Laboration 5
Dynamisk HTML och timers
– övningar/uppgifter

1M322 Webbteknik 2, 7,5hp

Medieteknik

Linnéuniversitetet
Kalmar Växjö

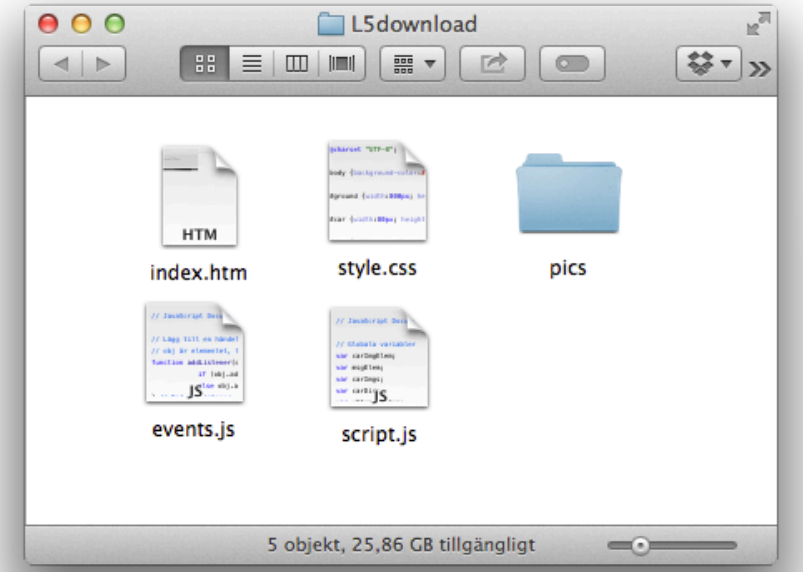
rune.kornefors@lnu.se

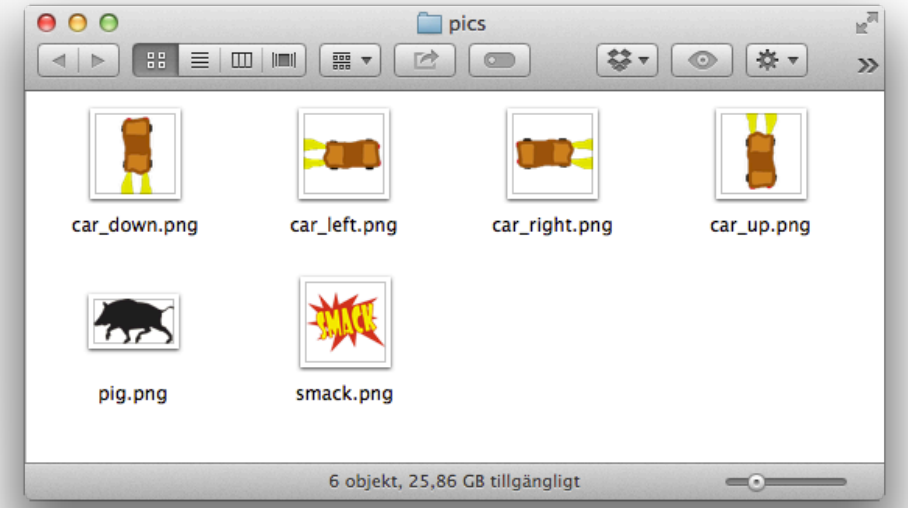
1. Ladda ner arbetsdokument

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.

Mappens filer är i stort sett desamma som det sista exemplet i föreläsning 5, dvs *js5-ex6-1b*, fast det finns ytterligare två bildfiler och några små tillägg i HTML- och CSS-koden.





2. Orientera dig i det nedladdade programmet

Om du inte redan gått igenom exemplet i föreläsningen, börjar du med att göra det. Orientera dig i filerna i den nedladdade mappen. I filen script.js finns det kommentarer med markeringar där du ska lägga till kod i övningarna.

Studera koden i HTML-, CSS- och JS-filerna,
så att du förstår hur programmet fungerar.

Det som finns inlagt nu är att man kan styra en bil
och köra runt med den inom en yta.

Tillägg i övningarna

De tillägg som ska göras i övningarna i denna laboration är följande:

- Det ska dyka upp vildsvin med slumpmässig placering. Totalt kommer det fram 10 vildsvin, men endast ett i taget.
- I verkligheten bör man ju undvika att krocka med vildsvin, men i detta spel ska du jaga dem med bilen och köra på dem. Det behövs då en funktion för att kontrollera om bilens bild och vildsvinets bild överlappar varandra.
- Det behövs också en timer, så att inte alla vildsvin kommer på en gång, utan de ska dyka upp med jämna mellanrum. När en viss tid gått ska också det vildsvin som visas tas bort (oavsett om man kört på det eller ej) och ett nytt vildsvin dyker upp på en ny plats.
- Det behövs också två räknare – en som räknar hur många vildsvin som dykt upp och en som räknar hur många vildsvin som användaren träffat.

