WEBBUTVECKLING – KLIENTSIDAN

Distanskurs på grundnivå 7,5 ECTS

Alexander Krasse

Institutionen för data- och systemvetenskap (DSV)

Webbutveckling – Klientsidan
Gesällprov
Grunder i HTML, CSS och JavaScript Vårterminen 2018

SAMMANFATTNING

Detta projekt är ett gesällprov i kursen Webbutveckling - Klientsidan, och är anpassat främst för att köras i Chrome.

Syftet med arbetet är att göra ett gesällprov. Kriterierna för gesällprovet är att det skall baseras på de tekniker som kursen bygger på och ska vara bra både till form och funktion.

Programmet har mestadels skrivits i Microsoft Visual Studio Community 2017. Koden har därefter flyttats ut till rena textfiler med hjälp av Notepad++ v7.5.7, för att få ett mer rent program för att begränsa sig till de tekniker som ingått i kursen. För att kunna visa kod till exemplen på ett mer visuellt tilltalande sätt än att bara använda `<code>` utan att behöva skriva allt för mycket HTML-kod har ett script gjorts. Detta script lägger till CSS för att efterlikna hur koden ser ut i ett utvecklingsverktyg. Scriptet bygger mestadels på RegEx för att dela upp varje rad i delar med olika färg.

För att lösa dessa RegEx-uttryck har en RegEx-tester används som går att hitta på https://regex101.com/ All HTML-kod och CSS-kod har validerats med w3s validator som finns på https://validator.w3.org/

Resultatet är en hemsida som visar kursens uppgifter liknande ett dokument i A4-format med interaktiv funktion, och beskriver hur sidan är uppbyggd.

INNEHÅLLSFÖRTECKNING

1. INTRODUKTION

Detta projekt är ett gesällprov i kursen Webbutveckling - Klientsidan, och är anpassat främst för att köras i Chrome.

1.1 Bakgrund

IB908C Webbutveckling - Klientsidan 7,5 hp är en kurs på Institutionen för data- och systemvetenskap (DSV) Stockholms Universitet. Kursen är en grundkurs som behandlar Struktur med HTML5, Design med CSS 3 och Dynamik med JavaScript. Förutom övningsuppgifter ingår ett gesällprov där tekniker från kursen tillämpas.

1.2 Syfte

Syftet med arbetet är att göra ett gesällprov. Kriterierna för gesällprovet är:

- Ska baseras på de tekniker som kursen bygger på men får givetvis även innehålla tekniker från andra områden
- Ska vara bra både till form och funktion
- Ska inte vara en av de tidigare uppgifterna på kursen men får gärna vara en utökning av en eller flera av dessa.

1.3 Mål

Målet med projektet är att skapa en hemsida som med HTML, CSS, och JavaScript efterliknar ett PDF-dokument med interaktiv funktion för att beskriva uppgifterna i kursen och sig självt.

1.4 Metod

Programmet har mestadels skrivits i Microsoft Visual Studio Community 2017. Koden har därefter flyttats ut till rena textfiler med hjälp av Notepad++ v7.5.7, för att få ett mer rent program för att begränsa sig till de tekniker som ingått i kursen. För att kunna visa kod till exemplen på ett mer visuellt tilltalande sätt än att bara använda <code> utan att behöva skriva allt för mycket HTML-kod har ett script gjorts. Detta script lägger till CSS för att efterlikna hur koden ser ut i ett utvecklingsverktyg. Scriptet bygger mestadels på RegEx för att dela upp varje rad i delar beroende på vilken färg som skall användas. För att lösa dessa RegEx-uttryck har en RegEx-tester används som går att hitta på regex101.com. För att lösa uppgifter har främst MacDonald, M (2013). *HTML5 The Missing Manual*, 2nd Edition.pdf, Sebastopol: O'Reilly Media, Inc. www.w3.org, developer.mozilla.org, w3schools.com använts.

