Laboration 4
Händelser och drag and drop
– övningar/uppgifter

1M322 Webbteknik 2, 7,5hp

Medieteknik

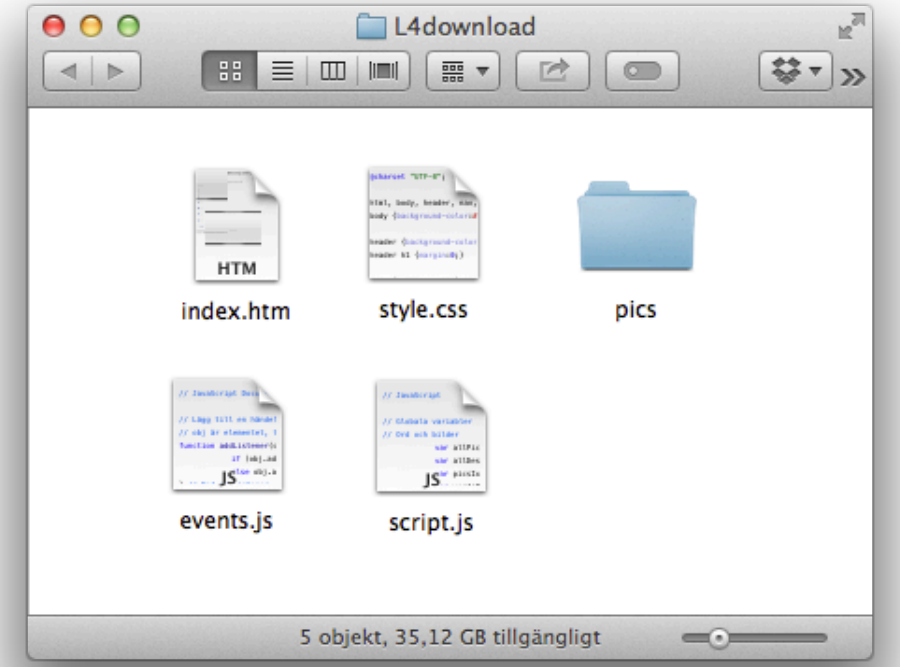
Linnéuniversitetet
Kalmar Växjö

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.

Då du packat upp zip-filen, får du en mapp med fyra filer och ytterligare en mapp med ett antal bildfiler.
Det är i filen script.js som du ska skriva koden i övningarna.



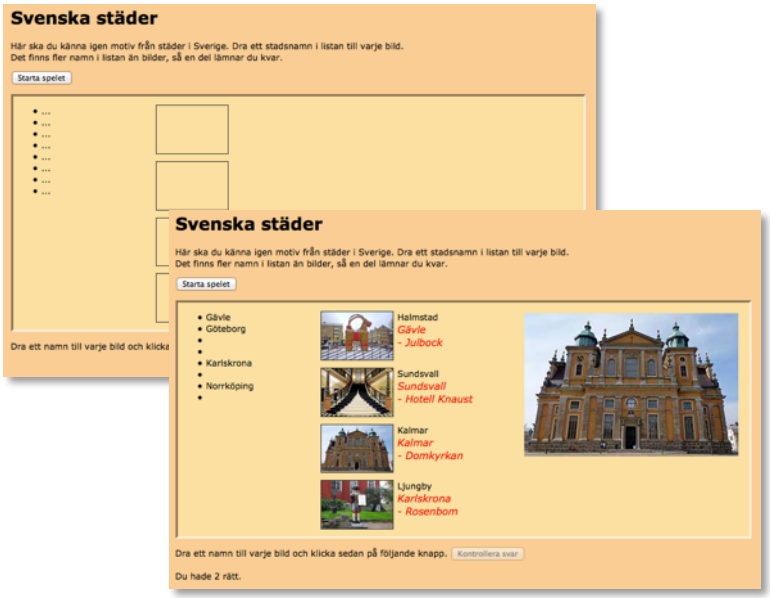
2. Gränssnitt och filerna

I denna laboration ska du skapa en applikation där det väljs ord och bilder slumpmässigt. I koden och bildfilerna är det svenska städer, men det går att byta ut orden och bilderna till vad som helst. Funktionaliteten kommer ändå bli densamma.