3a. Ett nytt vildsvin

Du ska nu lägga in kod för att kunna visa ett nytt vildsvin. I HTML-koden finns det en *img*-tagg med id *"pig"* för vildsvinet. I CSS-koden har den *img*-taggen en absolut positionering. Den är också dold med *visibility:hidden*.

Det som ska göras i funktionen för en ny gris är att slumpmässigt bestämma värden för *top* och *left* och sedan ändra *visibility*, så bilden visas.

Global variabel

Du behöver en global variabel med en referens till *img*-taggen för vildsvinet.

- Inför den globala variabeln *pigImgElem*.
- I *init*-funktionen tar du fram en referens till *img*-taggen med id *"pig"* och sparar i variabeln *pigImgElem*.

Funktionen *newPig*

Skapa en ny funktion kallad *newPig* längst ner i js-filen.

- Inför två variabler kallade *t* och *l*.
- I dessa variabler bestämmer du slumpmässigt värden för placering av vildsvinet.
  - Se kod i vidstående figur.
  - Höjden på den omgivande boxen är 500px. Bilden för grisen är 40px. Så största värde för *top* blir 460px, om grisen ska finnas inom boxen. Men för att inte placera den precis i kanten tar vi en marginal på 10px, dvs i slumptalet läggs det till 10 och övre gränsen minskas med 20. Då blir det ett slumptal mellan 10 och 450. På motsvarande sätt bestäms värdet för *1*.

```
t = Math.floor(440*Math.random())+10;
l = Math.floor(740*Math.random())+10;
```

- Lägg in *t* och *l* som nya värden för *top* och *left* för grisens bild (som du har en referens till i *pigImgElem*).
  - Glöm inte bort att lägga till enheten *"px"* då du lägger in *t* och *l* i *top* och *left*.
- Ändra bildens *visibility* till *"visible"*.

Anrop av funktionen

Lägg in ett anrop av funktionen *newPig* i slutet av funktionen *startGame*.

Testa

- Öppna filen *index.htm* i webbläsaren och klicka på knappen för att starta spelet. Det ska då dyka upp ett vildsvin någonstans på spelytan.
- Mer händer inte just nu, utan det behövs sedan lite mer kod i funktionen.

3b. Ett nytt vildsvin

Vildsvinen ska inte ligga framme hela tiden, utan det ska sedan komma nya vildsvin med jämna mellanrum, men endast ett i taget. Det behövs då en timer, för att bestämma när nya vildsvin ska dyka upp.

Globala variabler

- Inför en global variabel kallad *pigTimerRef* och en annan kallad *pigDuration*.
- I *init*-funktionen sätter du *pigTimerRef* till *null*, för att markera att ingen timer ännu startats.
  - Det behövs då du senare ska kontrollera variabeln, för att stoppa timern.
- I *init*-funktionen sätter du också *pigDuration* till *2000*.
  - Denna ska senare användas i timern, för att ange att det ska vara 2 sekunder mellan vildsvinen.

Funktionen *startGame*

- Ändra anropet av *newPig*, till att sätta en timer som anropar funktionen. Se kod i vidstående bild.
  - Då visas inte den första grisen med en gång, utan först efter 2 sekunder.

```
pigTimerRef = setTimeout(newPig,pigDuration);
```

Funktionen *newPig*

- Även i slutet av funktionen *newPig*, lägger du in ovanstående rad, så att det kommer en ny gris efter 2 sekunder.
  - Dvs, det som då händer är att *img*-taggen med grisen får en ny placering.

Funktionen *stopGame*

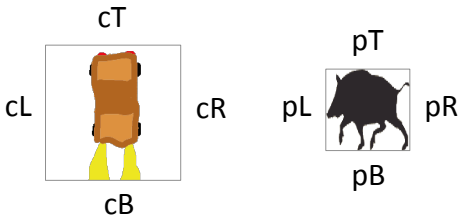
- Lägg in en rad, så att timern för grisarna (*pigTimerRef*) stoppas.
- Lägg också in en rad där *visibility* för bilden med grisen ändras till *"hidden"*.

Testa

- Ladda om sidan i webbläsaren och starta spelet.
- Det ska nu dyka upp nya vildsvin med 2 sekunders mellanrum.
- Klicka på knappen "Stoppa spelet". Då ska bilen stanna och det ska inte dyka upp några nya vildsvin.

