Projektuppgift

DT084G - Introduktion till JavaScript

Projekt - JavaScript

Albin Rönnkvist

Mittuniversitetet
MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

MITTUNIVERSITETET
Avdelningen för informationssystem och -teknologi

Författare: Albin Rönnkvist, alrn1700@student.miun.se
Utbildningsprogram: Webbutveckling, 120 hp
Huvudområde: Datateknik
Termin, år: HT, 2018

Sammanfattning

I detta projekt använder jag mig av JavaScript för att dynamiskt ändra och lägga till innehåll på en webbplats. Jag kommer att arbeta mot Arbetsförmedlingens API och använda mig av AJAX-anrop för att hämta data och på så sätt uppdatera webbplatsen dynamiskt.

Innehållsförteckning

1 Introduktion

1.1 Bakgrund och problemmotivering

I detta projekt ska jag använda mig av Arbetsförmedlingens API/webbtjänst för att dynamiskt uppdatera innehåll på en färdigstrukturerad webbplats. Kommunikationen med webbtjänsten kommer ske med JavaScript där jag använder mig av AJAX-anrop för att hämta data från API:t. JavaScript-filen kommer kopplas ihop med ett HTML-dokument som tillsammans ska bilda ett fungerande gränssnitt.

1.2 Avgränsningar

Jag kommer endast gå igenom JavaScript och hur jag konsumerar webbtjänster samt hur jag skriver ut informationen på webbplatsen. Jag kommer inte gå in på CSS och inte heller så mycket på HTML eftersom webbplatsens struktur och utseende redan är förskapad av kursens lärare.

1.3 Detaljerad problemformulering

Utformning av kod.

Först behöver jag ta reda på vilken data jag behöver hämta, alltså vilken information som ska skrivas ut. När jag vet det så måste jag ta reda vilka element som kräver implementation i JavaScript, till exempel i vilka element som information ska skrivas ut och i vilka element som funktioner ska finnas. När jag vet det så måste jag även sammanställa vilken typ av funktionalitet som behövs för de olika elementen.

Skapande av kod

I detta steg går jag igenom sammanställningen från förra steg och implementerar funktionalitet för ett element åt gången. Här testar jag även varje element så att det fungerar som det ska innan jag går vidare till nästa.

Testning av webbplats

När all funktionalitet är implementerad så testar jag webbplatsen för att se om all funktionalitet fungerar som det ska och om jag eventuellt skulle kunna göra några förbättringar för att förenkla användandet.

Jag kommer även testa webbplatsen i olika webbläsare och på olika enheter för att försäkra mig om att den implementerade funktionaliteten fungerar som den ska oavsett användares förutsättningar. Sedan kommer jag validera den genererade HTML-koden för att se eventuella valideringsfel eller varningar för att se om koden behöver ändras eller förbättras.

2 Teori

JavaScript – JavaScript är ett kraftfullt skriptspråk som initialt skapades för att göra webbsidor mer ”levande”. Det används till exempel ofta till att skapa dynamiskt innehåll och kontrollera vad som ska ske vid olika event som ett musklick eller när besökaren scrollar. Skriptspråket används även i andra sammanhang utanför webben så som i node.js och vissa databaser, som CouchDB och MongoDB. JavaScript kan egentligen användas i alla program så länge det finns en så kallad ”JavaScript-motor” som är ett specifikt program gjort för att exekvera JavaScript-kod. De flesta webbläsare har en JavaScript-motor, Google Chrome och Opera har till exempel en JavaScript-motor vid namn V8 och Firefox ha en annan vid namn SpiderMonkey. Det motorn gör är komplicerat men på en grundläggande nivå kan man säga att den går igenom tre steg:

