Linnéuniversitetet
Kalmar Växjö

Projektanvisning

Fönsterhanterare i JavaScript

PWD, Personal Web Desktop

Författare: Johan Leitet
Kurs: Webbteknik I
Kurskod: 1dv403

Inledning

Detta projekt kommer att repetera många kursen moment men även introducera några nya delar. Uppgiften är dessutom av en något större art än vad du tidigare gjort på laborationerna varför ett strukturerat arbetssätt är viktigt, speciellt gäller detta för betygsnivå 4 och 5.

Mål

Efter genomfört projekt kommer du att ha befäst de kunskaper som du tillgodogjort dig tidigare i kursen. För de högre betygen kommer du även att testas i att skriva strukturerad kod uppdelad i flera objekt samt få erfarenhet av "drag and drop"-funktionalitet.

Du kommer även att få testa på att göra asynkrona anrop till en webbserver med hjälp av XMLHttpRequest-objektet och tolka och arbeta med JSON.

Förberedelse

Du bör innan du påbörjar projektet ha genomgått i stort sett all teori som kursen behandlar. Läs även igenom **hela** laborationshandledningen innan du påbörjar laborationen.

Genomförande

Utför projektets uppgifter och moment samt dokumentera vad du kommer fram till på de olika delarna. Vid redovisning av projektet ska du kunna besvara frågor om hur du har löst de olika delarna och varför de är lösta på det sätt du löst dem på.
Givetvis är din kod snyggt formaterad och kommenterad där kommentarer fyller ett syfte.

När du anser dig vara klar med projektet, kontrollera att din källkod uppfyller laborationens samtliga krav. Redovisning sker muntligt på samma sätt som föregående laborationer.

Projektet är betygssatt med betygsgraderna U/3/4/5. Kraven för de olika delarna framgår av denna laborationshandledning. Kraven är inte absoluta. Examinator har möjlighet att sänka/höja betyget om anledning finns oavsett om samtliga krav är uppfyllda eller ej. En anledning till sänkning kan t.ex. vara en buggig applikation eller en applikation som inte är visuellt tilltalande.

Projektet ska lösas enskilt. Vid fuskmisstanke lämnas misstankar samt berörda dokument över till universitetets disciplinnämnd.

Exempelapplikation

På kursens webbplats hittar du en demonstration av denna applikation. Applikationen skapades av Thomas Dahlberg (http://dubbe.se) 2010 och skärmdumpar i denna handledning kommer från Thomas applikation.

Uppgiften

Din uppgift i detta projekt blir att skapa ett virtuellt skrivbord på webben.

Beroende på betygsnivå kommer du att bygga in mer och mer avancerad hantering av fönster och innehåll.
I grundutförande kommer uppgiften enbart att bestå av ett fönster som visar bilder och gör det möjligt att byta skrivbordsbakgrund. I mer avancerat utförande kommer användaren att kunna skapa nya fönster, flytta och skala om dessa samt påverka innehållet i fönstret.

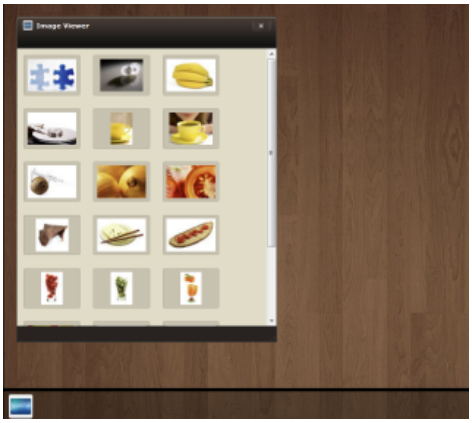
Uppgiften kan vid första anblicken kännas otroligt stor och komplex, men du ska se att om du implementerar del för del metodiskt, genomtänkt och lugnt så är det fullt genomförbart.

Det är inget krav att denna uppgift ska fungera utan JavaScript eller att användaren ska kunna använda applikationen utan mus men detta kan implementeras som ett extra krav.

Tips:
När en programmerare ställs inför uppgifter som denna är det otroligt viktigt att strukturera upp sitt arbete. Ett anteckningsblock och en penna framför sig när man arbetar är ett måste. Då kan man nämligen enkelt skissa upp applikationens delar, skriva små korta minnesanteckningar och skissa på lösningsförslag.

