# Rapport inlämningsuppgift: Städer och länder, 10 Yhp

## Beskrivning av koden

Uppgiften är löst med tre filer; index.html, style.css och script.js samt de två json-filerna. Koden är till största delen skriven i javascript. I html-filen finns bara lite grundkod i bodyn samt länkade css-filer och scriptfiler. Jag har valt att använda Bootstrap som grundläggande ramverk, vilket också gör att Bootstraps css och javascript ligger länkade i html-filen.
I den lokala CSS-filen ligger specifik styling för layouten, annars sköts det mesta av Bootstraps css.
I script.js sker den dynamiska layouten till DOM. I js-filen är till att börja med är några variabler definierade till tre fasta DOM-element i ett app-objekt. Koden består till största delen av sex funktioner, varav en tillsammans med en EventListener. En av funktionerna är också inuti en annan funktion, då den bara är till för en mindre uppgift.
Innehållet i json-filerna hämtas med hjälp av fetch i samarbete async och await.
Javascriptet jobbar dynamiskt med att göra element i DOM, beroende på innehållet i json-filerna. Ingen data från json-filerna är hårdkodad i javasriptet, vilket gör att det går att lägga till hur många länder och städer i json-filerna som helst, samt ändra namn på städer och dess id så länge samma struktur följs, d.v.s bara värden får ändras och länder och städer måste fortfarande vara kopplade.
Elementen läggs till i DOM med hjälp av funktionen {insertAdjacentHTML}. Id på elementen sätts utifrån värden i json-filerna, och i vissa fall också parametrar för funktioner som ska utföras vid ”onklick”. På några ställen används {innerHTML} för att ta bort element.
Datas sorteras och jämförs med hjälp av for-loopar och appliceras med hjälp av påståenden (statement), d.v.s. if- och else-satser.
Arrayer används för att lagra data. LocalStorage används också för att lagra data och göra så att sidan kan uppdateras utan att olika val försvinner. För att transportera information från en array till LocalStorage och vise versa, så används funktionerna JSON.parse() och JSON.stringify() tillsammans med funktionerna LocalStorage.getItem() och LocalStorage.setItem().

## Skäl till vald lösning

Valet av lösning har gjorts utifrån att webbsidan ska vara dynamisk. Helt oberoende av vilken information (värden) som tas bort, läggs till eller ändras i json-filerna. Det gör att ingen specifik information kan hårdkodas in i koden. Valet att använda Bootstrap var för att kunna göra en hyfsad responsive design med lite roliga funktioner på kort tid och på enklast/snabbast möjliga sätt.
Att endast använda index.html och skapa elementen till DOM med hjälp av javascript valdes också för att dels, göra sidan snabbare, d.v.s. slippa ladda in nya sidor men också för att slippa göra fler scriptfiler och html-filer.
Jag har valt att jobba med flera olika funktioner. Jag har endast en gång använt en {eventListener()}, då det var enda knappen som var hårdkodad i html-dokumentet. I övriga fall använder jag {onclick} i

html-koden. Skälet till det är att jag inte kan sätta en ”lyssnare” på en obefintlig knapp, är för att de skapas dynamiskt. Med {onclick} kunde jag få en viss knapp att aktivera en funktion och samtidigt skicka med en parameter så att funktionen visste vilken knapp som var klickad.
Jag valde också att hämta data från json-filerna med {fetch}. Skälet var enkelheten jämfört med {XMLHttpRequest}. Jag har tillsammans med {fetch} ¨jobbat med {async} och {await} för att data ska hinna läsas in. Skälet var för att jag tycker det också är enklare än {.then} samt för att jag känner mig säkrare på {async} och {await}.
Jag har med hjälp av arryer sparat data. I något fall för att slippa skriva en stor räkneformel (sumOfAllCitiesPopulation) rad: 79. Men också för att kunna spara och hämta data i LocalStorage.
Jag har använt for-loppar för att loopa igenom både arrayer och hämtad data från json-filer. Kunde ha använt {while} men jag tycker {for} är smidigare. Jag har också använt {if} och {else} för att stämma av olika kriterier.
Jag har använt andra inbyggda funktioner som; insertAdjacentHTML(), getElementById(), addEventListener(), setAttribute(), parse(), getItem() m.fl. Den funktion som jag valt att använda för att infoga element i DOM är {insertAdjacentHTML()}. Mitt skäl är för att jag kan skriva ganska mycket html-kod tillsammans och placera den på en specifik plats. Jag tycket det var ett smidigt och bra sätt, samtidigt som jag också kunde lägga i variabler i html-koden.
Tex. {onclick="countryInfo('+land[i].id+')"}. Jag kunde inte hitta några säkerhetsproblem med det, men tar gärna emot feedback om det finns. Nackdelen var att det blev en del långa rader i koden, eftersom jag inte kunde skriva koden på flera rader. Det gör kodsträngen något svårläst. Kanske finns det något sätt att göra det på som jag inte känner till.