1. Först läser motorn av scriptet(tolkar koden).

2. Sedan omvandlar(kompilerar) den scriptet till maskinkod som datorn ”förstår”.

3. Sedan körs maskinkoden.

Det som gör JavaScript unikt är möjligheten att ändra HTML och CSS på ett relativt simpelt sätt. Man kan till exempel lägga till och ändra innehåll dynamiskt, kontrollera vad som sker vid olika event, kommunicera med olika servrar, ladda upp och ladda ned filer(t.ex. med AJAX-tekniker). Skriptspråket stöds även av alla de största webbläsarna och är aktiverat som standard. [3][11]

API – Ett API(*Application Programming Interface*) är ett ordnat sätt att hämta och lämna data. API:er gör det möjligt för digitala tjänster att kommunicera med varandra i och med att data inte bara sparas utan även kan återanvändas och kombineras för att skapa nya webbplatser och applikationer. Ett öppet API innebär att man låter andra använda utvalda delar av sin data till skillnad från ett stängt API där endast användare internt i organisationen får ta del av datan. [1]

AJAX-anrop, XMLHttpRequest-objektet – poängen med XMLHttpRequest-objektet är möjligheten att skicka och hämta information som normalt skulle krävt omladdning av sidan. Detta görs ofta med HTTP-metoderna GET och POST. I detta projekt kommer jag använda mig av AJAX-anrop för att hämta data från arbetsförmedlingens API. När något händer på webbplatsen, till exempel att den laddas in eller att besökaren trycker på en knapp så skapas först ett XMLHttpRequest-objekt av JavaScript som sedan skickar en begäran till en server. Servern bearbetar sedan begäran och skickar tillbaka den data som önskas till webbplatsen. Datan omvandlas sedan med hjälp av JavaScript som då kan göra någon typ av skillnad på webbplatsen, till exempel uppdatera innehåll.

Jag kommer gå in mer specifikt på hur detta fungerar under konstruktionen. [3]
[10]

3 Metod

Utformning av kod

Jag kommer kolla på uppgiftsbeskrivningen för att ta reda på information om de olika elementen och dess funktionalitet samt vilken information som ska visas på webbplatsen och var. Jag kommer sedan skriva en lista med alla element och vad deras syfte är samt vilken funktionalitet de behöver och eventuellt mer information. Jag kommer även skapa flödesscheman för alla event.

Skapande av kod

Jag kommer koda i VS Code och använda mig av skriptspråket JavaScript för att skapa funktionalitet och dynamiskt skriva ut innehåll på webbplatsen. Här går jag igenom listan över element och skapar funktionalitet, en åt gången.

Testning av webbplats

Jag kommer gå igenom alla element för att testa om webbplatsen fungerar korrekt. Jag kommer validera den genererade HTML-koden med hjälp av W3C:s valideringstjänst för HTML-kod[2]. Jag kommer även testa webbplatsen i de mest använda webbläsarna och på några olika enheter.

4 Konstruktion

Utformning av kod

Lista över element:

1. **<select id=”Searchlan”>**

   Beskrivning: En <select>-meny för val av län.

   Event: ”*change*”.

2. **<ul id=”Mainnavlist”>**

   Beskrivning: vänstra sidomenyn för val av län.

   Event: ”*click*”.

3. **<input type=”search” id=”searchText”>**

   Beskrivning: textfält för söksträng.

   Event: inget. Endast en sträng.

4. **<button id=”searchbutton”>**

   Beskrivning: knapp för att starta sökning.

   Event: ”*click*”.

5. **<input type=”checkbox” id=”onlyit”>**

   Beskrivning: checkbox för att endast visa IT-jobb.

   Event: ”*change*”.

6. **<input type=”number” id=”numrows”>**

   Beskrivning: ändrar antal annonser som laddas in.

   Event: ”*change*”.

7. ** :**

   Beskrivning: högra sektionen där annonser skrivs ut.

Event: inget. Endast ett element för utskrift.

Händelselyssnare:

Alla listade element ovan behöver ges en funktion som triggas igång av deras olika event. Detta göra jag i JavaScript med hjälp av så kallade händelselyssnare som beskrivs nedan:

1. Först väljer jag det element som ska köra funktionen. Till exempel ett element med *id=”element”*. Jag hämtar sedan detta element och lagrar den i en variabel på detta vis: `var elementEl = document.getElementById('element');`.