Ett viktigt tips är alltså att rita, rita, rita för att åskådliggöra problemet från olika synvinklar.

För betygsgrad 3



För betyg 3 ska du skapa en applikation som har ett skrivbord med fastslagen höjd och bredd samt med möjligheten att klicka på en ikon som öppnar ett fönster innehållandes tumnagelbilder som ska vara klickbara.
När en tumnagelbild klickas på ska bakgrunden ändra bild till den bild som tumnageln föreställer.
Det ska gå att stänga fönstret och sedan öppna det igen genom att klicka på ikonen. Enbart ett fönster behöver kunna vara öppet samtidigt.

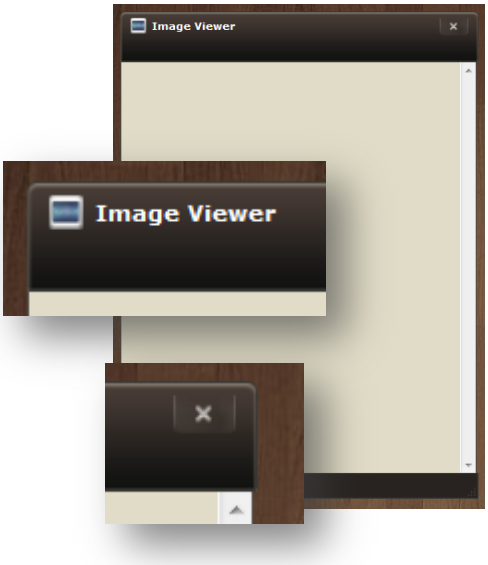
Applikationen ska fungera i senaste versionen av Firefox, Internet Explorer eller Chrome.
Ingen JavaScriptkod får finnas i HTML-dokumentet och inga variabler eller funktioner (förutom ett antal objekt) får skapas på window-objektet. (Dessa får alltså inte vara ”globala”)
ES5 Strict Mode ska användas. (”use strict”;)
Samtliga funktioner och variabler ska vara inkapslade i ett eller flera objekt. Om du planerar att fortsätta för betyget 4 eller 5 bör du redan nu titta igenom anvisningarna för dessa betygssteg och då främst ”En objektorienterad design” för att du ska slippa skriva om grunden i din applikation för dessa betygsnivåer. Titta främst på det som rör programmets uppbyggnad.

Desktopen

När applikationen först laddas så ska ett skrivbord stort som webbläsarfönstret alternativ ett skrivbord med fast bredd och höjd (t.ex. 1024x640) visas på skärmen. På denna yta ska en ikon finnas.

Du får använda valfritt skrivbordsunderlägg samt valfri ikon. Tänk dock på att det är viktigt att du har rättighet att använda bilderna.

Fönstret



När användaren klickar på ikonen så ska ett fönster visas för användaren. Detta fönster ska ha en tilltalande layout och minst innehålla följande:

- En ram runt hela fönstret.
- En liten ikon i övre vänstra hörnet tillsammans med en text.
- En knapp för att stänga fönstret.
- En statusrad i vilken det ska gå att skriva ut information.
- En tom yta i vilken man ska kunna lägga in valfritt innehåll.

Då användaren klickar på stängknappen ska fönstret stängas och det ska gå att öppna ett nytt.

Tumnagelbilderna

När fönstret väl har laddats så ska ett ajaxanrop mot adressen `"http://homepage.lnu.se/staff/tstjo/labbyServer/imgviewer/"` göras. Detta anrop kommer att returnera JSON innehållandes samtliga bilder i katalogen "pics/thumbs" tillsammans med deras storlek. Det är en array som ges och varje objekt i denna array är definierat enligt följande:

```
{
   "URL":"http://.../1050509.jpg",
   "width":500,
   "height":332
   "thumbURL":http://.../1050509.jpg,
   "thumbWidth":75,
   "thumbHeight":50,
}
```

- `"URL"` URL till orginalbilden.
- `"width"` innehåller orginalbildens bredd.
- `"height"` innehåller orginalbildens höjd
- `"thumbURL"` URL till tumnagelbilden.
- `"thumbWidth"` innehåller tumnagelbildens bredd.
- `"thumbHeight"` innehåller tummnagelbildens höjd.