## Personlig reflektion av vald lösning samt andra alternativ

En rolig och utmanande uppgift!
Jag gjorde som vid tidigare lite större uppgifter, avsatte en stor del av tiden till att förbereda mig innan kodningen. Ännu en gång upplever jag detta som mycket positivt!
På GitHub så skapade jag Issues, för olika uppgifter. Först skissade jag en layout, utifrån Bootstarps layouter och funktioner (skissen hittas längre ner i detta dokument). Efter det skissade jag på upplägg av javascript-koden som ett MindMap-upplägg (även denna hittas i slutet av detta dokument). Detta gjorde att jag var tvungen att både hitta lösningar och strukturera flödet i koden innan jag kunde börja koda. Det på något sätt förbereder hjärnan innan, så att kodandet går mycket lättare. När det var klart började jag med att göra layouten i html-filen och stylade det med CSS. Allt utan javascript. När jag sedan började med javascriptet så kunde jag vi de tillfällen jag skulle infoga kod i DOM med {insertAdjacentHTML()} nästan bara kopiera den kod som var färdig i html-filen. Jag behövde självklart modifiera den lite, som att med variabler skapa id och onclick, men det gick ändå mycket smidigt.

Självklart uppstår det en del problem som jag inte förutsett, men det gick ändå hyfsat enkelt att lösa dem, när programmeringen sker i ordning som på MindMap-skissen. Uppskattningen att planera ungefär lika mycket som kodningen, stämmer förvånansvärt bra. Dessutom är det mycket roligare när allt inte bara är rörigt (både kod och i huvudet), som det finns en tendens att det blir utan planering. Efter att jag fått en fungerande kod, så rensades koden på alla {console.log()}, strukturerades och kommenterades.
Gällande kraven som skulle uppfyllas:

- JSON-filerna används utan modifiering och det funkar att fylla på med både städer och länder, samt även byta värden.
- Sidan har en sidomeny där alla länder visas och dynamiskt ökar eller minskar beroende på antal i land.json. Klickar man på knappen visas landets städer på höger sida, eller under i mobilvy.
- Klickar besökaren på ”Läs mer om {stadnamn}” visas en text som talar om antalet invånare i staden.
- Det finns också en switch-knapp med beskrivning ”Besökt” vid varje stad. Klickar man på denna sparas stadsid:t i en array som också överförs till LocalStorage. Klickar man tillbaka switchen så försvinner stadsid:t urt arryen och LocalStorage.
- En knapp med titel ”Besökta städer” finns i menyn under alla städer.
- Knappen ”Besökta städer” visar de städer som besökaren markerat som besökta, samt antalet invånare i varje besökt stad. Under alla städer finns en collapsruta (default utfälld) med kuriosa om de besökta städerna. I detta fall den sammanlagda mängden invånare i de besökta städerna.
- Längst ner på samma sida finns en knapp som rensar historiken på besökta städer. Den tömmer arrayen, LocalStorage samt de utskrivna besökta städerna. Alla Besökt-switchar återställs också, men det sker automatiskt eftersom LocalStorage töms.
- Analysen, reflektionen av kod och struktur samt motiveringar finns i denna rapport.

Jag upplever det som att jag fått till en väl fungerande lösning på uppgiften och som uppfyller alla ställda krav.

Länk till repo: https://github.com/campus-varnamo/stader-lander-NikBjo72

Länder

Sverige

Finland

Norge

Besökta städer

# Sverige

| Städer: | Besökt: |
| --- | --- |
| Stockholm | ☑ |
| Läs mer om Stockholm | |
| *Stockholm har 345'000 invånare* | |
| Uppsala | ☑ |
| Läs mer om Uppsala | |
| Linköping | ☐ |
| Läs mer om Linköping | |

Länder

Sverige

Finland

Norge

Besökta städer

# Besökta städer

| Stad: | Invånare: |
| --- | --- |
| Stockholm | *345' 000 st* |
| Uppsala | *234' 000 st* |

**Möjliga kontakter**

*Under dina resor i dessa städer har du uppnått en potensiell möjlighet att träffa 579´000 människor.*

Rensa besökshistoriken

# Script.js

- Sätta variabler på statiska element
- function getContries()
  - fetchar land.json
  - Skapar knappar med loop och id="id" onclick="countryInfo(id)"
- Listener på "Besökta städer"-knapp som kör funktion.
  - function visitedCities()
    - Fetchar stad.json
    - Med (insertAdjacentHTML) skapa rubrik och två kolumner, Stad: och Invånare: samt "Rensa besökshistorik"-knapp som clearar LoSt.
    - Kolla vilka stad-Id som finns i LoSt och skriv ut staden samt invånarantal. Loopa igenom array.
    - Med (insertAdjacentHTML) skapa en yta som innehåller "möjliga kontakter"-text samt hämtar mängden människor för varje stad som finns i LoSt och lägger ihop till summa som skrivs ut i texten.
- function countryInfo(id)
  - fetchar stad.json
  - Kollar vilken knapp som är klickad genom inmatad parameter (land Id).
  - Skapa rubrik med landsnamn
  - Skapa två kolumner mer Städer: och Besökt:
  - Med (insertAdjacentHTML) i loop skapa:
    - stadsnamn
    - Besökt-check-box onChange="addCityToLocalStorage(city Id)". Bootstrap switch-knapp.
    - collapsknapp samt stadsinformation
    - Raddelare (streck)
  - If (stad Id finns i LoSt sätt checkbox med samma id till attribut "checked".

- function addCityToLocalStorage(cityId)
  - Först till array sedan till LoSt. Parse och stringify.
- function clearLocalStorage()
  - Rensar LocalStorage
  - Rensar besökta städer array
  - Rensar utskrivna städer och invånare