Obligatorisk uppgift 3 – Media och lokal lagring

Syfte

Uppgiften syftar till att ge en introduktion till hur vi på ett integrerat sätt kan använda oss av olika former av ljud, bild och 2D-grafik i HTML5. Det som särskilt skall tränas är följande.

- Spela ljud- och videofiler,
- skapa och använda ritytor,
- använda Javascript för att styra delar av beteendet,
- introducera local storage som är en trevlig nyhet i HTML5.

Till skillnad från tidigare uppgifter får ni inga givna filer (bortsett från ljud- och videofiler). I stället finns ett antal krav uppställda som er sida ska hantera på ett korrekt sätt. Det som är viktigt är funktionaliteten, men försök gärna ge er webbsidan ett trevligt utseende med CSS.

Uppgift

Den sida ni ska skapa har tre olika medier som ska hanteras, video, ljud och 2D-grafik. Sidan ska också visas olika beroende på vilken typ av enhet som visar sidan. Slutligen ska den också kunna spara information lokalt på webbläsaren.

Video

På er sida ska ni använda <video>-taggen för att visa ett litet filmklipp. Filmklippet kommer i två versioner: en Mpeg4 (.mp4) och en Ogg Vorbis (.ogv). Tyvärr finns det normalt inte ett videoformat som hanteras av alla webbläsare. En av uppgifterna är därför att tillhandahålla flera filer i olika format till användaren, vars webbläsare därefter får spela upp den fil som hanteras. Detta skall dock göras inom samma video-tagg. Ni ska länka till video-filerna på vår server och ej använda filerna lokalt.

Filmklippen ska inte starta av att webbsidan laddas, men det ska gå att kontrollera uppspelningen av klippet genom webbläsarens kontroller, d.v.s. webbläsaren ska tillhandahålla de vanliga start/stop/spola/volym knapparna.

Man ska kunna byta ut klippet som spelas till ett annat klipp. Hur ni sköter detta är valfritt, men exempelvis skulle ni kunna ha två knappar, där om man trycker på den första knappen så spelas det första klippet, och trycker man på den andra knappen spelas det andra klippet. En annan möjlighet skulle kunna vara att använda länkar, eller att köra med en HTML <select> tagg. Klippet man valt ska börja spela direkt man valt klippet.

Ljud

Ni ska använda en <audio>-tagg för att spela ett ljudklipp på er sida,. Ljudklippet kommer i två versioner: en mp3 (.mp3) och en Ogg (.ogg). Precis som med video finns det inte heller för ljud ett format som fungerar för alla webbläsare, så ni ska använda båda klippen på er sida (inom samma audio-tagg). Tänk på att länka till ljud-filerna med hyperlänkar och ej använda filerna lokalt.

Ljudklippet ska starta av att webbsidan laddas, och det ska gå att kontrollera uppspelningen av

klippet genom webbläsarens kontroller.

Kontroller på sidan för video och ljud
Det ska finnas två knappar på er sida: 'Play All' och 'Stop All', där 'Play All' ska starta både aktuellt ljud- och videoklipp, och 'Stop All' ska pausa/stoppa dem båda.

2D-grafik
På er sida ska ni också tillhandahålla en rityta med en <canvas>-tagg. På ritytan ska man kunna rita ut cirklar. Cirklarnas placering avgörs genom att man klickar på en viss position med musen. Cirklarnas storlek och färg anges via separata fält under, eller vid sidan av ritytan. Man ska också kunna tömma ritytan med en särskild knapp.

I samband med att ritytan skapas eller rensas ska först en valfri text ritas ut som bakgrund med en färg som ligger nära bakgrundsfärgen, ungefär som en s.k. vattenstämpel.

Övriga krav:

- Man ska inte behöva trycka på en knapp för att fastställa färg och storlek.
- Om färg eller storlek innehåller felaktig information (ex. negativ storlek, skräpdata, eller att man utelämnar ett värde helt), ska detta hanteras genom att en standardfärg/standardstorlek används istället.