2. Sedan måste jag välja vilken typ av event som ska trigga igång funktionen. Till exempel ett ”klick”-event som triggar igång funktionen när besökaren klickar på elementet. (Alla event är listade ovan).

3. Till sist kopplar jag ihop funktionen som ska triggas igång av eventet på det utvalda elementet. Detta gör jag genom att lägga till en händelselyssnare vars uppgift är att indikera vilken funktion som ska köras när ett event triggas igång på ett specifikt element. Jag lägger till en händelselyssnare i JavaScript med metoden ”addEventListener()” på detta vis: `elementEl.addEventListener('click', functionName, false);`. Första parametern är den typ av event som ska trigga igång funktionen som är angiven i andra parametern. Den sista parametern i metoden bestämmer vilken riktning man vill trigga eventet. False är ofta standardvalet och har bäst webbläsarstöd.

Hämta innehåll, AJAX-anrop:

När ett event triggas på något av de angivna elementen så körs en funktion som har angivits i händelselyssnaren. Jag kan sedan använda denna funktion för att skriva ut innehåll till HTML-dokumentet. Innehållet hämtar jag från arbetsförmedlingens API med hjälp av AJAX-anrop.

För att AJAX-anropet ska fungera så måste jag först skapa ett XMLHttpRequest-objekt: `var xhr = new XMLHttpRequest();`. Jag specificerar sedan vilken sida jag använder för att skicka min begäran(request) med metoden *open()* på detta vis: `xhr.open('GET', mainURL, true);`. Jag använder GET-metoden i första parametern eftersom jag endast vill hämta data från servern. I andra parametern skriver jag URL:en till API:t [6]. ”mainURL” är en variabel som håller grund-URL:en(`http://api.arbetsformedlingen.se/af/v0`). Grund-URL:en kommer byggas på och den fullständiga URL:en kommer skilja sig åt i de olika AJAX-anropen beroende på vad jag vill hämta för innehåll.

Jag skickar sedan min begäran till servern med metoden *send()*.

När servern har tagit emot min begäran så skickar den tillbaka JSON-data som en sträng. I AJAX-anropet hämtar jag JSON-datan med XMLHttpRequests egenskap *responseText()* [4]. Jag parsar/omvandlar sedan strängen till ett JavaScript-objekt med metoden *JSON.parse()*[5] på detta vis: `var jsonData = JSON.parse(xhr.responseText);`.

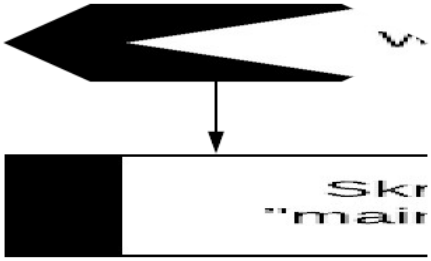
Sedan loopar jag igenom objektet med en for-loop: `for (var i = 0; i < jsonData; i++)`. Jag kan sedan använda denna for-loop för att skriva ut specifik data från servern till HTML-dokumentet. Utskriften görs med metoden innerHTML.

Med denna metod kan jag nu skapa flera AJAX-anrop för att begära specifik data från arbetsförmedlingens API som jag sedan kan skriva ut dynamiskt på webbplatsen.

Skapande av kod

document, onload():

Flödeschema:



Jag börjar med att ladda in all data som ska visas från början när besökaren öppnar sidan. Det är menyvalen för län i menyn till vänster och i select-menyn som ska laddas in vid start. Jag skapar då först en händelselyssnare som laddar in funktionen ”onload()” med hjälp av eventet *DOMContentLoaded* som triggas när webbplatsen laddats in. Den ser ut såhär: `document.addEventListener("DOMContentLoaded", onload, false);`

I onload()-funktionen använder jag mig av ett AJAX-anrop för att ladda in data från arbetsförmedlingens API. Min begäran ser ut såhär: `xhr.open('GET', mainURL+"/platsannonser/soklista/lan", true);`. ”mainURL” är grund-URL:en som följs upp av den specifika adressen där jag kan hämta listan över alla län [7].