Det väljs åtta ord och fyra bilder, dvs åtta stadsnamn och bilder från fyra städer.
Användaren ska sedan dra namnen till rätt bilder. Därefter kan man kontrollera hur många rätt man har.

Börja med att studera gränssnittet och filerna

- Öppna *index.htm* i webbläsaren.
  - På webbsidan visas en lista med åtta punkter, där det nu endast visas ... på varje rad.
  - Det finns också fyra bildrutor, för de bilder som ska väljas.
  - Till höger om dessa finns utrymme för de ord som ska dras dit, de korrekta svaren samt ett *img*-element för en förstorad bil.
    - Dessa delar är dock tomma från början, så därför syns inget där.
  - Det finns också två knappar och ett utrymme för meddelande längst ner.
- Öppna *index.htm* i editorn.
  - Identifiera de olika delarna i gränssnittet.
    - En *ul*-lista med *li*-element för orden.
    - *img*-element för bilderna.
    - *span*-element för ord och korrekta svar intill bilderna.
    - Många av dessa element har *id*- eller *class*-attribut, för att kunna referera till dem.
- Öppna *style.css* i editorn
  - Studera CSS-koden och vilka delar i HTML-koden som det refereras till.
- Öppna filerna *events.js* och *script.js* i editorn.
  - I *events.js* finns ett antal funktioner för händelsehantering. Dessa har introducerats i föreläsning 4. Du använder dem i detta program, men ska inte göra några ändringar i den filen i laborationen.
  - I *script.js* finns det en del kod inlagd redan. Det är sådant som du redan tränat på i tidigare laborationer, så för att underlätta ditt arbete i denna laboration finns redan de globala variablerna och skal till funktionerna inlagda.
    - Studera den kod som finns och läs de förklarande kommentarerna.



Då programmet är klart, ser du ut så här.

3a. Initiera globala variabler för element i gränssnittet

I *init*-funktionen ska du nu ta fram referenser till olika delar i gränssnittet och spara i globala variabler.

Under kommentaren *"// Referenser till element i gränssnittet"* lägger du in kod för följande:

- Ta fram referenser till de båda knapparna och spara i variablerna *startGameBtn* och *checkAnswersBtn*.
- Ta fram en referens till *ul*-listan med orden och spara i variabeln *wordListElem*.
  - Tänk på att det är det första elementet i en array, så indexera med *[0]*.

```
wordListElem = document.getElementById("words").getElementsByTagName("ul")[0];
```

- Ta fram referenser till alla *li*-element i listan. Det blir en array som sparas i variabeln *wordElems*.
- Ta fram en array med referenser till *img*-elementen inom *"pics"*. Spara i variabeln *picsElems*.
- Ta fram referenser till *span*-taggarna för användarens svar och *span*-elementen för rätt svar.
  - Använd *getElementsByClassName*.
  - Detta är något nytt som du inte gjort tidigare, så den koden visas i bilden här intill.

```
userAnswerElems = document.getElementsByClassName("userAnswer");
correctAnswerElems = document.getElementsByClassName("correctAnswer");
```

- Ta fram en referens till *img*-elementet för den stora bilden och spara i *largePictElem*.
- Ta fram en referens till *div*-elementet för meddelanden och spara i *msgElem*.

3b. Händelsehanterare

Nu ska du lägga på händelsehanterare på knapparna och de små bilderna, dvs du ska koppla dem till funktioner.

Under kommentaren *"// Lägg på händelsehanterare"* skriver du kod för följande:

- Använd funktionen *addListener*, för att lägga på händelsehanterare.
  - Den finns i filen *events.js*, som länkas in med ett *script*-element i HTML-filen.
- Lägg händelsen *"click"* på *startGameBtn*, så att funktionen *startGame* anropas, då man klickar på knappen.
- Lägg händelsen *"click"* och funktionen *checkAnswers* på *checkAnswersBtn*.
- Gå igenom de små bilderna i *picsElems* i en loop och lägg på följande händelsehanterare på varje element:
  - *"mouseover"* och funktionen *showLargePict*.
  - *"mouseout"* och funktionen *hideLargePict*.

