***Uppgift U3***

***Interaktivt program***

Syftet med uppgiften är att du ska tillämpa både teori och praktiska moment i kursen. Du ska skapa ett program och genom det visa att du kan sätta ihop ett program utifrån de anvisningar som ges i uppgiften.

Uppgiften bedöms med betygen U, G eller VG, så kraven är också uppdelade i krav för G och krav för VG. Se längre ner på denna sida.

***Uppgiftsbeskrivning***

I uppgiften ska du skapa ett program för spelet Memory. Det finns ett antal brickor med bilder. Varje bild finns på två av brickorna, så är det 16 brickor, finns det 8 par av bilder. Brickorna ligger med baksidan upp, så man inte ser bilderna. Sedan kan man vända på två brickor i taget. Är det samma bild på båda, tas brickorna bort. Är det olika bild, vänds brickorna igen och man kan sedan vända två nya brickor. Detta fortsätter tills alla par är funna.

I programmet ska det också räknas hur många gånger som man vänder två brickor, innan alla par är funna. Utifrån detta beräknas det sedan en poäng.

Poängen för varje spel summeras till en total poängsumma. Denna summa ska sparas i en cookie eller web storage och sedan läsas in nästa gång man kommer till webbsidan igen.

***Arbetsmaterial***

Genom denna länk laddar du ner en zip-fil med några filer som du utgår ifrån. Efter nedladdning får du packa upp filen, för att få en mapp med filerna.

[U3download.zip](http://medieteknik.lnu.se/1me322/uppg-lab/u3/U3download.zip)

I detta material finns det en html-fil, en css-fil, en js-fil och en mapp med bilder för brickorna. HTML- och CSS-koden är klara, så du sätter dig endast in i dem, för att se hur de är uppbyggda. Du får inte förändra HTML- eller CSS-koden, men vill du göra egna tillägg i filerna är det tillåtet. Js-filen är tom och det är där du ska skriva den kod som programmet ska bestå av. Vill du dela upp det i flera js-filer, får du göra det.

***Programmets funktion***

Programmet och interaktionen med användaren visas enklast i en film, som kommer läggas in här senare.

Funktionaliteten beskrivs också i nedanstående punkter.

**Normalt flöde i spelet:**

* Användaren klickar på start-knappen. Då väljs bilder slumpmässigt för brickorna, så att bilder kopplas till dem, men det är fortfarande brickornas baksida som ska visas. Är det 16 brickor, ska det vara 8 par av bilder. Varje bild ska alltså kopplas till två brickor.
* Användaren klickar på de två brickor man vill vända. Då man klickar på en bricka, visas bilden för den.
* Varje gång man vänt två brickor, räknas en räknare upp, som på sidan visar hur många gånger man vänt två brickor.
* Brickorna jämförs, för att avgöra om bilderna som hör till dem är lika eller ej.
* Användaren klickar på nästa-knappen, för att gå vidare. Är brickorna lika, tas de bort, annars vänds de tillbaks, så baksidan visas igen.
* Då sista paret är funnet, räknas det ut poäng enligt följande formel:
  + 20 - (antal vändor - antal par) \* 1.2
  + Detta avrundas till närmsta heltal.
  + Blir resultatet negativt, sätts poängen till 0.

**Aktivering och inaktivering av olika delar i gränssnittet**

* Då sidan laddats in ska man kunna göra val i menyn för antal brickor (endast VG) och man ska kunna klicka på start-knappen (både G och VG). Om man klickar på brickorna, ska det inte hända något. Nästa-knappen ska vara inaktiverad.
* Då man startat ska man kunna klicka på brickor, för att vända dem. Menyn och start-knappen ska då vara inaktiverade.
* Då två brickor vänts, ska man inte kunna vända fler. Det ska inte hända något, om man klickar på en bricka. Nästa-knappen aktiveras också.
* Då man klickar på nästa-knappen, inaktiveras den. Det blir också möjligt att klicka på brickor, för att vända dem igen.
* Då sista paret är funnet och poängen skrivs ut, ska menyn och start-knappen aktiveras igen.

**Menyn för antal brickor**

* *Denna funktion ingår endast i VG-kraven.*
* Då man gör ett val i menyn, ska antalet brickor på spelplanen uppdateras, så att det visas så många som man valt.
* Det antal brickor som man valt, ska sedan användas då spelet startas.

**Användarinfo**

* *Användarinfon visas i rutan i högra hörnet på sidan.*
* Användarinfon ska sparas i en cookie eller i web storage. Den ska läsas in då sidan laddats in. Den ska sparas, då det görs någon förändring av infon.
* För G:
  + Användarinfon består endast av totalt antal poäng. Varje gång spelet är klart och det räknats ut poäng för spelet, ska det summeras till den totala poängen.
* För VG:
  + Utöver total poäng, består användarinfon också av antal spel och medelpoäng per spel.
  + I cookien eller web storage sparas både total poäng och antal spel.
  + Medelpoängen beräknas med hjälp av de båda andra värdena.
  + I rutan för användarinfo finns det en länk, för att utöka eller minska rutan. Klickar man på denna ska rutans storlek förändras, genom att fälten för antal spel och medelpoäng visas/eller döljs.