Surfa gärna in direkt till sidan för att se hur datat du får tillbaka ser ut.

För att tolka JSON-strängen som kommer från servern ska du använda JSON.parse() eller annan för ändamålet avsedd JSON-tolk (vilket exkluderar eval).

Tumnagelbildens bredd och höjd behövs för att bestämma hur stora de inneslutande boxarna ska vara i vilka tumnaglarna ligger. Dessa boxar ska nämligen anpassa sig dynamiskt efter den högsta respektive den bredaste tumnageln enligt följande:

Den högsta bilden är t.ex. 80px och den bredaste är 200px. Således ska samtliga boxar vara 200px breda och 80px höga. Vid presentation kan då en hög men smal tumnagel se ut som bilden ovan (här med några pixlars padding). Observera att nya bilder med andra mått kan komma att läggas till så din applikation måste kunna anpassa sig till detta.

Om tumnagelbilderna tar upp större områden än vad som finns tillgängligt i fönstret ska fönstret få en scrollbar så att samtliga bilder kan nås. Bilderna ska ligga uppradade.

Tumnagelbilderna laddas inte genom XHR-objektet utan helt enkelt genom att img-taggar skapas dynamiskt och dess src-attribut sätts.

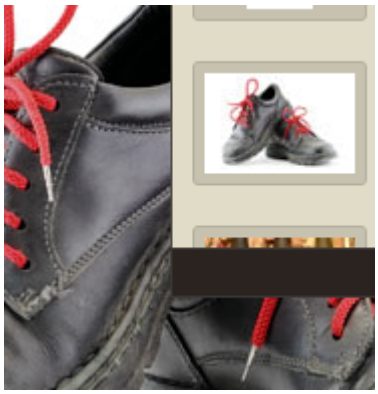
Om ajax-anropet till servern drar ut på tiden ska detta presenteras genom t.ex. en animerad gif-bild: (www.ajaxload.info)

Tips:
Servern där bildfilerna finns använder http-headern ”Access-Control-Allow-Origin: *” vilket möjliggör att ajax-anrop kan göras från skript som inte kommer från denna server. Servern är även inställd på att simulera olika svarstider (0-3000ms) så att du har möjlighet att testa att din laddningsikon fungerar.

Ändra skrivbordsbild

När tumnaglarna är färdigladdade och användaren klickar på en utav dem så ska ”skrivbordsbakgrunden” ändras till denna bild.

Om bilden är mindre än skrivbordsytan så ska bilden repeteras för att fylla ut.



Kontrollera kraven för betygsgrad 3 (bocka av)

- ☐ Applikationen fungerar i senaste versionen av Firefox, Internet Explorer eller Chrome.
- ☐ ES5 Strict mode används.
- ☐ Det finns ingen JavaScript-kod i .html-dokumentet.
- ☐ Inga objekt eller variabler, förutom de statiska objekt som används för uppgiften är skapade globalt på window-objektet.
- ☐ Man kan stänga ett fönster genom att klicka på stängknappen.
- ☐ Min applikation uppfyller i övrigt de krav som är satta enligt beskrivningen ovan.
- ☐ När Ajaxanropet dröjer ska användaren göras uppmärksam på detta. T.ex. genom en animerad gif-bild.
- ☐ Samtliga tumnagelbilders boxar är lika stora och dess storlek räknas automatiskt ut med hjälp av de storleksangivelser som tagits emot i JSON-strängen.
- ☐ Förutom funktionskraven är applikationen visuellt tilltalande och ger ett proffsigt intryck.

För betygsgrad 4

För detta betygssteg krävs det även att kraven för betygssteg 3 är uppfyllda.

Du ska för betyg 4 även implementera en RSS-läsare i din applikation. Användaren ska även ges möjlighet att öppna ett ”obegränsat” antal fönster. Det ska dessutom vara möjligt att öppna bilderna i bildvisaren i egna, anpassade fönster. Memoryspelet som du tidigare skapat ska gå att öppna i flera olika instanser i fönsterhanteraren.

Som om inte detta vore nog så ska du utöka funktionaliteten med något som du själv hittar på.