Testkod

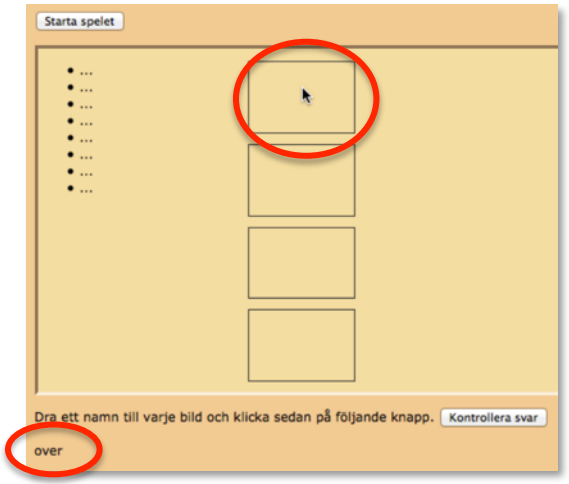
Nu ska du testa i webbläsaren, men då får du först lägga in tillfälliga meddelanden i de fyra funktionerna.

- I funktionerna *startGame*, *checkAnsers*, *showLargePict* och *hideLargePict* lägger du in en programsats, så att du skriver något i meddelandefältet, dvs det du refererar till från *msgElem*.
  - Glöm inte *innerHTML*.
  - Skriv t.ex. *"start"*, *"check"*, *"over"* respektive *"out"* i de fyra funktionerna.

Testa

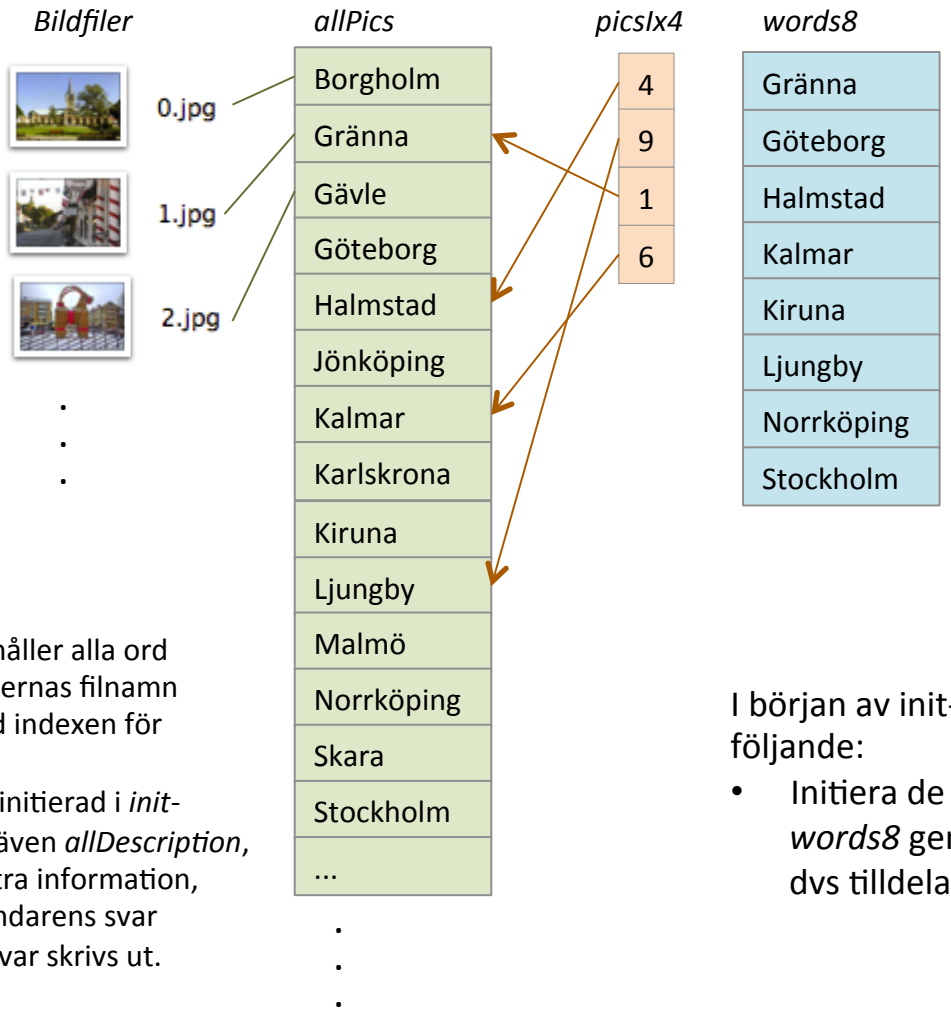
- Öppna nu *index.htm* i webbläsaren.
- Testa att klicka på knapparna. Längst ner på sidan ska du då få de meddelanden som du skrev i funktionerna.
- Testa också att föra muspekaren över en bild och sedan ut från en bild. För varje bild ska du få meddelandena i funktionerna.

