Inlämningsuppgift 2 - CITY TRAVEL

Täckta kursplansmål:

4. Redogöra för hur en kommunicerar med servern
5. Ge exempel på olika alternativ för att hantera data i webbläsaren

Övergripande

I denna uppgift ska du bygga en webbapplikation i vilken man kan söka efter städer i världen och få information om väder och tips på sevärdheter att besöka. Denna information hämtar du från tredje parts-API:er (Foursquare, Open Weather). Varje gång en användare söker efter en stad gör du anrop i bakgrunden till dessa api:er för att sedan presentera informationen.

Mål: En webbapplikation där användare kan söka efter städer i världen och få information om väder och sevärdheter att besöka.

Avgränsning: Använd inte JQuery. Uppgiften ska lösas med HTML, CSS och JavaScript (ES5&ES6). Bootstrap och Sass är tillåtet.

Använd Git för att verisionshantera koden. Gör minst en commit om dagen. Ha din kod på ett publikt GitHub repo som du länkar till när du lämnar in din .zip av projektet

G krav

- Skriv en fungerande sida enligt ovan
- Beskriv med kommentarer i din kod **vart** och **hur** du gör för att kommunicera med servern

Deadline

Den 15/2, Mån kl 23:55

Beskrivning

Foursquare Web API

För att använda Foursquare och Open Weather API:er behöver du skapa developer konton hos dessa. Foursquare är tjänsten du ska använda för att hämta information om sevärdheter, och Open Weather för väderleksinformation.

Följ länken här: https://foursquare.com/developers/login för att skapa ett konto hos Foursquare. När du har verifierat din email så kan du skapa en applikation. Döp applikationen till City Travel (Du kan använda vilken länk som helst för Company eller App). Foursquare kommer därefter ge dig ett client ID och client secret, dessa ska du skicka med sedan när du gör anrop till deras API, så spara undan dessa.

För att förstå hur du ska använda API:et följ länken: https://developer.foursquare.com/docs/api-reference/venues/explore/ och läs noggrant igenom. I dokumentationen så saknas tyvärr en url-parameter som behövs för att göra ett lyckat anrop: v=[dagens datum], ex: http://api.foursquare.com/v2/venues/explore?.....&v=20210114.

De url-parametrar som du kommer använda är:

- client_id
- client_secret
- near
- limit (högst 10 sevärdheter ska hämtas)
- v

Open Weather Web API

Följ https://openweathermap.org/appid för att skapa ett gratis konto hos Open Weather. När du skapar konto och ange ditt namn som företag och för "purpose" välj "other". När du har verifierat din email så kan du gå till "API keys", och där ska du ha en färdig API nyckel med namnet "default", spara denna. För att förstå hur du ska använda API:et följ länken https://openweathermap.org/current och läs noggrant igenom. Den url som du kommer använda utan parametrar ser ut som följande: https://api.openweathermap.org/data/2.5/weather.

De url-parametrar som du kommer använda är:

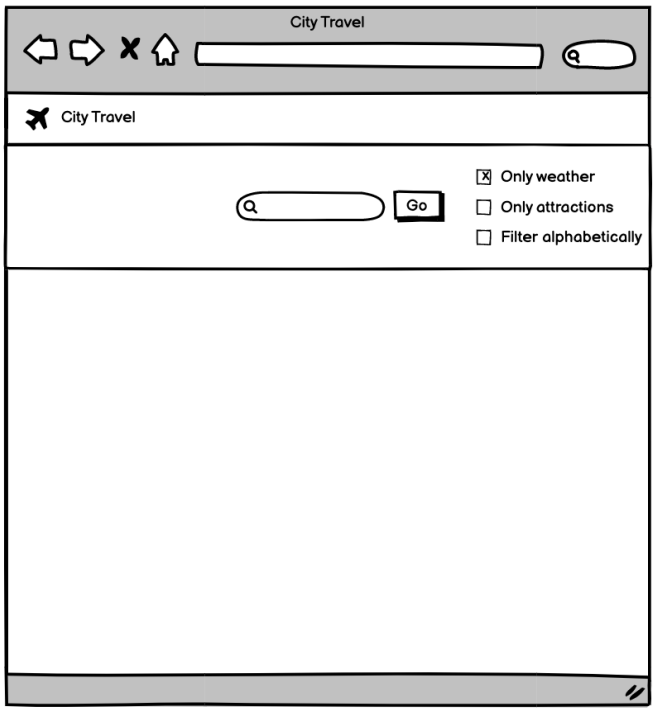
- q
- APPID

Layout i HTML och CSS

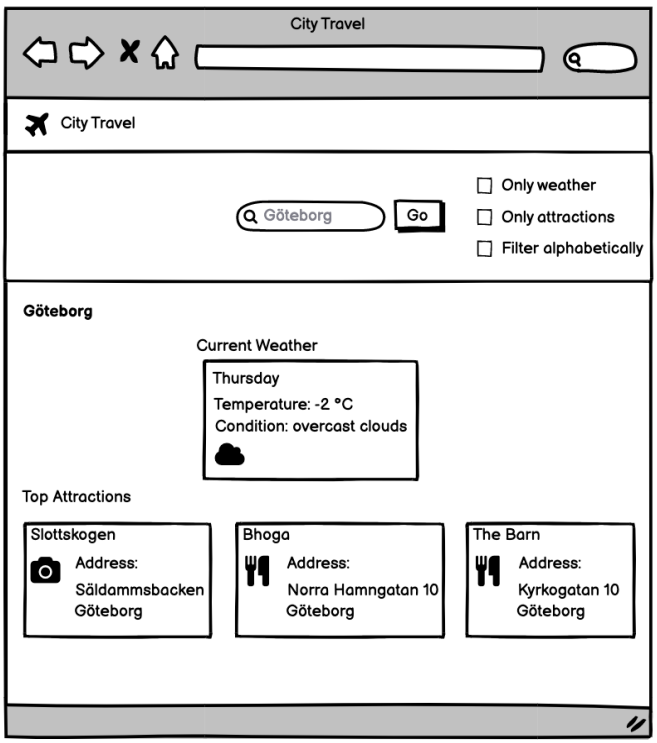
Första delen i uppgiften är att skapa en bra/tydlig layout för applikationen. I den här uppgiften behöver du inte tänka på responsiv design och länkar till andra dokument, utan vi använder endast 1 HTML dokument (index.html) och 1 CSS dokument (index.css). När du har skapat dessa dokument enligt designspecifikationen kan du börja med uppgiften att lägga till JavaScript kod, vilket är huvuddelen i den här uppgiften.

Designspecifikation

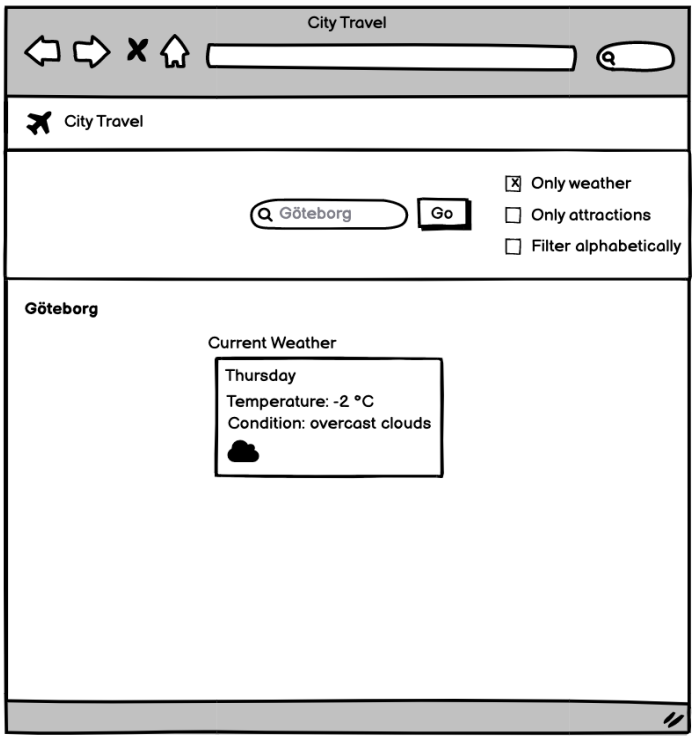
Nedan följer mockups på hur applikationen ska fungera.