En objektorienterad design

Eftersom du nu kommer att hantera många olika fönster så är det viktigt att noga tänka igenom designen av applikationen. Du kommer t.ex. att skapa en RSS-läsare som ska gå att öppna i flera fönster. Det är därför viktigt att inställningarna ska vara unika för varje fönster (t.ex. ska olika rss-feeds kunna läsas in i olika fönster). Ett lämpligt sätt att lösa detta på är genom att låta varje fönster vara en instans av en konstruktorfunktion. Du ska alltså för denna betygsnivå se till att varje fönster som skapas är en instansiering utifrån en konstruktorfunktion. Exempelvis:

```
myWindow = new Window();
```

Eftersom våra fönster i mångt och mycket kommer att skilja sig åt så kan det vara lämpligt att låta specialiseringar av våra fönster ärva basfunktionalitet från Window-konstruktorfunktionen så att vi kan skapa instanser enligt:

```
myRSSWindow = new RSSWindow();
```

Där RSSWindow är en specialisering av Window.

Du ska på denna och nästföljande betygsnivå se till att din applikation har en lämplig objektorienterad programarkitektur.

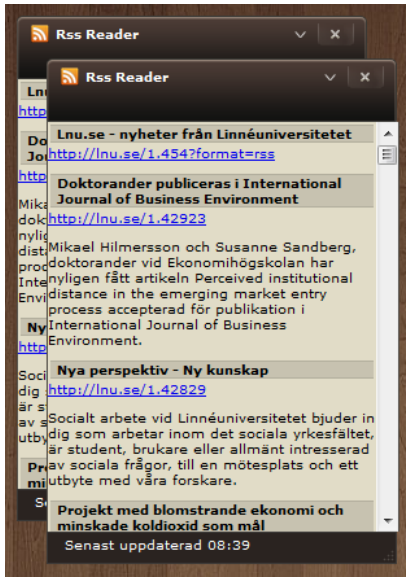
Namnrymd

På detta och kommande betygssteg ska vår nedsmutsning av det globala objektet vara minimal och du ska därför se till att din applikations samtliga delar är inkapslade i en namnrymd. Namnge namnrymden på ett vettigt sätt, helst med VERSALER och se till att skapa en bra mappstruktur som stämmer överens med din namnrymnd.

På windowobjektet ska det alltså inte tillkomma något mer än en egenskap när din applikation körs. Exempelvis skulle detta kunna vara: window.JAWM

RSS-läsaren

RSS-läsaren ska uppdateras dynamiskt via AJAX. För att hämta en speciell RSS-feed så skickar du ett anrop till



”`http://homepage.lnu.se/staff/tstjo/labbyServer/rssproxy/`” och skickar med getvariablen ”url” satt till den feed som du vill läsa av.

Genom att ”url-avkoda” RSS-länken som vi vill skicka så går det att skicka denna via url:n. Urlavkodningen görs i javascript med funktionen escape. (Se kodförslaget nedan)

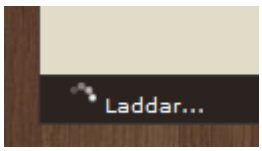
Följande kod ger adressen för att hämta RSS-feeden från Dagens Nyheter:

```
"http://homepage.lnu.se/staff/tstjo/labbyServer/rssproxy/?url="+escape("http://www.dn.se/m/rss/senaste-nytt")
```

HTML-koden som du får tillbaka kan du sedan stoppa in i ditt tomma fönster och stila om med CSS. Se den genererade källkoden för vilka css-klasser och id:n som kan användas.

Du kan enkelt hitta andra RSS-flöden genom att leta efter den ofta orangea ikonen på sajter som du brukar besöka. Osäker på vad RSS är? Använd http://www.google.com

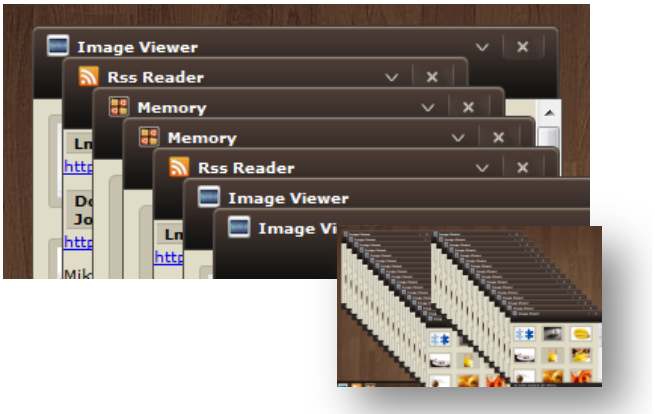
Precis som för bildgalleriet så ska en ikon visas då laddningen drar ut på tiden. (vilket ofta är fallet)