Då du sett att det fungerar, tar du bort meddelandena från funktionerna igen.



4. Globala variabler för ord och bilder

Här beskrivs de globala variabler som används för val av ord och bilder.



De fyra bilder som väljs slumpmässigt, ska sparas i variabeln *picsIx4*. Det är numren i filnamnen som sparas, vilket då också blir index till *allPics*.

Orden för de fyra bilderna plus ytterligare fyra ord sparas i variabeln *words8*.
Denna array sorteras sedan i bokstavsordning, så att orden och bilderna inte kommer i samma ordning.

Variabeln *allPics* innehåller alla ord (namn på städer). Bildernas filnamn stämmer överens med indexen för städerna i *allPics*.
Variabeln finns redan initierad i *init*-funktionen. Där finns även *allDescription*, som innehåller lite extra information, som skrivs ut då användarens svar kontrolleras och rätt svar skrivs ut.

I början av init-funktionen lägger du till följande:

- Initiera de båda variablerna *picsIx4* och *words8* genom att skapa tomma arrayer, dvs tilldela dem [].

```
picsIx4 = [];
words8 = [];
```

5a. Välja ord i funktionen startGame

Du ska nu slumpmässigt välja ord (och därmed också bilder) ur variabeln *allPics*. Ett ord får inte väljas mer än en gång, så då ordet är valt, ska det tas bort ur arrayen. Detta blir alltså på samma sätt som att dra kort ur en kortlek i ett exempel i föreläsning F3. Men *allPics* behövs igen oförändrad, då svaren ska kontrolleras och då man startar spelet igen. Så du ska börja med att ta en kopia av *allPics*.

Skriv kod i funktionen *startGame*

- Deklarera tre variabler: *i*, en loopvariabel, *r* en variabel för slumptal och *tempList*, kopia av *allPics*.
- Kopiera *allPics* till *tempList* med koden i bilden här intill.

```
tempList = allPics.slice(0);
```

  - För en array räcker det inte att endast skriva *tempList = allPics*.
    Då får man endast en referens till arrayen. Om man sedan gör ändringar i *tempList*, gäller det även *allPics*.
    Så för att få en riktig kopia, får man använda *slice*.
  - Med *slice(0)* får du en ny array med alla element från och med position 0, dvs hela arrayen.

Ta fram de fyra första orden, som också blir bilderna

- Skriv en loop som genomlöps fyra gånger. I loopen gör du följande:
  - Bestäm ett slumptal som är ett index till *tempList*. Spara slumptalet i variabeln *r*.
  - Spara ordet från *tempList*, som indexeras av *r*, i *words8[i]*.
  - Spara bildens nummer (dvs index till *allPics*) i *picsIx4[i]*.
    - Eftersom du sedan ska ta bort det valda ordet ur *tempList*, blir inte index till *tempList* detsamma som index till *allPics*.
      Så du måste nu ta fram index för valt ord i *allPics*. Det gör du genom att med *indexOf* söka efter ordet i *allPics*.
      Det nummer som ska sparas i *picsIx4[i]* blir alltså resultatet från den sökningen. Se koden i bilden.

```
picsIx4[i] = allPics.indexOf(tempList[r]);
```

  - Ta bort det valda ordet ur *tempList*.
    - Observera att funktionen nu är *splice* med *p* och inte *slice*.
    - Första parametern (*r*) anger från vilken position element ska tas bort och andra parametern (*1*) anger hur många element som ska tas bort.

```
tempList.splice(r,1);
```

fortsättning på nästa sida ...