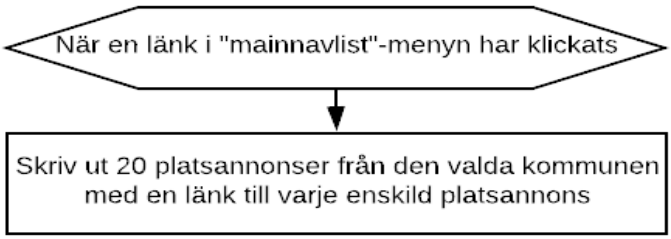
För att skriva ut datan på rätt ställe(till menyerna) så behöver jag deras id och i detta fall är det elementen med id ”*searchlan*” och ”*mainnavlist*”. Jag hämtar dessa id:n från HTML-dokumentet med *getElementById()*-metoden och lagrar de i en variabel på detta vis: `var mainnavlistEL = document.getElementById('mainnavlist');`. När jag sedan loopar igenom objektet så använder jag

dessa variabler för att skriva ut datan till HTML-dokumentet med *innerHTML*-metoden.

Det jag skriver ut för varje menyval(län) är dess id och namn samt andra mindre viktiga detaljer. Jag loopar igenom objektet med denna for-loop: `for (var i = 0; i < jsonData.soklista.sokdata.length; i++)`. ”jsonData” är JavaScript-objektet, i ”soklista” finns listan med alla län och i ”sokdata” finns en lista med sökkriterier. Loopen kör då igenom hela listan och sen kan jag skriva ut alla specifika parametrar. Ett exempel på hur jag skriver ut id:n för varje li-element i listan: `"<li id='"+jsonData.soklista.sokdata[i].id+"'>`. Id är alltså länens unika id:n som jag kommer använda senare för att skriva ut platsannonser från specifika län [7].

mainnavlist, mainnavlistContent():

Flödesschema:



När båda menyerna med länen är utskrivna så måste jag ge varje menyval funktionalitet. När man klickar på ett län i någon av menyerna så ska platsannonser från det länet skrivas ut. Här använder jag i princip samma metod som vid inladdningen av menyer.

För elementet mainnavlist använder jag en händelselyssnare som triggar igång funktionen loadContent() vid klick på ett av menyvalen. Den ser ut såhär: `mainnavlistEL.addEventListener("click", loadContent, false);`. I loadContent()-funktionen använder jag sedan ett AJAX-anrop med denna begäran: `xhr.open('GET', mainURL+"/platsannonser/matchning?lanid=" + e.target.id, true);`. URL:en hämtar en lista med platsannonser utifrån sökkriteriet ”län” som bestäms av parametern ”Lanid”. ”Lanid” är länets unika id som jag redan skrivit ut när sidan laddades in. När besökaren klickar på ett län så fylls den parametern i, till exempel Lanid=10. Då skickar begäran med id:t på länet och då kan platsannonser för det länet hämtas från servern. Om länet inte har något id så är denna parametern tom(Lanid=), då skrivs inget ut eftersom begäran inte skickar med en full URL-kombination som då inte är godkänd för att erhålla en matchningslista med platsannonser från servern. Lanid är en {M}-parameter vilket innebär att minst en parameter måste finnas [7].

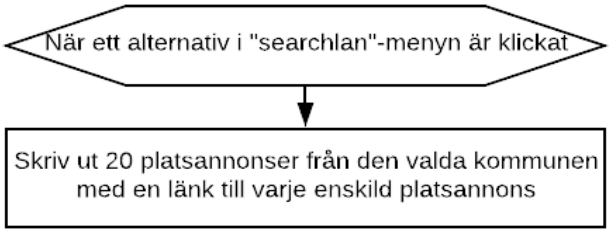
För att lägga till id:t i URL:en så använder jag mig av kodsnutten: `e.target.id`. ”e” är en förkortning för event och ”event.target” refererar till det ele-

ment som triggade igång eventet [8]. Jag lägger också till id för att referera till det attributet som jag vill ha.