För att användaren ska kunna öppna olika RSS-flöden så kan du lägga ut olika ikoner på skrivbordet som öppnar olika RSS-flöden. (För betyg 5 ska det gå att ladda olika RSS-flöden i samma RSS-läsare)

Flera instanser

I detta betygssteg ska det vara möjligt att öppna flera instanser av både RSS-läsaren och bildvisaren och Memoryt.



När en ny instans av ett fönster visas ska detta läggas något förskjutet i förhållande till det senast öppnade fönstret. Om så många fönster öppnas att dessa

riskerar att hamna utanför skrivbordsytan så ska fönstren antingen ”studsa” mot botten eller börja om från toppen.

Fokusera fönster

Då man nu kan ha flera fönster som ligger ovan på varandra ska det vara möjligt att ge ett speciellt fönster fokus genom att klicka på det. När ett fönster får fokus ska detta hamna ovanför samtliga andra fönster.

Uppdatering av bildvisaren

Bildvisaren ska uppdateras så att när användaren klickar på en bild så ska denna inte längre hamna som bakgrundsbild utan öppnas i ett nytt fönster. Fönstret ska ändra storlek så att bilden får plats.

Här kan ett tips vara att använda de storleksangivelser för ursprungsbilden som följer med vid AJAX-inladdningen av tumnagelbilderna. (Se betygssteg 3)

Memoryt

Om du på laboration ”Memory” gjort stjärnuppgiften kommer du nu att få nytta av denna. I annat fall bör du nu gå tillbaka till den laborationen och se till att du kan skapa flera instanser av ett och samma Memoryspel på samma sida. Du ska nämligen i detta steg se till att du kan ha flera fönster med Memoryspel öppna samtidigt och att dessa kan spelas oberoende av varandra.

Övrig funktionalitet

Förutom de krav som angetts i uppgiften måste du för att uppfylla betygskriteriet även implementera någonting förutom kraven.

Exempel på detta kan vara:

- Ytterligare en applikation.
- Högerklick på tumnaglarna för att välja bild som skrivbordsbild.
- Information om aktuell bild i statusraden då muspekaren förs över tumnageln.
- Maximeraknapp bredvid stängknappen för att låta aktuellt fönster täcka hela skrivbordet.
- Egen funktionalitet. Är du osäker om det du vill göra är tillräckligt så kontakta kursansvarig för bekräftelse.

Kontrollera kraven för betygsgrad 4 (bocka av)

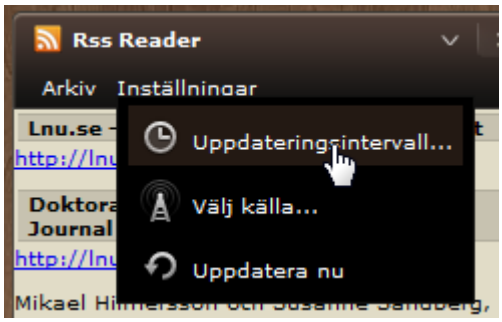
- □ Samtliga krav för betyg 3 (förutom skrivbordsbildsbyte) är uppfyllda.
- □ Applikationen fungerar i såväl senaste versionen av Firefox som i senaste versionen av Internet Explorer och Chrome.
- □ En objektorienterad applikationsdesign används för fönsterhanteringen.
- □ Det går att öppna nya fönster som hamnar förskjutna i förhållande till sina syskon och dessa fönster hamnar aldrig utanför skrivbordets yta.
- □ Fönster kan få fokus genom att användaren klickar på dem
- □ De olika fönstren är inte beroende av varandra. (Man kan t.ex. ha olika RSS-flöden i olika fönster samtidigt eller spela flera memoryspel samtidigt.)
- □ Min applikation uppfyller i övrigt de krav som är satta enligt beskrivningen ovan.
- □ Förutom funktionskraven ska applikationen vara visuellt tilltalande, lättarbetad och ge ett proffsigt intryck.