5b. Välja ord i funktionen startGame

... fortsättning från föregående sida

Testa

- Efter loopen lägger du in två *alert*-satser, där du skriver ut arrayerna, så du kan testa koden.

```
alert(words8);
alert(picsIx4);
```

- Ladda sedan om sidan i webbläsaren och klicka på knappen "Starta spelet".
  - Du ska då få ut en lista med fyra ord och sedan en lista med fyra tal.
- Då du ser att det fungerar, tar du bort *alert*-satserna igen.

Välj ytterligare fyra ord

- Lägg in en loop till, där du väljer ytterligare fyra ord ur *tempList*.
  - Loopen blir nästan likadan som föregående loop, fast du börjar nu på 4 och du ska inte lägga in något i *picsIx4*. Det är endast i *words8*, som du lägger till ord.

```
for (i=4; i<8; i++)
```

- Efter loopen lägger du in en programsats, där *words8* sorteras.

```
words8.sort();
```

Testa

- Efter loopen lägger du in en *alert*-sats, där du skriver ut *words8*, så du kan testa koden.

```
alert(words8);
```

- Ladda sedan om sidan i webbläsaren och klicka på knappen "Starta spelet".
  - Du ska då få ut en lista med åtta ord sorterade i bokstavsordning.
- Då du ser att det fungerar, tar du bort *alert*-satsen igen.

5c. Skriv ut orden och visa bilderna

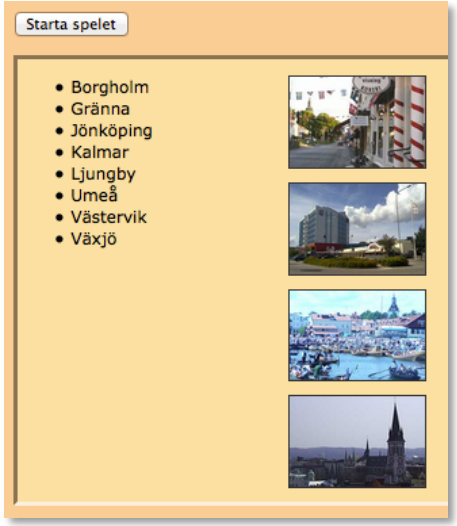
Du ska nu skriva ut de valda orden och visa bilderna på webbsidan.

I funktionen *startGame* lägger du till följande:

- En loop där du går igenom *words8* och lägger in orden i *li*-elementen, som du har i *wordElems*.
- En loop där du går igenom *picsIx4* och lägger in bilderna i *img*-taggarna, som du har i *picsElems*.
  - Tänk på att *picsIx4* endast innehåller numret i filnamnet, så du får lägga till sökvägen och filändelsen i url:en.

Testa

- Ladda om sidan i webbläsaren.
- Klicka på knappen "Starta spelet".
  - Du ska då få fram fyra ord i listan och fyra bilder i rutorna.



6. Visa/dölj den stora bilden

Nu ska du skriva kod för att visa och dölja den stora bilden, då man för muspekaren över en liten bild.
Denna kod ska ligga i funktionerna *showLargePict* och *hideLargePict*.

Du har redan i övning 3b lagt händelserna *mouseover* och *mouseout* på bilderna och kopplat det till funktionerna.
Så nu ska du endast skriva koden i funktionerna.

Kod i funktionen *showLargePict*

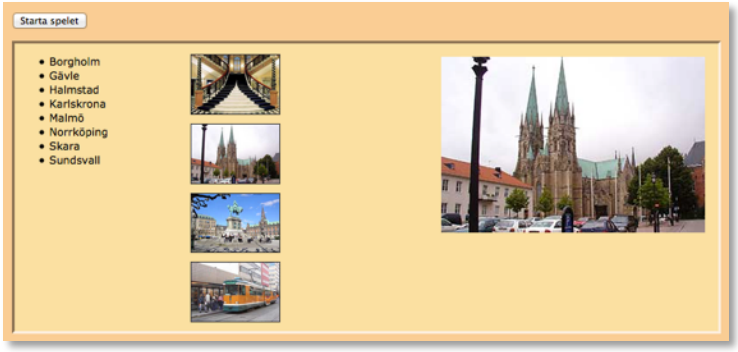
- Då du kommer in i funktionen är *this* en referens till den *img*-tagg som muspekaren är över.
- Avläs *src*-attributet och lägg in det i *src*-attributet för *largePictElem*.