1.5 Begränsningar

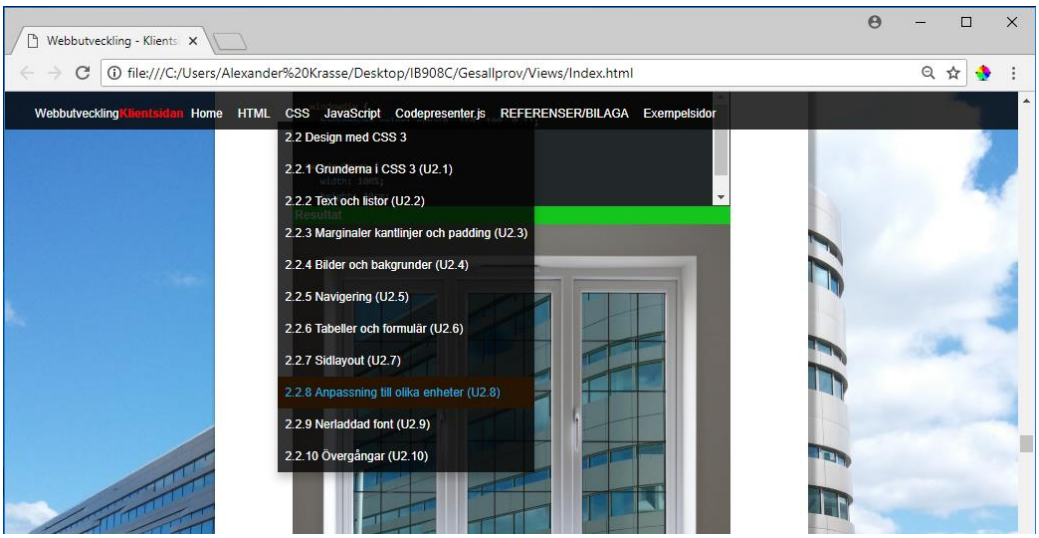
Projektet har begränsats till HTML5, CSS3, JavaScript i den slutgiltiga versionen. Målet har varit att i så stor utsträckning som möjligt undvika att använda externa bibliotek. Projektet använder dock i viss mån av jQuery. Till en av övningsuppgifterna har jQuery UI använts, och till övningsuppgifter som kräver validering av <form> har jQuery validering använts. För att presentera scheman (se Bilagor) har ett script använts från www.draw.io.

2. GESÄLLPROV

Målet med detta gesällprovet är att göra en hemsida som presenterar uppgifterna i kursen som ett interaktivt dokument. Hemsidan är uppbyggd av en meny och divar som representerar A4-papper. Se Bilaga 1 för schema över fil-relation.

```
<div class="paper-A4" id="pageNumber">
     <div class="header-A4"></div>
     <div class="writing-area">
          <div class="left-margin"></div>
          <div class="article">
               <!--Content-->
          </div>
          <div class="right-margin"></div>
     </div>
     <div class="footer-A4"></div>
</div>
```

Via menyn kan man navigera till de olika uppgifterna.



Figur 1. Dropdownmeny med länk till uppgifterna med uppgiftsnummer inom parantes

2.1 Körning av programmet

Först laddas all HTML-kod. När all HTML inklusive bilder har lästs in anropas scripten. Codepresenter.js letar upp alla `<div>`ar som har klassen ”codepresenter”. Med hjälp av CSS byggs en behållare med olika sektioner vars utseende beror på vad för klasser som är satta på de `<div>`ar som finns i `<div class=”codepresenter”>`. Klassnamnen ersätts med en ny klass (detta för att ingen CSS skall aktiveras innan innehållet har gjorts om. Innehållet i divarna bevaras varefter funktioner anropas som gör om innehållet till text (ifall innehållet skulle vara inskrivet som taggar ersätts dessa med entity-tecken). Sedan körs metoder som letar upp alla divar i HTML-en som har ett klassnamn som motsvarar om det är HTML, CSS, JavaScript eller ett resultat som skall presenteras. Innehållet för varje `<div>` gås igenom per rad för att kontrollera vilken färg varje ord skall ha. Denna indelning görs med hjälp av RegEx. För varje rad beräknas även hur stort indraget skall bli.

2.2 SCRIPT

2.2.1 Codepresenter.js

Script för att lättare kunna presentera kod på ett mer tilltalande sätt utan att behöva skriva allt för mycket kod. Scriptet gör om inmatad text för att efterlikna hur den ser ut i en utvecklingsmiljö. Beroende på hur CSS-filen ser ut kan man efterlikna olika programvaror. I detta exempel har färger använts för att efterlikna Microsoft Visual Studio 2017 Dark Theme. Nedan är ett kort exempel hur resultatet ser ut när det används för att beskriva hur man skapar ett `<table>`.