CSS Media
Använd CSS media-attributet för att skriva en stilmall som endast gäller för handhållna (handheld) enheter. I denna stilmall skulle ni kunna ge webbsidan en annan layout som bättre passar sig för de oftast lägre upplösningar som handhållna enheter har, det skulle även kunna vara en bra ide att byta ut ev. bakgrundsbilder mot versioner i lägre kvalitet i stilmallen då detta skulle spara bandbredd och möjligen inte ens märkas pga. lägre upplösning på den handhållna enhetens skärm.

I den här uppgiften har vi dock inte använt oss av så mycket layout eller bakgrundsbilder, så här räcker det att ni visar att ni kunnat skapa en sådan stilmall genom att ni ger webbsidan en annan bakgrundsfärg för handhållna enheter.

Ni ska fortfarande tillhandahålla en vanlig stilmall som antingen gäller oavsett media, eller som bara gäller för för en vanlig dator (screen).

Local Storage
Local storage (lokal lagring) är en ny teknik i HTML5 som kort innebär att information kan sparas i webbläsaren mellan besöken på en sida. Detta har man tidigare behövt göra via Cookies, men då med nackdelen att all lagrad data måste skickas till servern varje gång något ska laddas ner, vilket är ohållbart om man vill lagra någon större mängd data. Ett alternativ till detta har då varit att lagra data på serversidan, så slipper man skicka med datat vid varje laddning, men det är dels krångligt då den som besöker en sida måste identifiera sig, och dels kan det vara en säkerhets- och integritetsrisk av samma anledning, man kan inte heller använda data på serversidan lokalt via javascript.

I denna uppgift ska vi använda local storage till att spara våra ritade bilder; inte som fil, utan just i webbläsaren. Efter att en cirkel har ritats ska bilden på ritytan sparas i local storage. När ritytan sedan skapas för första gången när sidan laddas ska sedan denna bild hämtas från local storage och ritas ut på ritytan igen.

Om webbläsaren inte stödjer local storage ska ett meddelande som varnar om detta visas när man laddar sidan, webbsidan ska dock fortfarande gå att använda, men bilderna kommer naturligtvis inte kunna sparas så att de överlever en omladdning av sidan.

Övriga Krav

- Om ni vill skriva ut någon sorts spårtext i er kod ska `console.log()` användas, `document.write()` och `alert()` ska inte användas. Er Javascript-kod ska inte ge ut några varningar/fel, den ska vara väl strukturerad med indentering, och eventuella ’svåra’ avsnitt ska översiktligt kommenteras.
- Javascript ska skrivas i ”strict” läge, genom att det längst upp i filen står *”use strict;”* (med citattecknen). Detta läge tar bort bakåtkompabilitet mot tidigare versioner av Javascript, så en del saker som bedömts vara dåliga/olämpliga och därmed tagits bort i de senare versionerna av Javascript (och som därmed inte borde användas) kommer inte heller gå att använda.
  (http://ejohn.org/blog/ecmascript-5-strict-mode-json-and-more/)
- Ni ska inte använda jQuery eller liknande paket för Javascript. Ni kan däremot använda modernizr enligt kursboken för att hantera fel och undersöka stöd för media etc.
- Inlämningen måste fungera i Firefox (v10.0+).

Givna media-filer

- Samtliga filer finns för länkning på följande länk.
- http://www8.cs.umu.se/kursmaterial/html5/mediafiler
- Ljudklipp 'Independence Day' av DoKashiteru, är licenserat under Creative Commons BY NC, och det är nerladdat från http://ccmixter.org/. Videoklippen är från http://www.lucidmovement.com, och kan användas för icke komerciellt bruk och så länge de inte används på sidor med liknande format och användning.
- Om ni lägger er webbsida på en publik webbserver (som är åtkomlig via internet) så bör ni lägga till en text likt ovanstående paragraf för att hålla er inom licensens ramar, eftersom licensen kräver att den som använder klippet:
- Inte använder det kommersiellt.
- Ger 'attribution' (berättar vem som är skaparen).

Inlämning

Döp er webbsida till `index.html`, övriga filer (stilmallar och Javascript) ger ni lämpliga namn (undvik dock tecken som kan bli ett problem med teckenkodningar, exempelvis å, ä och ö).