Kod i funktionen *hideLargePict*

- I *src*-attributet för *largePictElem* lägger du in url:en för bilden *empty.png*, så att den bild som visades tas bort igen.

Testa

- Ladda om sidan i webbläsaren.
- Klicka på knappen "Starta spelet".
- För sedan muspekaren över de små bilderna.
  - Bilden som du pekar på ska då visas som en större bild till höger.



7a. Dra ord

Nu ska du skriva den kod som behövs för att kunna dra orden som finns i listan. För att lägga på och ta bort händelsehanterare för drag and drop ska du använda funktionen *eventsForDrag*, som är påbörjad i js-filen. Du börjar i denna övning med koden för att dra och ska i en senare övning komplettera med resterande kod för drop.

I funktionen *eventsForDrag* lägger du in följande

```
if (drag) {

}
else {

}
```

- Skriv först in en *if*-sats, där du kontrollerar parametern *drag*.
  - Parametern ska vid anrop av funktionen vara *true* eller *false*, så det räcker att ange parameterns namn i *if*-satsens villkor.
- I *if*-delen lägger du in en loop där du går igenom alla element i *wordElems*, dvs alla *li*-element. I loopen lägger du in följande:
  - Referera till *wordElems[i]* och sätt egenskapen *draggable* till *true*.
  - Med *addListener* lägger du på händelsen *"dragstart"* och funktionen *dragStarted*.
- I *else*-delen lägger du in en likadan loop, fast du sätter då *draggable* till *false* och använder *removeListener*, för att ta bort händelsehanteraren.
- Glöm inte att också deklarera loopvariabeln med *var* i funktionen.

Funktionen *startGame*

- Sist i funktionen lägger du in ett anrop av *eventsForDrag* med parametern *true*.

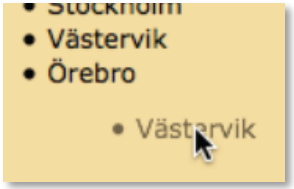
Funktionen *dragStarted*

```
function dragStarted(e) {
    e.dataTransfer.setData("text",this.innerHTML);
    dragWordElem = this;
} // End dragStarted
```

- I denna funktion lägger du in vidstående kod.
  - *this* är en referens till det *li*-element som man ska dra. Ordet i det elementet sparas i Event-objektets egenskap *dataTransfer*.
  - Referensen till *li*-elementet sparas också i den globala variabeln *dragWordElem*, som sedan behövs i en övning längre fram.

Testa

- Ladda om sidan i webbläsaren och klicka på start-knappen.
- Prova sedan att dra ett ord.



7b. Släpp ord över en bild

Nu ska du lägga till kod, så att man också kan släppa orden över en bild.

Funktionen *eventsForDrag*

- I *if*-delen lägger du till en loop där du går igenom alla *picsElems*, dvs alla bilder.
  - I loopen lägger du två händelsehanterare på *picsElems[i]*. Den ena ska vara *"dragover"* och funktionen *wordOverPict* och den andra ska vara *"drop"* och funktionen *wordOverPict*.
    - Egentligen ska vi endast använda *drop*, men båda händelsehanterarna måste ändå läggas på.
    - Det är samma funktion för båda händelserna.
- I *else*-delen lägger du in en likadan loop, fast använder *removeListener*.

