Inlämningsuppgift 3 (20.12.02)(U/G)

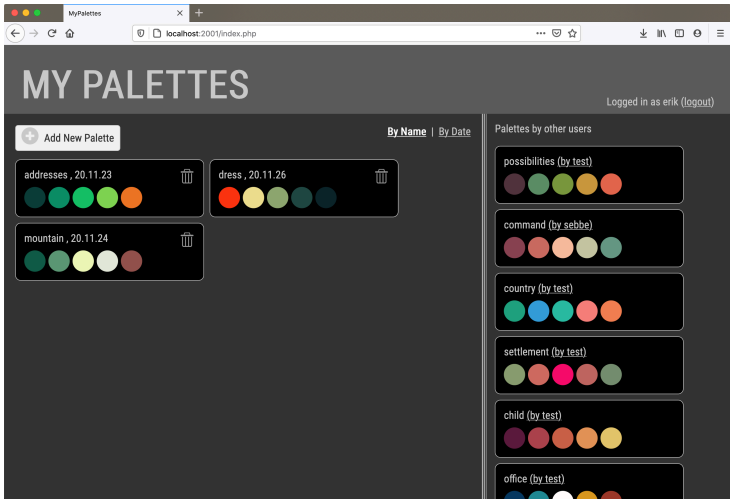
Webb-applikationen My Palettes

Ni ska koda en enklare webbapp som låter en användare skapa och spara färgpaletter.

Uppgiften lämnas in:

- På Canvas som en zip-fil med alla nödvändiga filer på plats.
- Som en URL (appen ska alltså finnas tillgänglig på webben). URL:et lämnas gärna som kommentar när ni lämnar in på Canvas.

Beskrivning av appen



Instruktioner (se videon för mer info):

A. I uppgiften ingår det att sätta sig in i och diskutera en lösning till uppgiften (som ni får när deadline för inlämning har varit). Ni ska också jämföra er lösning med den som ni får i efterhand. Mer specifika instruktioner kommer i efterhand.

B. Ni ska koda en app som den som visas på videon. I det ingår både front-end och back-end (bla. API och databas).

C. Appen ska i princip se ut som den på videon: Liknande färger (inte de i paletterna, såklart), liknande storlekar (margins, width, etc), liknande ikoner (finns att hitta på nätet). Använd Roboto Condensed som typsnitt. Det kan bli kompletteringar om sidan inte ser ut som den ska.
   A. Sidan behöver endast se bra ut för bredder >= 950px.

D. På höger sida i viewern visas alla paletter som har skapats av alla (andra) användare. Om man inte är inloggad så visas alla paletter. Om man är inloggad så visas alla paletter förutom de som skapades av den inloggade användaren. Notera att namnet på skaparen syns men inte datumet när paletten skapades.

E. Om man på höger sida klickar på ett användarnamn så visas där endast paletterna som den användaren har skapat. Då visas också texten "See All Palettes" som tar bort filtret. Detta gäller oavsett om man är inloggad eller inte.

F. Användaren ska kunna logga in. Inloggningen ska vara permanent under sessionen (den ska exvis vara kvar efter page-reload).
   A. Om användaren skriver i fel användarnamn eller lösenord så ska båda dessa få röd bakgrundsfärg. Så fort användaren börjar skriva igen (så fort någon av fälten får fokus… se focus-eventet) så ska bakgrundsfärgen i båda fälten bli genomskinlig igen.

G. Användaren ska kunna logga ut genom att klicka på logout-texten. Hemsidan laddas om då.

H. Appen använder två publika API:n.
   A. Colorminds API (http://colormind.io/api-access/) är en API som levererar arrays av fem färger som passar ihop och tillsammans utgör en palette.

   B. Nouns API är en superenkel API som bara accepterar en GET-request (https://erikpineiro.se/dbp/nouns/api.php) och som levererar ett random substantiv på engelska. Svaret är ett JSON-formatterat objekt: {data: "randomWord"}
   C. Ni ska utveckla en API och en databas för att spara på paletter och användare. Varje grupp utvecklar sin egen databas och API.
      A. Er databas ska ha minst tre olika användare.
      B. Ni behöver inte koda skapa-nytt-konto-funktionaliteten (att registrera sig i appen) eller radera-konto funktionaliteten.

I. När den inloggade användaren klickar på knappen "Add New Palette" så skapas det en ny palette som innehåller fem färger. Färgerna i paletten hämtas från Colorminds API:n. Namnet på paletten hämtas från Noun-API:n. Notera att två paletter kan ha samma namn men inte samma id.
   A. Den nya paletten syns tydligt (se videon) under 3 sekunder.
   B. Paletten tar plats i listan så att den aktuella sorteringen behålls (det enklaste i detta fall är att köra en ny sortering direkt efter det att paletten har adderats).

J. Användaren kan ordna sina sparade paletter enligt namn eller datum. Alltid ascending vad gäller namn (a,b,c, …) och alltid descending vad gäller datum (senast skapad först). I filen useThisIfYouWant.js finns det funktioner som hjälper er med datumet. Använd dem gärna.
   A. Notera att appen visar datumen som "20.11.28" men tid-informationen ska inkluderas när man ordnar paletterna byDate. Med andra ord: i databasen måste ni spara både datum och tid. Funktionen getNowAsISO() returnerar en sträng som innehåller datum och tid. Den ska ni spara i databasen. Om man tittar noggrant så inser man att om man sorterar dessa texter i bokstavsordning så motsvarar det "ascending" sortering av datumen.
   B. När användaren loggar in så är default sorteringen "byName".

