa ska vara arrayer, så lägg in värden i dem direkt.

```
var car = ["Volvo", "BMW", "Ferrari"];      // Bilmärken
var accTime = [10.5, 7, 4.3];                // Accelerationstid 0-100
```

- Tiden anges i sekunder och sträckan ska anges i meter, så hastigheten 100 km/h får först räknas om till enheten m/s, genom att multiplicera med 1000 och dividera med 3600.
- Accelerationssträckan beräknas som hastigheten gånger accelerationen delat med 2.
  - Vi antar att det är en jämn acceleration hela tiden. Genom division med 2 får vi alltså ett medelvärde mellan sträckan för den lägsta hastigheten (0) och den högsta hastigheten.
  - Detta beräknas för de tre bilarna och tiden erhålls ur arrayen *accTime*.

```
// Accelerationssträcka 0-100 km/h för olika bilar. 100 km/h räknas först om till m/s.
speedMS = 100 * 1000 / 3600;
distance = speedMS * accTime[0] / 2;
resultElem.innerHTML += car[0] + " 0-100 på " + accTime[0] + " sek. på " + distance + " meter. <br>";
distance = speedMS * accTime[1] / 2;
resultElem.innerHTML += car[1] + " 0-100 på " + accTime[1] + " sek. på " + distance + " meter. <br>";
distance = speedMS * accTime[2] / 2;
resultElem.innerHTML += car[2] + " 0-100 på " + accTime[2] + " sek. på " + distance + " meter. <br><br>";
```

- Testa i webbläsaren.

Volvo 0-100 på 10.5 sek. på 145.83333333333334 meter.  
BMW 0-100 på 7 sek. på 97.22222222222223 meter.  
Ferrari 0-100 på 4.3 sek. på 59.72222222222222 meter.

# 9. Egna tillägg

Nu har du gått igenom ett antal övningar, där du fått koden given. Du har alltså sett i bilderna hur du ska skriva koden. Nu kommer ett par uppgifter, där du inte får koden. Det beskrivs vad du ska göra, men det är upp till dig att avgöra hur det ska göras. Du skriver alltså kod för beräkningarna och skriver ut resultat på webbsidan. Vid behov inför du också nya variabler.

## 9a. Beräkning av sträcka

- Beräkna den sträcka som man kör under 20 sekunder om hastigheten är 40 km/h högre än den som angetts i textfältet. Sträckan ska anges i meter, så hastigheten får omvandlas till m/s.
- Skriv sedan ut sträckan tillsammans med lämplig text under de övriga utskrifterna i programmet.

## 9b. Beräkning av acceleration

- Inför en ny variabel för acceleration.
- Beräkna accelerationen i  $\text{m/s}^2$  för en BMW för 0-100 km/h och med den tid som anges i arrayen. Formeln för acceleration är hastighet dividerat med tid.
- Skriv sedan ut accelerationen tillsammans med lämplig text under de övriga utskrifterna i programmet.

**Input**  
Ange hastighet för en bil och hur länge bilen kör.

Hastighet:  km/h  
Tid:  minuter

**Kör programmet**

---

**Resultat**  
Sträckan blir 67.5 km.  
Tiden för samma sträcka, om hastigheten är 20 km/h lägre blir 57.85714285714286 minuter.  
Om reaktionstiden är 3 sekunder blir reaktionssträckan 75 m.  
Volvo 0-100 på 10.5 sek. på 145.83333333333334 meter.  
BMW 0-100 på 7 sek. på 97.22222222222223 meter.  
Ferrari 0-100 på 4.3 sek. på 59.72222222222222 meter.

Sträckan, om hastigheten är 40 km/h högre, blir 722.222222222223 m. på 20 sekunder

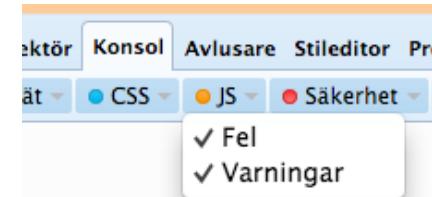
Accelerationen för en BMW är 3.9682539682539684  $\text{m/s}^2$ .

# 10a. Finn fem fel

Att hitta fel (dvs "debugging") är en viktig del av programmering.  
Här ska du öva på att hitta några fel i ett program.



- Öppna filen *index.htm* i mappen *fel* i webbläsaren Firefox.
- Öppna webbkonsolen.
  - Välj kommandot **Webbutvecklare->Webb konsol** i menyn **Verktyg** i Firefox.
  - Webbkonsonlen öppnas då i den undre halvan av fönstret.
  - Ta fram menyn för JS och se till att både **Fel** och **Varningar** är markerat.
- Öppna också filen *script.js* i mappen *fel* i din webbeditor, så att du kan rätta felet där.
- Det första felmeddelande som debuggern ger är på rad 29.  
Felmeddelandet är att det saknas semikolon efter width.
  - Felet är dock att det saknas en operator för multiplikation.
  - Rätta felet, spara filen och ladda om sidan i webbläsaren.
- Det andra felet är på rad 33. Felmeddelandet säger att det förväntades ett uttryck, men det kom <. I koden står det *area +* och då förväntar sig webbläsaren att uttrycket ska fortsätta.
  - Felet här är att det första citationstecknet kring "<br><br>" saknas.
  - Rätta felet, spara filen och ladda om sidan i webbläsaren.
  - När du nu laddar om sidan kommer inget felmeddelande i debuggern.