När jag sedan får tillbaka data från servern så loopar jag igenom objektet med denna for-loop: `for (var i = 0; i < jsonData.matchningslista.matchningdata.length; i++)`. matchningslista är en rotnod som wrappar in listan och matchningdata är en array med alla platsannonser [7]. Jag skriver sedan ut varje platsannons i varsitt <article>-element med information om tjänsten samt en länk till fulla annonsen på arbetsförmedlingens webbplats. Annonserna skrivs ut i elementet med id ”info”.

Searchlan, searchlanContent():

Flödesschema:

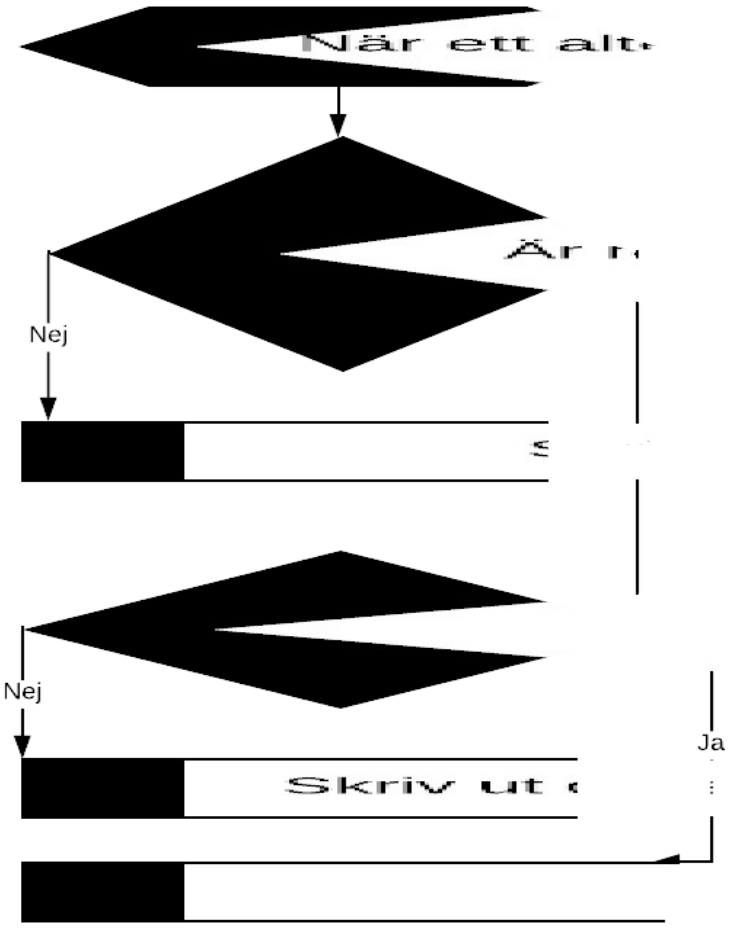


För elementet searchlan gör ungefär jag samma sak fast jag refererar till id:t på ett lite annorlunda sätt. När jag skrev ut select-menyn vid inladdning så skrev jag ut alla länens id i attributet *value* i varje <option>-element. Attributet value refereras sen med kodsnuttet: `searchlanEl.value`. Händelselyssnaren är även lite annorlunda eftersom det är ett <select>-element. Här använder jag eventet *change* som triggas när ett menyval ändras i <select>-elementet [9].