Figur 2. Exempel på hur scriptet ser ut när det använts.

2.2.2 Implementation

För att kunna använda scriptet behövs förutom själva scriptet och tillhörande CSS även att jQuery implementeras. Och scriptet måste köras efter att hela dokumentet har lästs in.

```
<link href="Styles/Codepresenter.css" rel="stylesheet" />
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script src="Scripts/Codepresenter.js"></script>
```

2.2.3 Användning

För att använda klass-biblioteket skapar man en `<div>` med classnamnet codepresenter. I denna kan man lägga divar som skall innehålla den kod som skall presenteras. Generellt kan man klistra in kod direkt från källan utan modifiering bortsett från några undantag av html-element (generellt element som inte ligger i `<body>`).

De fyra olika alternativen är:

** ** Här läggs html-kod. Man kan klistra in koden i grundform med taggar och allt, dock anropas scriptet först efter att hela DOM-en och bilder har laddats så vissa typer av element kommer ej att validera som till exempel <body>, vilket kräver att man använder entity istället för taggar på dessa så att de inte tolkas som htmlelement när sidan laddas.

** ** Här läggs text som representerar CSS-kod.

** ** Här läggs text som representerar JavaScript-kod.

** ** Här läggs koden i grundform och behandlas ej för att behålla funktion.

Ordningen de fyra ovanstående alternativen står i saknar betydelse utan de ritas upp i den ordning som de är inskrivna. Man kan välja att ha flera divar av samma typ eller välja att inte ha med någon alls. Höjden på varje behållare är satt till 7 rader med text plus padding. Om Antalet rader överskrider detta appliceras en scroll på behållaren.

för att få ovanstående resultat skriver man:

```
<div class="codepresenter">
     <div class="codepresenter-html">
          <div class="square"></div>
     </div>
</div>
```

Efter att scriptet har körts kommer ovanstående kod att skrivas om till

```
<div class="codepresenter-container">
    <div class="codepresenter-menu">
        <div class="HTML-example">HTML</div>
    </div>
    <div class="codepresenter-source">
        <div class="codepresenter-HTML">
            <code>
                <span class="HtmlBracket">&lt;</span>
                <span class="HtmlElementColor">div</span>
                <span class="HtmlAttributeColor"> class</span>
                <span class="HtmlTextColor">="square"</span>
                <span class="HtmlBracket">&gt;</span>
                <span class="HtmlBracket">&lt;</span>
                <span class="HtmlElementColor">div</span>
                <span class="HtmlBracket">&gt;</span>
            </code>
        </div>
    </div>
</div>
```

För att få alla typer av behållare skriver man

```
<div class="codepresenter">
    <div class="codepresenter-html">
        <!--HTML-kod-->
    </div>
    <div class="codepresenter-css">
        /*CSS-kod*/
    </div>
    <div class="codepresenter-js">
        //JS-kod
    </div>
</div>
```

2.3.4 Funktioner

I Bilaga 2 finns schema för hur funktionerna integrerar med varandra.

`window.addEventListener('load', function())`
Körs när all html-kod har laddats in inklusive bilder. Anropar createCodepresenterContainer() och parseCode(string), som ändrar utseende på text så att denna efterliknar en IDE.

`function createCodepresenterContainer()`
När all html-kod har lästs in anropas createCodepresenterContainer som letar upp alla ar med classen codepresenter och bygger behållarna med utseende efter vad för class som angivits. Text och färg på bannern sätts beroende på om classen anger om det är HTML-, CSS-, JavaScript, eller ett resultat. Lägger till denna HTMLkod till DOMen.

`function parseCode(string)`
parseCode letar igenom all html-kod efter element med det classnamn som skickades med. Alla dessa element lagras i variabeln node som är en `HTMLCollectionOf<Elements>`. För varje element i node splittas inre HTML per rad som lagras i en array som skickas till den funktion som matchar classnamnet (d.v.s html-, css- eller js-kod) varefter DOMen uppdateras.