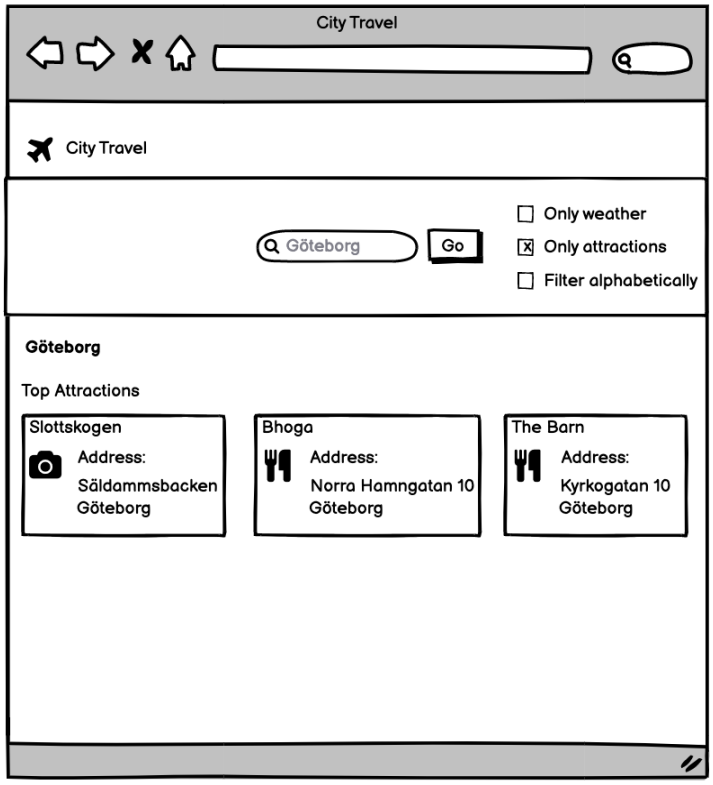
I det här fallet har användaren öppnat sidan och inte sökt något än.



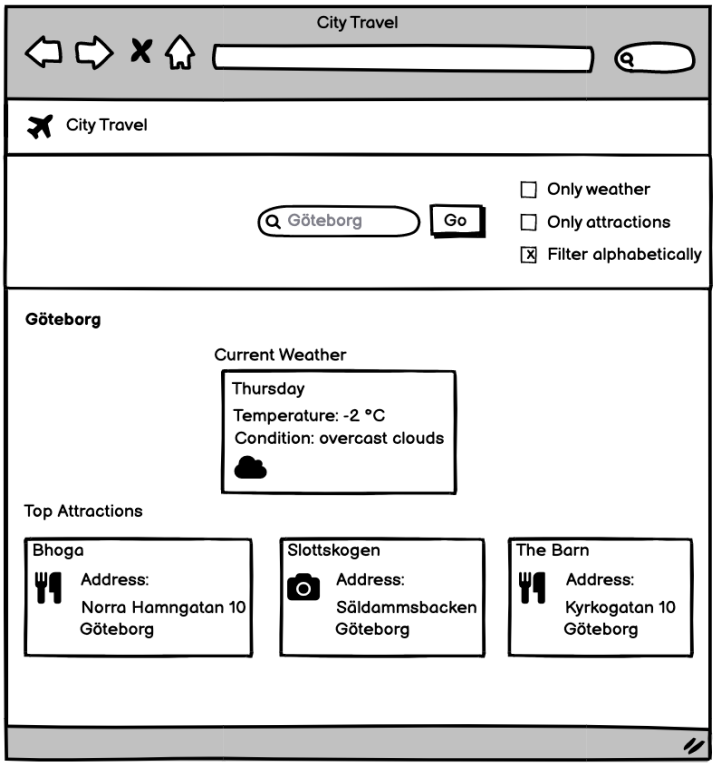
I det här fallet har användaren sökt efter staden Göteborg och fått följande information; Stadens namn, aktuell väderleksinformation, dom 3 (kan vara upp till 10) populäraste sevärdheterna i staden med adress och kategori av sevärdhet genom en ikon.



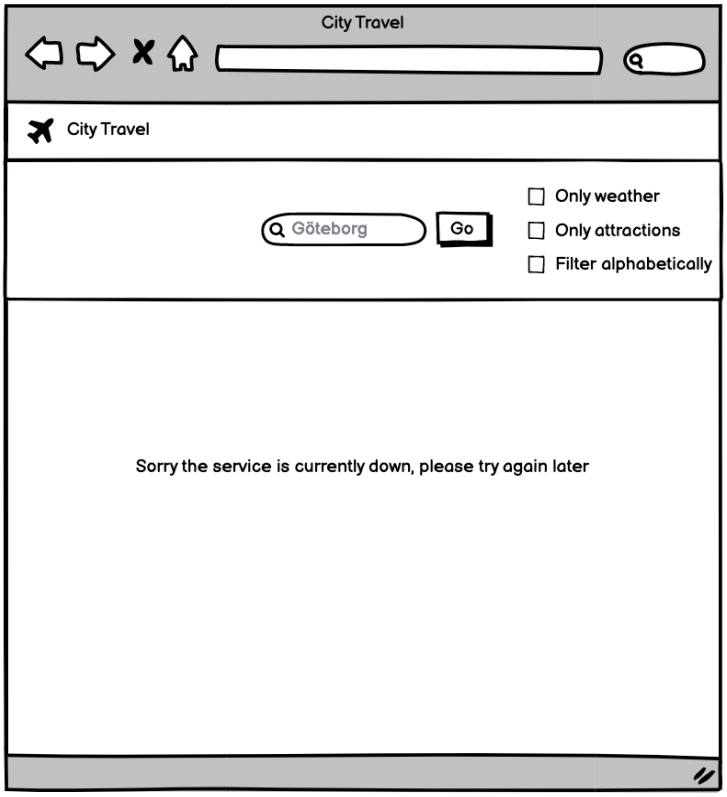
I det här fallet har användaren sökt efter staden Göteborg och fyllt i att denne enbart vill se väderleksinformation.