Jag måste även lägga till att searchlan-menyn alltid visar det valda länet oavsett vilken meny besökaren klickar på. Jag lade till denna kodsnutt i mainnavlistContent()-funktionen: `searchlanEl.value = e.target.id;`. Det innebär att om man klickar på ett menyval i mainnavlist så ändras även menyvalet i serchlan, alltså dess värde ändras. Jag kan sedan använda searchlan.value för att ange det valda länet i alla AJAX-anrop som inte kan skicka med en egen ”lanid”-parameter.

numrows, numrowsChange():

Flödesschema:

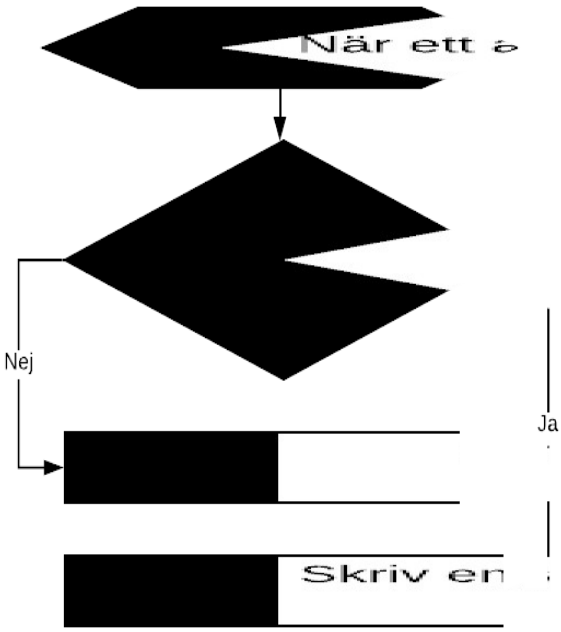


Till detta <input type=”number”>-element behöver jag implementera funktionalitet som hämtar det antal platsannonser som fylls in i elementet. Jag skapar då först en händelselyssnare som ser ut såhär: `numrowsEl.addEventListener("change", numrowsChange, false);`. När elementets värde ändras så körs funktionen numrowsChange()-funktionen. I funktionen skapar jag sedan ett nytt AJAX-anrop där jag lägger till en extra parameter i min begäran. Den parameter jag lägger till heter ”antalrader” och anger hur många rader som ska returneras i matchningslistan(antal platsannonser). ”antalrader”-parametern är en {V}-parameter vilket innebär att den kan utelämnas och om inget argument anges så läses 20 platsannonser ut.

Värdet till parametern får jag från <input>-elementet med id ”numrows”. Detta sker på samma sätt som i <select>-menyn ovan. Det jag lägger till i begäran är då: `+ "&antalrader=" + numrowsEl.value`. Parametern och värdet som fylls i <input>-elementet med id:t ”numrows.

onlyit, onlyitChange():

Flödesschema:



Till detta checkbox-element behöver jag implementera funktionalitet som hämtar platsannonser inom yrkesområdet IT när den är ifylld. Varje yrkesområde har ett specifikt id som ligger i parametern ”yrkesomradeid”. IT-området har id=3 så det är den parametern jag vill lägga till i begäran till servern i mitt AJAX-anrop.

För att ändra denna parameter när besökaren klickar i checkboxen så använder jag mig av en händelselyssnare med eventet ”*change*” som kör en funktion vid namn ”onlyitChange()”: `onlyitEl.addEventListener("change", onlyitChange, false);`.

I denna funktion skapar ett nytt AJAX-anrop med en ny begäran med en extra paramter. Det jag lägger till som parameter i AJAX-anropet är elementets värde likt det jag gjorde för elementet ”numrows”: `+ "&yrkesomradeid=" + onlyitEl.value`.