*fortsättning på  
 nästa sida ...*

# 10b. Finn fem fel

*... fortsättning från föregående sida*

- Skriv in några värden i textfälten och klicka sedan på knappen. Då får du ett nytt felmeddelande i debuggern. Felet är på rad 27 och felmeddelandet säger att *number* inte är definierat.
  - Felet är att det är felskrivet. Det ska vara *Number* med stort *N*.
  - Rätta felet, spara filen och ladda om sidan i webbläsaren.
- Kör programmet igen, så får du nästa felmeddelande på rad 32. Felmeddelandet säger att *heigth* inte är definierat.
  - Här är det också en felstavning, där *h* och *t* är omkastat. Det ska vara *height*.
  - Rätta felet, spara filen och ladda om sidan i webbläsaren.
- Kör programmet igen, genom att klicka på knappen.  
Nu får du inga fler felmeddelanden och utskriften kommer ut.

*fortsättning på nästa sida ...*

# 10c. Finn fem fel

*... fortsättning från föregående sida*

- Men rubriken på övningen är ju "Finn **fem** fel" och hittills har det endast varit fyra fel. Så vad är det femte felet?
- Arean för ovalen blir alltid 3, oavsett vilka värden som skrivs in i webbläsaren.
  - Felet är alltså i uträkningen av arean på rad 35. Där står det  $area = 3,14\dots$  etc.
  - Arean sätts till 3. Sedan kommer ett kommatecken och ett nytt uttryck. Detta är inte språkligt fel, utan man kan räkna ut flera uttryck åtskilda av komma.
    - Man kan t.ex. skriva `a=5, b=7;` för att ge två variabler värden samtidigt.
  - Men felet här är att det inte ska vara komma, utan en punkt, för att få korrekt värde för pi.
    - Flyttal i JavaScript skrivs med decimalpunkt och inte decimalkomma.
    - Rätta felet, spara filen och ladda om sidan i webbläsaren.
- Kör programmet igen och kontrollera att det nu blir rätt resultat.

## 10d. Finn fler fel

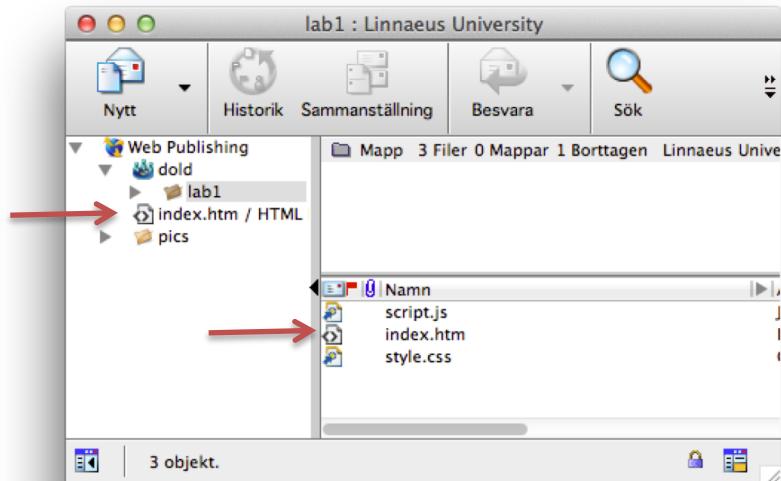
Experimentera nu med programmet genom att själv införa fel i det. Se vilka fel debuggern kan upptäcka och vad felmeddelandena blir.

- Inför egna fel i programkoden och ladda om sidan i webbläsaren.
- Kontrollera om debuggern ger felmeddelande och vad det säger.
- Rätta felet, ladda om och fortsätt tills du inte får några fler felmeddelanden.

# 11. Publicera ditt program

Då du är klar med ditt program publicerar du det i *Web publishing* i FirstClass.

- Lägg upp ditt program i en mapp kallad *lab1* i mappen *dold* i *Web Publishing* i FirstClass.
  - Dvs mappen *dold*, som du skapade i kurset 1ME321.
- Skapa sedan en länk till filen *index.htm* i *lab1* från filen *index.htm* i *Web publishing*, dvs ingångssidan för din portfolio.
  - Portfolion, som du skapade i kurset 1ME321.
- Öppna din portfolio i webbläsaren och testa att länken och ditt program fungerar.



Mappen (konferensen) *dold* skapade du och ställde in behörighet på i lab 1 i kurset Webbteknik 1 (<http://medieteknik.lnu.se/1me321/uppg-lab/l1.htm>). Ingångssidan till portfolion (*index.htm* i *Web Publishing*) skapade du i lab 2 i kurset Webbteknik 1.

Om du inte längre har kvar detta eller gått en annan motsvarande kurs, så att du inte har dessa delar, måste du nu skapa det. Du behöver då inte göra hela lab 1 och 2 i Webbteknik 1, utan det räcker att skapa *dold* och ställa in behörighet samt skapa en enkel ingångssida, där du har ditt namn och länkar till det du nu skapar i Webbteknik 2.