4a. Kontrollera om bilen krockar med vildsvinet

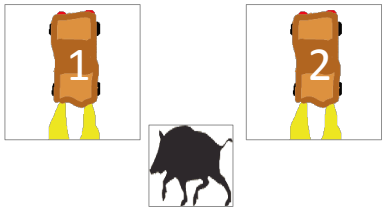
Nu ska du skriva kod som kontrollerar om bilen och vildsvinet krockar med varandra. I programmet inträffar detta, om bilderna för bilen och vildsvinet överlappar varandra. Vi betecknar kanterna på bilen med cT, cL, cR och cB och motsvarande för grisen.



Vi börjar med att analysera situationen att bilen kör nedåt.

Det är nog lättare att titta på villkoren för att det inte ska vara någon träff.

Vi har då två fall:



Vi får villkoret (cR < pL || cL > pR).

Samtidigt är det heller ingen överlappning om (cB < pT || cT > pB).

Så totalt får vi villkoret (cR < pL || cL > pR || cB < pT || cT > pB).

Detta gäller även då bilen kör i en annan riktning, t.ex. åt höger:

Detta villkor beskriver alltså situationen att vi **inte** har en krock.

För att få fram villkoret för en krock, får vi invertera det och får då:

(cR >= pL && cL <= pR && cB >= pT && cT <= pB)

Bilden med bilen har ju också lite marginaler till vänster och höger om bilen. Även strålkastarljuset blir ju en marginal mellan bildens kant och bilen. Så för att kompensera för detta tar vi bort 10px från cR och cB och lägger till 10px på cL och cT. Formeln blir då:

(cR-10 >= pL && cL+10 <= pR && cB-10 >= pT && cT+10 <= pB)

De koordinater som vi har för bilderna är *top* och *left* som vi använder för att placera dem. Detta motsvarar det som här kallats cT, cL, pT och pL. För att få fram de andra kanterna, får vi lägga till bredd och höjd. Båda bilderna är kvadratiska, så bredden och höjden är densamma. Men det är olika storlek på bilen och grisen, så vi kan kalla dem cSize och pSize. Formeln blir då:

(cL+cSize-10 >= pL && cL+10 <= pL+pSize &&
cT+cSize-10 >= pT && cT+10 <= pT+pSize)

Detta är alltså villkoret som ska kontrolleras, för att se om vi har en krock.

4b. Kontrollera om bilen krockar med vildsvinet

Kontrollen av om bilen träffat vildsvinet läggs i en separat funktion. Denna anropas sedan från funktionen där bilen förflyttas. Varje gång det sker en förflyttning, får man kontrollera om det har blivit en träff.

Globala variabler

- Deklarera de båda globala variablerna *cSize* och *pSize*.
- I *init*-funktionen lägger du in 80 i *cSize* och 40 i *pSize*.
  - Det är dessa storlekar bilderna ges i CSS-koden, men vi måste nu också lägga in dem i variablerna.

Funktionen *checkHit*

- Skapa en ny funktion som du kallar *checkHit*.
- Deklarera variablerna *cT*, *cL*, *pT* och *pL*.
- Avläs sedan *top* och *left* för bilen och vildsvinet och spara i dessa variabler.
  - Det blir på samma sätt som för variablerna *x* och *y* i funktionen *moveCar*.
- Lägg sedan in en *if*-sats med villkoret som togs fram på föregående sida.
  I *if*-satsen ska du sedan skriva flera satser, så lägg in klamrar.
- I *if*-satsen lägger du in följande:
  - Stoppa timern för grisen med *clearTimeout*.
  - Byt ut bilden för grisen till *"smack.png"*.
  - Starta timern för att ta fram en ny gris efter den tid som anges i *pigDuration*.
    - Dvs samma rad med en timer som ges i övning 3b.

Funktionen *newPig*

Eftersom du vid en träff byter bild i *img*-taggen för grisen, måste du lägga in bilden på grisen igen, då en ny gris skapas.

- Lägg in en rad ovanför den där du ändrar *visibility*. Skriv där kod för att lägga in bilden *"pig.png"* i *img*-taggen.

Funktionen *moveCar*

- Sist i funktionen *moveCar* lägger du in ett anrop av *checkHit*.

Testa

- Testa i webbläsaren. Du ska nu få bilden Smack, då du träffar en gris. Sedan kommer det en ny gris efter 2 sekunder.

5a. Räknare

Du ska nu införa ett par räknare. Den ena ska räkna hur många vildsvin som tagits fram. Maximalt ska man kunna få 10 vildsvin.
Den andra räknaren ska räkna hur många gånger man kört på vildsvinen.