`function IdeAppearanceHTML(string[])`
Får in en string array där hela arrayen motsvarar innehållet i en div som har classnamnet ”codepresenter-HTML” och varje element i arrayen motsvarar en rad av divens innehåll. Först skapas ett tomt <code>-element och arrayen tvättas på ogiltiga tecken. Sedan loopas hela arrayen igenom. Mellan varje rad läggs en radbrytning till. För varje rad beräknas hur stort indraget skall vara. Indraget läggs till <code>-elementet som ett icke radbrytande mellanrum. Om raden är en kommentar så skapas ett span med grön text, annars delas raden upp med ett regexuttryck som delar texten per html-tagg. Taggarna tas sedan bort och läggs till <code>-elementet som ett span med mörkgrå textfärg. Innehållet i taggen delas sedan upp så att attribut och värde sätts i olika span. Text som ligger mellan taggar läggs till som vanlig text. Till sist returneras <code>-elementet som nu innehåller hela raden fast där all text ligger inuti olika spans beroende på vilket färg som skall sättas.

`function calculateIndent(string)`

Tar emot en string som motsvarar en kodrad i html och kontrollerar vad denna innehåller för tecken för att beräkna hur stort indraget på raden skall vara.

`function IdeAppearanceCSS(string[])`

Får in en string array där hela arrayen motsvarar innehållet i en div som har classnamnet ”codepresenter-CSS” och varje element i arrayen motsvarar en rad av divens innehåll. Först skapas ett tomt <code>-element och arrayen tvättas på ogiltiga tecken. Sedan loopas hela arrayen igenom. Mellan varje rad läggs en radbrytning till. För varje rad beräknas hur stort indraget skall vara. Indraget läggs till <code>-elementet som ett icke radbrytande mellanrum. Om raden är en kommentar kontrolleras om raden innehåller ett CSS nyckelord. Resten av raden kontrolleras ifall färgen för varje del skall vara property, string eller text. Annars kontrolleras om raden innehåller selector eller property och text. Om raden har ett sluttecken adderas en radbrytning. Till slut returneras hela <code>-elementet med alla spans där texten ligger.

`function IdeAppearanceJavaScript(string[])`

Får in en string array där hela arrayen motsvarar innehållet i en div som har classnamnet ”codepresenter-JavaScript” och varje element i arrayen motsvarar en rad av divens innehåll. Först skapas ett tomt `<code>`-element och arrayen tvättas på ogiltiga tecken. Sedan loopas hela arrayen igenom. Mellan varje rad läggs en radbrytning till. Först delas raden upp i kod-respektive kommentarsdel. Om det finns en koddel delas denna upp i string (text mellan '' eller ""), RegEx uttryck och övrigt. Programmet tolkar RegEx inuti strings som strings och string inuti RegEx som RegEx. Om Raden innehåller RegEx kontrolleras om det finns tillhörande flaggor (dvs g, m eller i) då dessa skall sättas i en egen färg. Den del som inte klassas under någon av de föregående delas upp för varje alphanumerical eftersom "function" är reserverat men inte "function1". Dessa ord kontrolleras mot en array som innehåller reserverade ord. Sedan kontrolleras fristående siffror som även de skall sättas i en egen färg. Varje del lagras som ett element i en array som sedan loopas igenom och hamnar i ett span motsvarande den textfärg som skall sättas. Alla `<span>` läggs till `<code>`-elementet som returneras.

`function isNumber(string)`

Tar in en string och kontrollerar om denna är en siffra. Anropas vid färgsättning av JavaScript som skiljer på text och siffror. Returnerar -1, 0, 1 beroende på om indraget skall minskas, ökas, eller vara oförändrat.

`function replaceAll(string, string, string)`

Ersätter alla förekomster av en sträng i en text med en definierad sträng. Returnerar den nya texten där alla ord av en viss typ har bytts ut mot ett annat ord.