För att ta reda på detta värde har jag skapar en if-sats inuti funktionen som ser ut såhär:

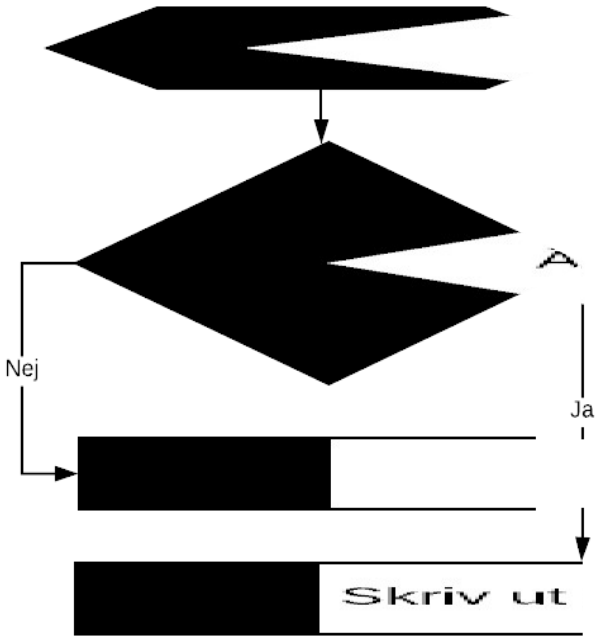
```
if (e.target.checked) {
	onlyitEl.value = "3";
} else {
	onlyitEl.value = "";
}
```

Funktionen kollar om checkboxen är ifylld med hjälp av attributet ”*checked*”. Om den är ifylld så ändras värdet till 3, annars är värdet tomt. Vid inladdning

måste jag även ange att värdet är tomt eftersom standardvärdet är ”on” vilket kommer resultera i att URL:en i AJAX-anropet blir felaktig.

searchText, searchbutton, search():

Flödesschema:



Dessa element samt funktion ska tillsammans fungera som en sökfunktion där besökaren kan söka på olika nyckelord. SearchText håller den text som besökaren fyllt i och searchbutton är knappen som kör sökningen. För att sökningen ska köras så skapade jag först en händelselyssnare på sökknappen: `searchbuttonEl.addEventListener("click", search, false);`. När knappen klickas så körs funktionen search(). I denna funktion skapar jag ett nytt AJAX-anrop med en ny begäran där jag lägger till en extra parameter: `+ "&nyckelord=" + searchTextEl.value`. Parametern heter ”nyckelord” och ska bestå av fri text med ett eller flera sökord [7]. Texten är helt enkelt elementet ”searchText”:s värde vilket jag hämtar med metoden value.

Övrigt:

Län-rubrik: Jag lade till en rubrik för valt län i varje AJAX-anrop med denna kodsnutt: `infoEl.innerHTML += "<h2>" + searchlanEl.options[searchlanEl.selectedIndex].text + "</h2>";`.

Aktuell sökning: Jag skriver ut värdet från sökfältet om någonting är ifyllt vid en sökning. Detta för att användaren ska ha en bättre översikt över det den söker.

Lättläst datum: Datumet som returneras från JSON-filen är svårläst och innehåller onödigt mycket information för min webbplats. Därför har jag kortat av strängen så att den endast visar år, månad och dag. Detta gör jag med metoden *substring()* som returnerar en ny förkortad sträng där jag anger startvärde och slutvärde i den redan existerande datum-strängen. För att läsa ut endast datumet(t.ex. 2018-08-01) så behöver jag 10 tecken. Jag anger då 0 som första värde eftersom den ska starta på första bokstaven och 10 som andra värde eftersom endast de 10 första tecknen ska läsas ut.

Testning av webbplatsen

Jag har testat all funktionalitet på element i massvis av olika kombinationer och allting fungerar felfritt. Jag testade även i olika webbläsare: Chrome, Firefox, Safari, Opera, Edge och Explorer. Allting fungerar som det ska oavsett webbläsare. Jag testade även på min iPhone 6s och det fungerande felfritt.

Jag validerade den generade HTML-koden vilket inte gav några fel. De varningar som kom upp berodde inte på den JavaScript-koden utan på det färdiggjorda HTML-dokumentet vilket jag inte fick göra några ändringar på. Validering gjorde jag med W3C:s HTML-valideringstjänst [2].