För betygsgrad 5

För detta betygssteg krävs det även att kraven för betygssteg 3 och 4 är uppfyllda.

Nu ska vi lägga till den sista kryddan till vår applikation för att den ska kännas riktigt användbar. I detta sista betygssteg ska vi framförallt koncentrera oss på hanteringen av fönstren och då möjligheten att flytta fönstren samt ändra storlek på dessa. Du kommer även att få komplettera projektet med en förändrad version av ”Labby Mezzage” för att få ett mikrobloggsystem tillsammans med övriga betyg-5-laboranter.

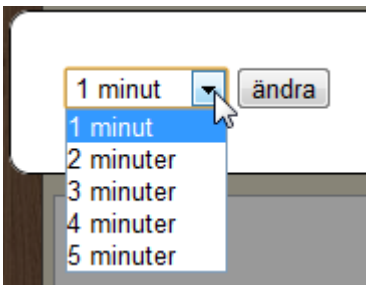
RSS-läsaren - Contextmeny



RSS-läsaren ska ha en meny i vilken det går att göra tre val. ”Uppdateringsintervall...”, ”Välj källa...” och ”Uppdatera nu”
De två förstas alternativen ska öppna en konfigurationsruta inuti läsaren medan det tredje alternativet verkställs direkt när man klickar på knappen.

Då användaren klickar utanför menyn eller väljer ett alternativ ska menyn stängas.

Menyalternativen ska ha följande funktionalitet:

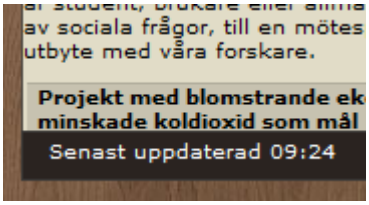


- *Uppdateringsintervall…*
  Här ska användaren kunna ställa in hur ofta det aktuella flödet ska uppdateras. Uppdateringsfrekvensen ska vara unik för varje öppnad instans av RSS-läsaren. Du väljer själv vilka uppdateringsintervall som ska finnas. (Minst två stycken)

- *Välj källa…*
  Här ska användaren kunna välja på några av dig utvalda RSS-flöden med hjälp av radioknappar. Det sista alternativet ska vara en textruta där användaren själv kan ange vilket flöde som ska laddas.

  När inställningarna sparas ska flödet laddas om.

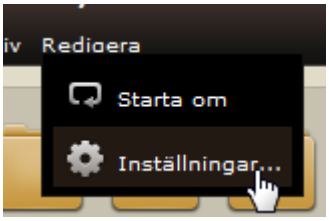
- *Uppdatera*
  En omedelbar uppdatering av RSS-flödet sker.



Det ska framgå av statusfältet när flödet senast uppdaterades:

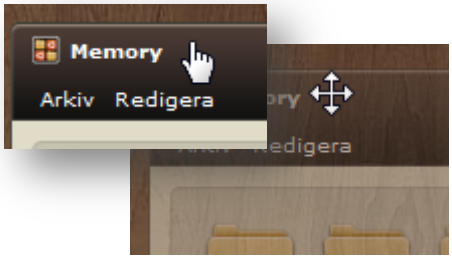
Contextmeny – Memoryt

Även Memoryt ska få en contextmeny som ska möjliggöra att spelet enkelt kan startas om. Här ska man även kunna ändra egenskaper på memoryt så som hur många brickor spelet ska bestå av.



Förflyttning av fönstren (Samtliga fönster)

Du ska nu implementera så kallad ”drag n drop”-funktionalitet vilket innebär att man ska kunna ta tag i toppdelen på fönstren och dra runt dem till valfri plats på skrivbordet. När användaren för muspekaren över fönstrets övre del ska muspekaren ändras till en hand och när sedan användaren trycker ner vänster musknapp så ska muspekaren ändras till en flyttikon enligt bilderna till höger.



Användaren ska sedan kunna dra runt fönstret **inom skrivbordsytan**.
När användaren släpper musknappen ska fönstret stanna kvar på vald position.