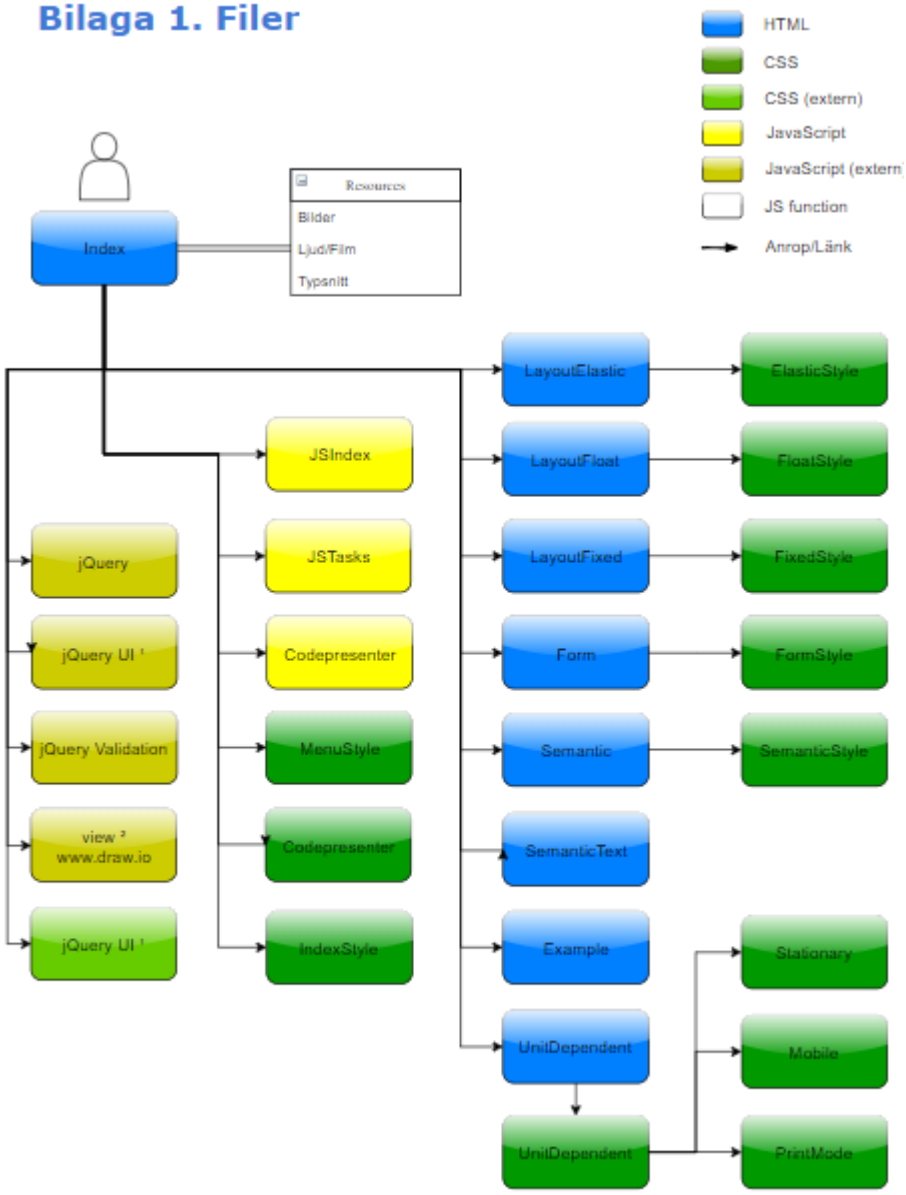
`function cleanArray(string[])`

Tar in en string array och tvättar denna på alla specialtecken. Det vill säga tecken som ligger utanför allt mellan space och tilde och åäö. Anledningen av detta är att förhindra oönskade radbrytningen och andra specialtecken som sätts av Visual Studio vid automatisk indragning. Returnerar string arrayen utan specialtecken.

```
function cleanArray(string[])
```
Tar in en int som avgör hur många gånger ett span med en fördefinierad leftpadding skall läggas till för att motsvara ett indrag. Returnerar en string som motsvarar html-kod för span.

```
function nbspTab(int)
```
Tar in en int som avgör hur många gånger tre entity för ett icke radbrytande mellanslag skall läggas till för att motsvara ett indrag. Returnerar en string som motsvarar tre mellanslag gånger valt antal.

Bilaga 1. Filer



¹ Används till uppgift 2.4 Utvidgbara gränssnitt

² Används för att rita UML/Flowchart

Bilaga 2.
Codepresenter.js

Översikt över funktioner och dess relation