K. Användaren kan ta bort en palette från sina sparade genom att klicka på papperskorgen. Inga bekräfta-frågor ställs till användaren. Paletten tas bort direkt från listan med sparade paletter utan att sidan laddas om.

L. Databasen måste vara uppdelad i två delar (två nycklar):
   A. En för paletter, som måste ha exakt dessa nycklar: id, name, creatorID, date, colors
   B. En för användare, som måste ha exakt dessa nycklar: id, name, password.

M. Paletterna ska representeras i koden av JS-klasser. Detta innebär att ni inte kan skapa HTML-koden för paletterna på back-end. Datan måste komma till JS och där ska paletternas HTML-kod skapas, se nedan. Det är lite overkill men ni måste använda dessa tre klasser:
   A. PaletteBase. Nycklar: **id**, **name**, **colors**, **html**. html-nyckeln pekar på ett HTML-element (eller jQuery-objekt) som innehåller de element som är gemensamma för PaletteSave och PaletteOthers.
   B. PaletteSaved extends PaletteBase. Denna klass används för paletterna på vänster sida. Extra nyckel: **date**. HTML-Elementet (eller jQuery-objekt) som html pekar på och som skapades i superklassen ska kompletteras.
   C. PaletteOthers extends PaletteBase. Denna klass används för paletterna på vänster sida. Extra nyckel: **creator**. HTML-Elementet (eller jQuery-objekt) som html pekar på och som skapades i superklassen ska kompletteras.

N. Det är tillåtet, men inte ett krav, att använda jQuery.

TIPS

A. Vi rekommenderar att ni, innan ni börjar koda, diskuterar hur ni kommer att strukturera filerna och vilken "logik" som ska placeras i dem. Med andra ord: Var olika "beslut" i koden tas. Aspekter som ska tas upp i er interna diskussion:
   A. Variabelnamn: Hur tänker ni kring namnen? Vi förväntar oss genomtänkta namn för alla variabler och funktionsnamn.
   B. Struktur vad gäller variabler. Vi vill att ni specifikt diskuterar:
      A. Vilka globala variabler, om några, ska finnas i koden (JS och PHP). Hur ska ni strukturera dessa så att de är lätta att komma åt och svåra att förstöra av misstag.
      B. Vilka variabler, om några, ska användas för att skicka/ta emot data mellan back- och front-end.
      C. Hur ska datan skickas till klass-constructors när man skapar en instans?
   C. Vilka funktioner ska ni implementera, i vilka filer ska de finnas, vilka parametrar tar de emot och vad returnerar dem (om något). Skissa på papper!

   D. HTML-struktur: Vilka element behövs. Gå ner i detalj, försök att ha klart för er alla element som kommer att behövas. Använd skärmdumparna för att inte glömma något.
   E. All tid ni ägnar åt detta får ni tillbaka med råge när ni sedan kodar.

B. Troligtvis kommer ni att behöva "informera" JS-koden om vem det är som är inloggad. Detta kan göras på olika sätt. Läs detta inlägg: https://stackoverflow.com/a/23740549. Alternativ 1 har vi jobbat en del med i kursen. Alternativ 2 kan fungera men vi rekommenderar att undvika att ha hidden-fields i HTML-koden. Titta på alternativ #3: idén är att echo:a ett script-element som initialiserar en global JS-variabel som sedan är tillgänglig för alla JS-scripts. Lite beroende på hur ni lägger upp logiken i koden kan det vara ett väldigt smart att "informera" JS om vem som är inloggad.

C. Vi rekommenderar att ni lämnar CSS till slutet. Koda all logik först så appen fungerar som den ska. För att underlätta formateringen skulle vi rekommendera att skapa egna HTML-element av allt som användaren kan interagera med och allt som kan ändras från JS.

Vidare övningar

Följande saker är inte krav för G men är mycket bra övning (och imponerar på lärarna). Gör dem, eller någon av dem, om ni hinner:

A. Använd cookies för att säkerställa att inloggningen kvarstår även efter avslutad session.
B. Använd endast vanilla JS.
C. Koda funktionaliteten för att skapa ett nytt användarkonto (registrering).
D. Designa och koda ett elegant sätt att ge feedback till användaren om kommunikationen med databasen.
E. Bekräftelse innan en palette tas bort. Borttagning sker i två steg:
   A. Steg1: Första klicket på papperskorgen aktiverar den (byter färg till vit eller något i den stilen).
   B. Steg2: Andra klicket tar bort paletten. Inga andra bekräfta-frågor ställs till användaren.
   C. Om användaren klickar en gång (steg1) och aktiverar papperskorgen men inte klickar igen (inget steg2 alltså) så ska aktiveringen försvinna efter 3 sekunder. Alltså färgen på papperskorgen blir den vanliga gråa och processen är tillbaka till steg 1.

Andra krav

A. Korrekt indentering av all kod.