Filerna lämnas in via kurssidan på Cambro. Undvik helst att packa filerna utan skicka med alla filerna som bilagor till inlämningen. All HTML/CSS/Javascript ni skriver ska vara korrekt och validerande.

Läsanvisningar:

Läs Dive Into HTML avsnittet om Canvas innan ni börjar, speciellt följande sida.

http://diveintohtml5.net/canvas.html#halma

Det är också bra att läsa in sig på audio/video taggen:

http://diveintohtml5.net/video.html#markup

Och det är starkt rekommenderat att göra som kurslitteraturen gör för att upptäcka om något stödjs i webbläsaren eller inte:

http://diveintohtml5.info/detect.html

Tips

Byta videokälla

För att kunna välja vilket av filmklippen som ska spelas så måste ni använda er av Javascript, och en lämplig bindning i HTML-koden. Använder ni knappar eller länkar kan onclick-bindningen vara lämplig, och använder ni `<select>` så kan i stället onchange-bindningen vara lämplig.

html, som länk:

```
<a href=”#” onclick="spelaKlipp('klipp1')">Spela klipp 1</a>
```

html, som knapp:

```
<button onclick="spelaKlipp('klipp1')">Spela klipp 1</button>
```

html, som select:

```
<select id="movieSelector" onchange="spelaKlipp(this.value)">
<option value="klipp1">Klipp 1</option>
<option value="klipp2">Klipp 2</option>
</select>
```

Som ni lägger märke till så skickar vi in argumentet this.value till `spelaKlipp` funktionen i select-taggen. Detta kommer från att `this` är själva `select`-elementet (samma som ni får om ni ex skulle kört `getElementById(movieSelector"))`, och `value` är attributet som innehåller det valda värdet ("klipp1"eller "klipp2"i det här fallet).

Bytet av den faktiska video-källan kan ske på lite olika sätt, antingen har ni ett id-attribut på de två source-taggarna (som ni lär behöva använda för att stödja flera filformat), och sedan bara ändra src-attributet på dem båda. Ni kan också ta bort alla barn för video-taggen, och sätta src-attributet på video-taggen direkt, detta skulle dock kräva att ni kör `canPlayType`-funktionen på video-taggen för att kolla vilken av de två givna videoformaten för de givna klippen som webbläsaren kan spela.

html, före byte:

```
<video>
<source id="videoSource1" src="video1.mp4" type="video/mp4">
<source id="videoSource2" src="video1.ogv" type="video/ogg">
</video>
```

html, efter byte:

```
<video>
<source id="videoSource1" src="video2.mp4" type="video/mp4">
<source id="videoSource2" src="video2.ogv" type="video/ogg">
</video>
```

HTML, alternativ lösning (med canPlayType-funktionen) efter byte (där src och type sätts efter det format som webbläsaren kan spela, i det här fallet mpeg4):

```
<video src="video2.mp4" type="video/mp4">
</video>
```

OBS: Om ni byter videoklipp genom att ändra på de två source-taggarnas src-attribut så måste ni anropa .load() på ert videoelement för att videotaggen ska uppdateras och börja spela de nya klippen. Notera att i Chrome behövs ytterligare ett knep som ni kan läsa om i Wikin.

Ritytan

Det kan vara värdefullt för att få fram hur man översätter muskordinater (x,y relativt webbsidan) till ritytekordinater (x,y relativt ritytan). Ni kan förslagsvis göra en funktion som sätter upp en EventListener för musklick på er rityta (läs avsnittet om canvas i kurslitteraturen som länkas ovan), denna funktion kan ni sedan anropa från onload=”” i body-taggen, för att säkerställa att DOM-trädet är genererat innan ni försöker skapa en EventListener. En annan möjlighet skulle kunna vara att använda attributet 'defer' på er script-tagg, för att säkerställa att inget i scriptet kommer köras innan hela sidan laddats:

http://www.w3schools.com/TAGS/att_script_defer.asp

Utöka gärna webbsidans funktionalitet, men se till att ni i så fall fortfarande uppfyller de krav som det här dokumentet har listat. Färgen för ritytan kan ni antingen läsa in som ett hex-värde (ex. #FF0000) genom ett textfält, eller som tre separata värden (ex. 255,0,0) genom tre textfält. Har ni något annat alternativ för att välja färg så får ni såklart göra så istället, men tänk på att color-picker i HTML5 inte stödjs av så många webbläsare ännu (bara Opera?), så ni kan inte använda den för den här labben.