5 Resultat

Jag listade alla utvalda elementen och tog reda på vad de skulle ha för funktion. Jag implementerade sedan elementens listade funktionalitet med hjälp av JavaScript. Jag gav även några övriga element funktionalitet och lade till extra innehåll som jag tycker förbättrar användarupplevelsen. Till exempel att webbplatsen laddas om när man trycker på loggan och felmeddelanden för att informera besökaren om det som gick fel.

Jag har testat all funktionalitet på element i massvis av olika kombinationer, på olika enheter och i olika webbläsare. Jag har även validerat koden för att se om den är korrekt. Inga ändringar behövde göras efter sista testet och allting ska fungera felfritt.

6 Slutsatser

Arbetet hade en trög start när jag försökte mig på det första gången och det kändes nästan omöjligt i början. I mitt andra försök så var jag mer påläst om varenda liten detalj och lade ned mer tid på att lära mig JavaScript som helhet istället för endast de delar som projektet omfattade. Det resulterade i att det blev oerhört simpelt att skapa koden och arbetet bara flöt på.

Jag tycker att jag har skapat en väl fungerade webbplats med extra funktioner / innehåll som skapar mer tydlighet och användbarhet. Det går ju alltid att lägga till ännu fler filtreringsfunktioner, felmeddelanden och övriga detaljer men jag tycker ändå att den nuvarande webbplatsen uppfyller dess syfte och gör det lätt för användaren att uppnå sitt mål.

Jag var lite osäker när jag skapade mina flödesscheman. Hur mycket information jag behövde ta med i varje ruta och om jag endast skulle ta med ett specifikt event eller även de event som körs innan. Jag var även lite osäker på hur jag skulle formulera mig och hur teknisk beskrivningen skulle vara. Jag tror och hoppas ändå att mina flödesscheman är lätta att förstå för både nybörjare och mer erfarna.

Som helhet var projektet oerhört givande med många roliga utmaningar som jag kan dra nytta av i framtiden. Jag är nu relativt bekväm med språket och det känns som jag lagt en bra grund att bygga vidare på för att lära mig ännu mer om JavaScript och kunna dra nytta av dess många fördelar.

Källförteckning

[1] Portabla Media, ”Vad är API:er och hur använder man dem?” https://www.portablamedia.se/vad-ar-api-er-och-hur-anvander-man-dem/ Publicerad 2013-09-26. Hämtad 2018-07-12

[2] W3C, ”Markup Validation Service” https://validator.w3.org/ Hämtad 2018-07-12

[3] Jon Duckett. JavaScript and Jquery: Interactive Front-End Web Development. 1 uppl. Wiley. 2014.

[4] Mozilla, ”XMLHttpRequest.responseText” https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/responseText Publicerad 2018-04-22. Hämtad 2018-07-21

[5] Mozilla, ”JSON.parse()” https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse Publicerad 2018-01-10. Hämtad 2018-07-21

[6] Mozilla, ”XMLHttpRequest.open()” https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open Publicerad 2018-05-02. Hämtad 2018-07-21

[7] Arbetsförmedlingen, ”Tjänstegränssnitt, API platsannons” https://www.arbetsformedlingen.se/download/18.40fa4e7b159ff029331706ca/1486976282357/teknisk-beskrivning-lediga-jobb.pdf Publicerad 2016-11-28. Hämtad 2018-07-22

[8] Mozilla, ”Event.target” https://developer.mozilla.org/en-US/docs/Web/API/Event/target Publicerad 2018-04-13. Hämtad 2018-07-22

[9] Mozilla, ”change” https://developer.mozilla.org/en-US/docs/Web/Events/change Publicerad 2017-12-22. Hämtad 2018-07-22

[10] WikiPedia, ”AJAX” https://sv.wikipedia.org/wiki/AJAX Publicerad 2018-08-21. Hämtad 2018-08-26

[11] JavaScript.info, ”An introduction to JavaScript” http://javascript.info/intro Hämtad 2018-08-26