Funktionen *wordOverPict*

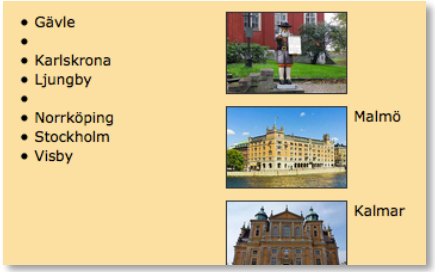
- I funktionen lägger du in vidstående kod.
  - Med *preventDefault* förhindras det som webbläsaren eventuellt annars gör, då man släpper ett objekt i fönstret.
  - I funktionen *dragStarted* sparade du en referens till det element som dras i variabeln *dragWordElem*. Den används nu för att ta bort ordet ur listan.

```
function wordOverPict(e) {
    var i;
    e.preventDefault();
    if (e.type == "drop") {
        dragWordElem.innerHTML = "";
        i = this.id;
        userAnswerElems[i].innerHTML = e.dataTransfer.getData("text");
    }
} // End wordOverPict
```

  - Med *this* får vi en referens till den *img*-tagg där ordet släppts, men det vi behöver är index till den *span*-tagg där ordet ska skrivas. För att få fram det, har indexen sparats i *img*-taggarnas *id*-attribut. Se HTML-koden.
  - I *span*-taggen (*userAnswerElems*) skrivs det ord som tidigare sparades i *dataTransfer*.

Testa

- Ladda om webbsidan och klicka på start-knappen.
- Dra sedan ord till bilderna.



fortsättning på nästa sida ...

7c. Släpp ord över en bild

... fortsättning från föregående sida

Då du testar, ser du nog att du kan dra ord även till en bild som redan har ett ord. Då läggs det nya ordet intill bilden. Detta är korrekt, men det gamla ordet ska då flyttas tillbaks till listan.

Funktionen *wordOverPict*

- Lägg in en *if*-sats mellan raden där *i* tilldelas och *userAnserElems* tilldelas, dvs innan det nya ordet läggs in.
- I *if*-satsen kontrollerar du om *userAnserElems[i].innerHTML* är skilt från "" (två citattecken, utan något mellan dem), dvs om det inte är tomt, utan det finns ett ord intill bilden.
- I så fall anropar du funktionen *moveBackToList*. Som parameter skickar du med det ord som finns intill bilden, dvs det du testar i *if*-satsen villkor.

Funktionen *moveBackToList*

- Lägg in koden som visas i vidstående bild.
  - Du söker efter den plats där ordet ska läggas in, genom att söka efter ordet i *words8*.
  - Det index som då erhålls används som index till *wordElems*.

```
function moveBackToList(word) {
    var i;
    i = words8.indexOf(word);
    wordElems[i].innerHTML = words8[i];
} // End moveBackToList
```

Testa

- Ladda om webbsidan och klicka på start-knappen.
- Prova nu att dra ord till bilderna och även till bilder där du redan lagt ett ord.

8. Rensa för nytt spel

Om du klickar på knappen "Starta spelet" igen, så ser du att de gamla orden ligger kvar intill bilderna. *Span*-taggarna där måste alltså tömmas, då man startar spelet.

Funktionen *startGame*

- I den sista loopen, där du lägger in bilderna i *img*-taggarna, lägger du till två rader.
- Där refererar till *userAnswerElems* och *correctAnswerElems*, som du indexerar med *i*.
  I deras *innerHTML* lägger du in "" (två citattecken, utan något mellan dem).

Testa

- Testa i webbläsaren.

9a. Kontrollera svaren

Nu ska du skriva koden för funktionen *checkAnswers*, där du ska kontrollera om användarens svar är korrekta, dvs om rätt stadsnamn är draget till de olika bilderna.

Funktionen *checkAnswers*

- Deklarera variablerna *i* och *points*, som ska användas som loopvariabel och poäng för antal rätt.
- Lägg in en loop där du går igenom alla *userAnswerElems*, dvs *span*-taggarna med de ord som användaren dragit till bilderna. I loopen gör du följande:
  - I en *if*-sats kontrollerar du om *span*-taggen är tom.
  - I så fall skriver du med *alert* "Dra först ord till alla bilder." och sedan avbryter du funktionen med *return*.
- Efter loopen anropar du *eventsForDrag* med parametern *false*.
  - Då tar du bort händelsehanterarna för drag and drop, så man inte längre kan dra orden.
- Sätt *points* till 0.
- Skriv sedan en loop där du går igenom alla *userAnswerElems* igen. I loopen lägger du in följande två programsatser:

```
if (userAnswerElems[i].innerHTML == allPics[picsIx4[i]]) points++;
correctAnswerElems[i].innerHTML = allPics[picsIx4[i]] + "<br>" + allDescription[picsIx4[i]];
```