Tänk på att en felaktig färg (ex. #GG0000 om man skulle köra hex) ska ge en standardfärg, så välj ett sätt som du kan validera mot (ex. kan tre separata värden mellan 0-255 valideras med `parseInt()` och `isNaN()` funktionerna, medan en sträng #FF0000 kan kräva lite mer avancerade uttryck för validering).

Spara bild i local storage

Local storage används som ett associativt fält eller lista (ett fält eller lista med namngivna fält). Exempelvis

```
localStorage['myValue']=5;
```

För att ha något värde att lägga i fältet krävs att vi hämtar bilden i ett format som vi kan spara undan. Detta kan göras med `toDataURL()` på ritfältet. För att återskapa bilden måste detta värde göras om till en bild som sedan kan ritas på ritytan igen.

```
var img = new Image();
img.onload = function() {
        var ctx = canvas.getContext("2d");
        ctx.drawImage(img,0,0);
}
img.src = savedValue;
```

CSS-Media

För att testa att er stilmall fungerar för handhållna enheter kan det vara en bra ide att använda webbläsarutökningen (eng. Extension) ”Web Developer” som finns för både Google Chrome och Mozilla Firefox. Genom denna extension så får ni tillgång till en rad olika verktyg som kan vara behjälpligt vid webbutveckling, däribland ”CSS – Display CSS By Media Type” där ni kan välja att webbläsaren ska ladda in de CSS filer som en handhållen enhet skulle ha laddat ner.

Google Chrome:

https://chrome.google.com/webstore/detail/bfbameneiokkgbdmiekhjnmfkcnldhhm

Mozilla Firefox:

https://addons.mozilla.org/en-US/firefox/addon/web-developer/

Moderna smartphones låtsas oftast att de är riktiga datorer och kommer därför troligtvis inte att använda CSS filer för handhållna enheter utan istället använda samma CSS filer som er dator använder. Därmed är det inte rekommenderat att använda en smartphone för att testa att sidan hanterar CSS för handhållna enheter.

Outline på arbetsflöde

Om ni finner det svårt att komma i gång med uppgiften kan detta vara till hjälp.

1. Skapa grundstrukturen för webbsidan (html, head, body).
2. Lägg till en video på webbsidan med kontroller, lägg till två källor för ett av klippen (mp4 och ogv).
3. Lägg till en audio-tagg på webbsidan med kontroller, lägg till två källor (mp3 och ogg) för det givna klippet.
4. Lägg till en canvas-tagg.
5. Skapa en Javascript-fil, länka in i head med en script-tagg. Sätt att den körs först när webbsidan laddat klart, eller sätt att body kör en funktion i onload-bindnigen.
6. I Javascriptet (antingen utanför en funktion, om scriptet körs först när sidan har laddat, eller i setup-funktionen som ni anropat från onload i body) lägg till en EventListener enligt beskrivningen i Dive In-to HTML5 boken, dvs. `ertCanvasElement.addEventListener("click", erCanvasClickFunktion, false);`
7. Skapa en funktion som hanterar era klickningar på ritytant (ovan kallad `erCanvasClickFunktion`), denna funktion kommer anropas med ett MouseEvent som argument. Funktionen `getCursorPosition` som definieras i Dive Into HTML5 boken är en lämplig grund att bygga från, bortsett från de sista raderna som rör "cell". Låt funktionen skriva ut x och y koordinaterna med `alert()` (Denna alert() använder ni såklart bara när ni utvecklar sidan, er inlämning ska inte köra alert varje gång man klickar på ritytan).
8. Nu har ni då en rätt bra grund att bygga vidare på, eftersom ni har: En audio och en

video som kan spela upp de givna testfilerna och som kan kontrolleras från webbläsaren. En rityta, som när man klickar på den skriver ut vart (i förhållande till ritytan) man klickade.