Tips:
Följande sida ger värdefull information om hur man t.ex. hittar muspekarens aktuella position:
http://www.quirksmode.org/js/events_properties.html

För att undvika att text och objekt markeras när du håller musknappen intryckt och flyttar omkring dina fönster kan följande metod vara användbar:

```
disableSelection:function(element)
{
   element.onselectstart = function() {return false;};
   element.unselectable = "on";
   element.style.MozUserSelect = "none";
   element.style.cursor = "default";
}
```

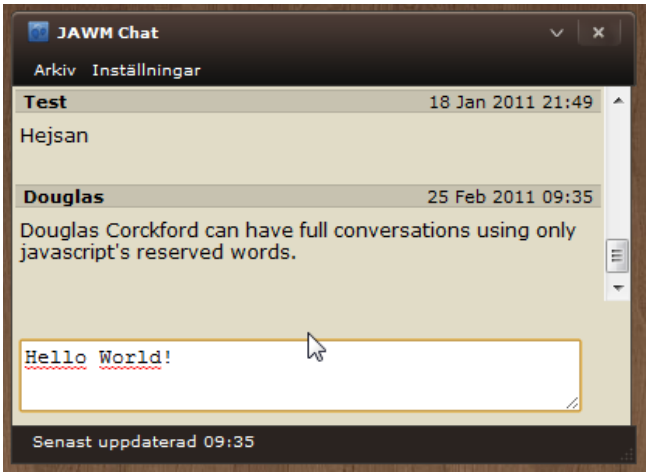
Ändra storlek på fönstren (Vissa fönster)

Nu är det dags att även bygga in stöd för att förändra storleken på fönstren. Längst ner till höger ska du göra ett område mottagligt för att ändra fönstrets storlek. Det bör tydligt framgå var detta område finns så att användaren direkt lägger märke till att funktionen finns.

När användaren för muspekaren över denna position ska den ändras till en ändrastorlekikon enligt bilden. Användaren ska sedan kunna förminska och förstora fönstret inom skrivbordets gränser. Du bör sätta en minimigräns för hur litet fönstret får bli.

Enbart fönster där det är meningsfullt att kunna ändra storleken ska ha denna funktion. Du bör alltså se till att kunna sätta en egenskap ”resizable” eller liknande på dina fönster.

Labby Mezzage, kopplad till server

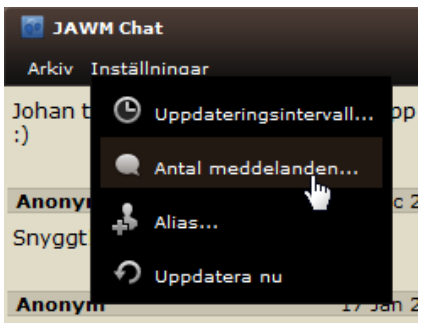


Du ska nu få användning av den meddelandeapplikation som du skrivit i en tidigare laboration. Tanken är att denna applikation ska kopplas mot en server och på så sätt spara ner meddelanden och läsa upp meddelanden. Eftersom samtliga laboranter kommer att arbeta mot samma server så kommer du att kunna läsa vad andra skriver och hålla en diskussion via meddelandesystemet. Detta blir helt enkelt ett enklare chat, eller mikrobloggsystem.

Uppgiften går ut på att med hjälp av XMLHttpRequest-objektet skapa uppkopplingar mot en server för att kunna läsa och skriva meddelanden. Eftersom vår applikation inte i nuvarande skick tål en sidomladdning så är det viktigt att kommunikationen sker asynkront. I denna uppgift blir du tvungen att själv sätta dig in i vissa delar av XHR-objektet som vi tidigare inte tittat på och du blir även tvungen att utforska andra delar som inte tagits upp i kursen. Detta är en naturlig del i att du satsar på att få betyget 5.