I det här fallet har användaren sökt efter staden Göteborg och fyllt i att denne enbart vill se dom populäraste sevärdheterna.



I det här fallet har användaren sökt efter staden Göteborg och fyllt i att denne vill se sevärdheterna sorterade i alfabetisk ordning.



I det här fallet så har användaren sökt efter staden Göteborg men någon av tjänsterna ligger nu nere och är oåtkomlig.

JavaScript

För att lyckas utforma applikationen med *god struktur* och *modularitet* så bör du göra separationer av logik i applikationen m.h.a funktioner och objekt, enligt tycke och smak. Det viktiga är att det är enkelt för en utomstående att läsa koden och följa den logiska tråden i applikationen. Skriv inte bara en funktion eller metod som gör allt, utan dela upp logiken; se https://refactoring.guru/es/smells/long-method för en utförligare beskrivning.

För att göra anrop till tredjepartstjänsterna (API) så kan du använda antingen den nyare 'fetch' APIn eller den äldre 'XMLHttpRequest'. Välj det som du känner att du har lättast för eller som du tycker känns mest intuitivt.

Ni bör också tänka på att använda felhantering i din kod, speciellt för anrop mot tredjepartstjänsterna, dessa kan trots allt ligga nere för tillfället och då vill man som användare få information om detta, som beskrivs i designspecifikationen. Exempelvis skulle ett anrop med felhantering kunna se som följande:

```
try {
  const response = await fetch(urlToFetch);
  if (response.ok) {
    const jsonResponse = await response.json();
    // Visa resultat för användare
  } else {
    // Visa felmeddelande för användare
  }
} catch(error) {
  // Visa felmeddelande för användare
}
```

En viktig princip för spara tid från sökande efter onödiga buggar är att alltid börja med kod som är så enkel som möjligt, exempelvis när du gör en funktion för att hämta data från API:er börja med att först bara göra en console.log med stadens namn, sedan lägg till en fetch med console.log av det resultatet, osv, så att du sakta dubbelkolla varje steg du tar vid utformningen av funktionen.

Tänk även på att använda variabler för saker som url:er, nycklar, osv, detta gör också koden lättare att läsa och modulär.

Punktlista

1. Börja med att skapa API-nycklar på Foursquare och Open Weather enligt beskrivningen ovan.

2. Skapa en mapp på din dator med namnet “inlämningsuppgift 2”.

3. Om du använder git (jag rekommenderar detta) så gör en commit för varje grej i sidan du får att fungera, så att du kan revert:a (“spola tillbaka”) om något går fel.

4. Skapa ett HTML dokument (index.html) och ett CSS dokument (index.css) i mappen.

5. Skapa en layout och styling för sidan med hårdkodade element (fakade sevärdheter och väderleksinformation). På det sättet kan du sedan koncentrera dig på JavaScript delen som är huvuddelen i den här uppgiften.

6. Tänk på att ge id eller class attribut till de element som du behöver hämta data ifrån, modifiera, ta bort osv, så att du sedan kan använda DOM-API:et för att göra detta.

7. Skapa ett JavaScript dokument (main.js) och koppla samman <script src=”main.js”></script> i index.html.

8. Gör en funktion för input från sökfältet. Detta kommer bli din huvudfunktion och döp den till något lämpligt, exempelvis executeSearch.

9. Skapa variabler för API url:erna, använd string interpolation för de olika url-parametrarna.

10. Gör funktioner för själva hämtningen av data från API:erna, en för FourSquare och en för Open Weather. **Gör mycket console.log och step-through debugging**.

11. Använd objekt där det är lämpligt. Exempelvis skulle svaret från ett API-anrop kunna representeras som ett objekt, med en property som anger om resultatet var ok eller inte, samt data som rensats ut från svaret.

12. Börja modifiera DOM med den data du får tillbaka via dina hjälpfunktioner.

13. Skriv så att det dyker upp bra felmeddelanden om något går snett. Dubbelkolla att det fungerar genom att sänka/stänga av ditt internet via googles debug verktyg https://helpdeskgeek.com/networking/simulate-slow-internet-connection-testing/ och genom att söka på städer som inte finns.

14. Slutgiltligen, se över koden och kommentera det du behöver, dubbelkolla G kraven, kanske städa upp bland variabler och variabelnamn.