Ovanstående är en beskrivning av hur programmet och interaktionen ska fungera. Din uppgift är att skriva den JavaScript-kod som behövs för att implementera detta. Se också de tips som ges i rutorna i sidokolumnen.

***Krav för uppgiften***

Kraven är i denna uppgift uppdelade i krav för betyget G respektive VG. Alla krav för G är obligatoriska. Kraven för VG är valfria, men för att kunna erhålla betyget VG, måste alla krav (både de för G och VG) vara uppfyllda i din redovisning. Det görs också en kvalitativ bedömning av hur pass väl du uppfyllt kraven och hur välskriven din programkod är.

**Krav för G**

* De allmänna kraven för uppgifterna måste vara uppfyllda. Se [sidan med uppgiftsbeskrivning](http://medieteknik.lnu.se/1me322/uppg-lab/index.htm#generalDemands).
* Av det som beskrivs under Programmets funktion ovan, ska allt utom det som markerats som VG-delar, finnas med.

**Krav för VG**

Utöver kraven för G, måste även följande krav vara uppfyllda.

* Alla delar, som beskrivs under Programmets funktion ovan, måste finnas med.

För bedömning av betyg kommer programmets kod bedömas. Då granskas hur det är strukturerat, användning av variabler, namngivning av funktioner och variabler, kommentarer i koden och hur funktionaliteten är implementerad.

**Rapport**

Du ska också skriva en rapport, där du redogör för hur du löst uppgiften. I rapporten skriver du fritt om hur du strukturerat ditt program, vilka val du gjort och motiverar dessa. För VG måste du också ha med referenser till kursboken Beginning JavaScript. Referenserna måste skrivas enligt Harvardsystemet.

Omfattning på rapporten är 2-3 A4-sidor för G och 3-4 A4-sidor om du dessutom gjort VG-delen.

***Redovisning***

Du redovisar uppgiften på följande sätt:

1. Publicera ditt program för u3 i din dolda (lösenordsskyddade) mapp i Web Publishing i FirstClass.
2. Lägg in en länk till programmet på ingångssidan i din portfolio (index.htm i Web Publishing).
3. Kontrollera i webbläsaren att de publicerade sidorna är korrekta och programmet fungerar.
4. Muntlig examination i enskilt möte med läraren (c:a 5 min.). Du får då förklara några delar av din kod.
5. Lägg in ett meddelande med rapporten i mappen U3 i kursmappen i FirstClass.
   * Antingen klistrar du in rapportens text i meddelandet eller så bifogar du rapporten som en bilaga.
   * Om du bifogar rapporten, måste den vara sparad i pdf-format.

Datum för muntlig examination och deadline för rapporten ges i schemat.

**Tips 1: Arbetsgång**

En rekommendation är att du arbetar med programmet i ett antal steg i den ordning som det beskrivs i avsnittet Programmets funktion. Börja med att skapa programmet för det "normala flödet" och bry dig då inte om aktivering och inaktivering av knappar, etc.

Då grundprogrammet är klart, går du vidare med att komplettera programmet med aktivering och inaktivering av de olika delarna i gränssnittet.

Fortsätt sedan så med en del i taget. Hantering av användarinfo med total poäng implementerar du alltså sist, då du först fått de andra delarna klara.

**Tips 2: Val av bilder**

Bilderna för brickorna ska väljas slumpmässigt. Jämför med exemplet i en föreläsning, där spelkort dras, men tänk på att varje bild ska förekomma två gånger.

Har du 16 brickor, använder du bild 0 till 7, har du 20 brickor använder du bild 0 till 9, osv.

En framtida utveckling, då du fått allt annat att fungera, kan vara att lägga till kod för att slumpmässigt välja vilka av de 21 bilderna som ska användas.

**Tips 3: Bilder som hör till brickorna**

Varje bricka består av en img-tagg. Då brickans baksida visas, ska det vara bildfilen backside.png. För att spara information om vilken bild som finns på framsidan, kan du använda img-taggens id-attribut.

**Tips 4: Brickornas status**

En bricka kan visa baksidan, framsidan eller vara borttagen.

Då baksidan visas, ska det vara bildfilen backside.png. Då ska du också använda classen "brickBack". Då framsidan visas ska det vara den bildfil som du slumpmässigt valt till brickan samt classen "brickFront". För en borttagen bricka är det bildfilen empty.png samt classen "brickEmpty".

Då du byter bild i img-taggen, ska du alltså också byta class. För att referera till classen används egenskapen className i JavaScript.

**Tips 5: Aktivering och inaktivering av delar i gränssnittet**

För knappar och menyn kan du använda egenskapen disabled.

För att se om en bricka kan vändas, kan du kontrollera om baksidan visas. Vill du förhindra alla brickor, kan du använda en flaggvariabel, som du ändrar vid olika lägen i programmet. Flaggan kontrollerar du i funktionen för brickorna.

**Tips 6: Ändring av antal brickor**

Brickorna ligger som img-taggar inom div-elementet med id "bricks". Det är bredden på det elementet som avgör hur många brickor det får plats på en rad. Så då du ändrar antal brickor genom menyn, kan du också behöva ändra bredden på "bricks"-elementet.