  - För att hänga med i alla referenser här, kan du behöva titta tillbaks på figuren i övning 4, där de globala variablerna illustreras.
  - *userAnswerElems[i].innerHTML* är det ord som finns i *span*-taggen.
  - *picsIx4[i]* är ett index till det ord som hör till bilden. Så för att få fram ordet, får man använda *picsIx4[i]* som index till *allPics*.
  - Om dessa ord är lika, räknas poängen upp med 1.
  - På nästa rad skrivs det rätta svaret ut tillsammans med den information som finns i *allDescriptions*.
- Efter loopen skriver du i elementet för meddelanden (*msgElem*) hur många poäng användaren fick.

fortsättning på nästa sida ...

9b. Kontrollera svaren

... fortsättning från föregående sida

Testa

- Testa i webbläsaren.
- Då du dragit ord till bilderna klickar du på knappen "Kontrollera svar".

10. Aktivera/inaktivera knaparna

De båda knapparna ska nu aktiveras och inaktiveras i olika skeden i programmet. Detta gör du genom att ändra egenskapen *disabled* mellan *true* och *false*. Du gör alltså på samma sätt som du gjorde i laboration 3.

Funktionen *init*

Sist i funktionen lägger du in kod, så att start-knappen blir aktiv och kontroll-knappen blir inaktiv.

Funktionen *startGame*

Sist i funktionen lägger du in kod, så att start-knappen blir inaktiv och kontroll-knappen blir aktiv.

Funktionen *checkAnswers*

Sist i funktionen lägger du in kod, så att start-knappen blir aktiv och kontroll-knappen blir inaktiv.

Testa

Testa i webbläsaren.

11. Extra funktionalitet för drag and drop

Programmet är nu så pass långt kommet så att det är ett helt fungerande spel. Men det finns en sak till som kan läggas till, för att öka funktionaliteten vid drag and drop.

Orden kan nu dras från listan till bilderna. Du ska nu lägga till kod, så att man också kan dra orden mellan bilderna och tillbaks till listan. För att göra detta behöver du lägga på händelsehanterare på *span*-taggarna med orden intill bilderna, så att de kan dras. Du behöver också lägga på händelsehanterare på listan, så ord kan släppas där. Slutligen behöver du skriva kod för funktionen *wordOverList*, som används för den sista händelsen.

Detta kanske låter som en hel del, men det är egentligen inte så mycket som behöver läggas till.

Funktionen *eventsForDrag*

- I *if*-delen lägger du in följande:
  - En loop där du går igenom *userAnswerElems*, dvs alla *span*-taggar för orden intill bilderna. I loopen sätter du *draggable* till *true* och lägger på händelsen *"dragstart"* och funktionen *dragStarted*.
    - Loopen blir alltså likadan som den första loopen, fast du har nu *userAnswerElems* istället för *wordElems*.
  - Efter loopen lägger du på två händelsehanterare på *wordListElem* (dvs *ul*-listan). Händelserna ska vara *"dragover"* och *"drop"* och funktionen för båda ska vara *wordOverList*.
- I *else*-delen lägger du in motsvarande kod, fast sätter *draggable* till *false* och använder *removeListener*, för att ta bort händelsehanterarna.

Funktionen *wordOverList*

Denna funktion liknar *wordOverPict*, fast blir lite enklare.

- Skriv in koden som visas i bilden här intill.

```
function wordOverList(e) {
    e.preventDefault();
    if (e.type == "drop") {
        dragWordElem.innerHTML = "";
        moveBackToList(e.dataTransfer.getData("text"));
    }
} // End wordOverList
```

Testa

- Testa i webbläsaren.

12. Publicera och testa

Då du är klar med ditt program publicerar du det 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 *lab4* 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 *lab4*).

Testa

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