Kraven på detta steg är:

- Då du skriver ett meddelande i din mikroblogg ska detta skickas till serven asynkront för lagring.
- Din applikation ska automatiskt visa nya meddelanden för användaren oavsett vem som skrivit dessa.
- Det ska tydligt framgå vem som skrivit meddelandet och när.
- Användaren ska via contextmenyn kunna ange ett användarnamn och ett uppdateringsintervall för hur ofta applikationen ska leta efter nya meddelanden. Dessutom ska användaren kunna ange hur många meddelanden som ska visas.
- Inställningarna i contextmenyn ska sparas undan i en kaka (cookie) så att användarens inställningar behålls även om sidan laddas om. Inställningarna behöver bara



läsas från kakan då contextmenyn laddas eller ett nytt fönster öppnas.
- Det ska vara möjligt att ha flera, av varandra oberoende, mikrobloggfönster aktiva samtidigt. Detta är en bra koll om man vill testa att allt fungerar. Går det att skicka meddelanden till servern som sedan läses upp i ett annat fönster?
- För att inte överbelasta servern ska det inte gå att ställa in tätare uppdateringsintervall än 10 sekunder.

För att lyckas med denna uppgift krävs en server som kan lagra meddelanden och svara på ajaxanrop. En sådan server finns tillgänglig på nedan angivna adresser. Det är en adress för läsning och en för skrivning.

Skrivning:
Adress:
`http://homepage.lnu.se/staff/tstjo/labbyserver/setMessage.php`

Metod:
`POST`

Parametrar:
`text`: Via denna parameter skickas meddelandets text in.
`username`: Via denna parameter skickas användarnamnet in.

Retur:
Om meddelandet sparas korrekt så innehåller responsen ett unikt id för meddelandet.

Läsning:
Adress:
`http://homepage.lnu.se/staff/tstjo/labbyserver/getMessage.php`

Metod:
`GET`

Parametrar:
`history`: Heltal som anger hur många poster som ska hämtas. 0 ger samtliga poster.

Retur:
XML-representation av det begärda antalet meddelanden. Parsas på valfritt sätt på klienten.
<id> - Unikt id för att identifiera meddelandet.
<text> - Meddelandetexten HTML-enkodad.
<author> - Författaren av meddelandet
<time> - Tiden då meddelandet sparades. Angivet i POSIX-tid. (millisekunder. sedan 1/1 1970)

Övrig funktionalitet
Förutom de krav som angetts i uppgiften måste du för att uppfylla betygskriteriet även implementera någonting förutom kraven.

Exempel på detta kan vara:

- Möjlighet att även flytta ordning på ikonerna på skrivbordet.
- Ytterligare applikationer på ditt skrivbord
- Möjlighet att styra hela applikationen med hjälp av tangentbordet, så som förflyttning av fönster etc.
- Möjlighet att ändra storlek på fönstret från samtliga fyra hörn och fyra sidor.
- Spara position och inställningar i en cookie på användarens dator så att denne får se samma skrivbord som senast den besökte sidan.
- Ritmöjligheter ovanpå bilden.
- Dockningslist till vilken man kan minimera sina fönster. (Jämför med Windows/OSX)
- Små ”widgets” som klockor etc. som man kan ”dekorera” sitt skrivbord med.
- Egen funktionalitet. Är du osäker om det du vill göra är tillräckligt så kontakta kursansvarig för bekräftelse.

Du ska göra en extrafunktion för betygssteg 4 och en för betygssteg 5.

Kontrollera kraven för betygsgrad 5 (bocka av)

- □ Samtliga krav för betyg 3 och 4 (förutom skrivbordsbildsbyte) är uppfyllda.
- □ Applikationen fungerar i såväl senaste versionen av Firefox som i senaste versionen av Internet Explorer och Chrome.
- □ RSS-läsaren har en meny med ovan beskriven funktionalitet.
- □ Memoryt har en meny med ovan beskriven funktionalitet.
- □ RSS-läsaren har en statusrad som talar om när den senast uppdaterades.
- □ Fönstren går att dra runt på skrivbordet.
- □ Det går inte att placera ett fönster helt eller delvis utanför skrivbordet.
- □ Det går att ändra storlek på de fönster som detta är intressant för.
- □ Det går kodmässigt enkelt att ändra storleken på skrivbordsytan, om denna inte täcker hela webbläsarfönstret.
- □ Storleken på ett fönster kan inte bli större än skrivbordsytan.
- □ Mikrobloggen fungerar enligt kraven för denna och har en contextmeny enligt beskrivningen.
- □ Min applikation uppfyller i övrigt de krav som är satta enligt beskrivningen ovan.
- □ Förutom funktionskraven ska applikationen vara visuellt tilltalande och ge ett proffsigt intryck.