Globala variabler

- Deklarera de globala variablerna *pigNr* (som är nummer för aktuell gris) och *hitCounter* (som är antal träffar).
  - Dessa ska sedan initieras i funktionen *startGame*, alltså inte i *init*.
- Deklarera också *pigNrElem* och *hitCounterElem* som globala variabler.
- I *init*-funktionen tar du fram referenser till elementen med id *"pigNr"* och *"hitCounter"* och lägger in i de två sistnämnda variablerna.

Funktionen *startGame*

- Lägg in 0 både i variabeln *pigNr* och *hitCounter*.
  - Du lägger lämpligen in raderna innan du startar timern för *newPig*.

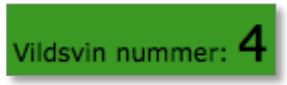
Funktionen *newPig*

- Lägg in en *if*-sats, där du kontrollerar om *pigNr* är mindre än 10. I så fall ska det skapas en ny gris, så omge de programsatser du redan har i funktionen med klamrar, så att de endast utförs om villkoret i *if*-satsen är sant.
- Lägg till en *else*-del till *if*-satsen, där du anropar funktionen *stopGame*.
  - Om det redan skapats 10 grisar, ska spelet avslutas.
- I den övriga koden i *if*-satsen lägger du också in en rad som räknar upp *pigNr* med 1 och en rad som skriver ut *pigNr* i elementet som refereras av *pigNrElem*.

Testa

Vi väntar med uppräkning av *hitCounter* och testar för *pigNr*.

- Ladda om sidan i webbläsaren och starta spelet.
- Kontrollera att räknaren längst ner på sidan visar vildsvinets nummer samt att du sedan inte får fler grisar, då räknaren kommit upp till 10.



5b. Räknare

Nu går vi vidare med att också hantera räknaren för antal träffar.

Funktionen *checkHit*

- I *if*-satsen, dvs då det blivit en träff, lägger du in en rad för att räkna upp *hitCounter* med 1.
- Lägg också in en rad där du skriver ut *hitCounter* i elementet som refereras av *hitCounterElem*.

Testa

- Testa i webbläsaren.
- Då du kör på ett vildsvin, räknas *hitCounter* upp flera gånger.
  - Det beror på att kontrollen av träff görs för varje förflyttning av bilen. Bilen rör sig över vildsvinet och det registreras då som flera träffar.

5c. Räknare

Du ska nu justera koden, så att det endast registreras en träff då ett vildsvin träffas.
För att kunna göra det, behövs en "flagga" (variabel som är *true* eller *false*) för att markera om vildsvinet redan är träffat eller ej.

Global variabel

- Deklarera en global variabel som du kallar *catchedPig*.

Funktionen *newPig*

- I *if*-satsen, där där du la in uppräkningen av *newPig*, lägger du också in en rad där du sätter *catchedPig* till *false*.
  - Vildsvinet är ännu inte träffat.

Funktionen *checkHit*

- I *if*-satsen, där du registrerat träff, lägger du in en rad där du sätter *catchedPig* till *true*.
  - Vildsvinet är nu träffat.
- I början av funktionen lägger du in en *if*-sats, där du kontrollerar *catchedPig*.
  Om variabeln är *true*, utför du satsen *return*, för att lämna funktionen.
  - Då är grisen redan träffad, och du ska inte kontrollera träff igen.

Funktionen *startGame*

- Lägg in en rad där du sätter *catchedPig* till *true*.
  - Det finns ju då inget vildsvin ännu, men bilen börjar röra sig och då anropas ju *checkHit*.
    Så variabeln markerar då att ingen kontroll ska göras.

Testa

- Testa i webbläsaren.
- Nu ska räknaren för antal träffar endast räknas upp engång per vildsvin.

6. Publicera och testa

Spelet är därmed klart och du publicerar nu, som vanligt, ditt program i *Web publishing* i FirstClass, på samma sätt som du publicerade i föregående laborationer.

Kommentarer

- Gå först igenom ditt program och kontrollera att du skrivit en förklarande kommentar för varje variabel och varje funktion.

Publicera ditt program

- Lägg upp ditt program i en mapp kallad *lab5* i mappen *dold* i *Web Publishing* i FirstClass.
- Skapa en länk från ingångssidan i din portfolio (*index.htm* i *Web Publishing*) till den första sidan i ditt program (*index.htm* i *lab5*).

Testa

- Ta fram din portfolio i webbläsaren och kontrollera att länken fungerar samt att allt i ditt program fungerar.