DU2 - Inlämningsuppgift (U1)
Brief

Pedagogiska tanken

Tanken med U1 är att låta dig få en känsla av hur det kan vara att koda en enkel, men inte såå enkel, interaktiv webbsida.

Du ska använda instruktioner och begrepp som vi har sett i kursen, fram till lektion 6 (första lektionen om funktioner). Du får använda andra instruktioner än de vi har sett, men du måste förstå dem **helt**.

Vi har föreslagit en struktur för programmet, och förklarat den i en guide. Du behöver dock inte följa den föreslagna strukturen så länge webbsidan ser ut och funkar som på videon och koden uppfyller alla krav (se nedan).

Amanuenser

Amanuenserna har fått tydliga direktiv om att **inte** svara på frågor som är direkt relaterade till U1.

Kodredovisning

Precis som i DU1 ingår det i U1 att redovisa sin kod. Detta gör vi för att säkerställa att du förstår all kod som du lämnar in.

Det är ok att lösa uppgiften med andra, men du måste förstå ALL kod som du lämnar in och du måste kunna svara på alla våra frågor. Vid redovisningstillfället har du tillgång till din kod, som i DU1. Det får dock inte förekomma några kommentarer i koden.

Varför vi föreslår en struktur

I början kan det vara svårt att ordna ens tankar kring hur man strukturerar ett program. Eller att ens få några tankar om hur man strukturerar ett program. Det är helt förståeligt och supervanligt.

Det som brukar hända är att man startar någonstans i koden, vanligtvis något man vet hur man löser, och sen bygger runt den startpunkten. Det resulterar ofta i rätt så komplex kod som inte är särskilt robust och där små ändringar kan göra att programmet inte längre fungerar.

Det är väldigt frustrerande men samtidigt väldigt **lärorikt** att lösa programmet på ”ens eget” sätt, utan en så genomtänkt struktur. Vi rekommenderar faktiskt att du försöker lösa U1 först utan att använda dig av vår guide. Det är mycket mer lärorikt att läsa guiden **efter** att ha kämpat några timmar (tre/fyra) med ”ditt sätt”.

Duktig utvecklare

Duktiga utvecklare vet hur man tänker igenom ett program innan man börjar koda den. Du är på väg att bli en duktig utvecklare men är – antagligen – inte där ännu. Så vi förväntar oss inte att du kan tänka igenom detta program (webbsida) helt på egen hand.

Vi inkluderar guiden för att det är viktigt att få exempel på hur ett någorlunda strukturerat program ser ut, och hur man kommer fram till det.

Det finns inget krav på att du ska använda dig av strukturen för att bli godkänd, men målet för dig är inte att bli godkänd utan att bli en duktig utvecklare och då måste du lära dig koda program på ett strukturerat sätt.

Databasen

Databasen består av två arrayer. All info i dem behövs inte för att lösa uppgiften.

Den ena har information om städer (land, elevation, etc). Den andra har information om distans mellan städer. Distanserna har jag fått av CoPilot, så det kan finnas fel i dem.

Varje element i arrayen cities består av ett objekt med information om en stad. Varje stad har ett unikt id (siffra), det är mycket vanligt att element i en databas har ett unikt id (notera dock att det är ovanligt att id startar med 0… det bara blev så denna gången).

Varje element i arrayen distances består av ett objekt med information om distansen mellan två städer. Städerna representeras av deras id. Så första objektet i arrayen:

```
{
  "city1": 1,
  "city2": 0,
  "distance": 930
}
```

…säger att distansen mellan staden med id 1 (Bordeaux) och staden med id 0 (Strasbourg) är 930km.

Notera att objektet nedan INTE finns i arrayen:

```
{
  "city1": 0,
  "city2": 1,
  "distance": 930
},
```

…eftersom det är samma distans mellan Bordeaux och Strasbourg oavsett vilken som är city1 eller city2. Detta är viktigt att ta hänsyn till för att lösa uppgiften.

CSS-kod

fonts.css innehåller @font-face-koden.

index.css innehåller en del regler och de räcker för att formatera sidan. Ids är anpassade till html-filen. Du måste ange rätt klasser när du skapar webbsidan från JS. Du måste förstå reglerna för att kunna använda dem. Det är ok om du föredrar att skapa dina egna regler eller anpassa de som finns (alla eller en del). Sidan ska dock se ut som den på videon.

Kodkrav

Säkerställ att ditt program uppfyller **alla** punkter i kravlistan nedan. Annars kan du inte genomföra redovisningen.

Funktionalitet

1. Sidan ska fungera som den på videon.

Befintlig kod och filstruktur

2. index.html
    1. Innehållet i <body> får inte ändras, förutom href-attributen i a-elementen (se nedan).
    2. Lägg till link-elementen som behövs direkt på HTML-filen.
    3. Lägg till script-elementen som behövs direkt på HTML-filen.
3. font.css får inte ändras. Du måste använda just det typsnittet.
4. index.css får (men behöver inte) ändras. Om du ändrar i filen måste du säkerställa att endast regler som används finns med; rensa koden!
5. database.js får inte ändras alls.

6. index.js måste kompletteras. Befintlig kod (förutom ev. kommentarer) måste vara kvar och användas.
7. Filstrukturen måste behållas. Inga nya filer får adderas.

Visuellt (CSS-formatering)

8. Sidan måste se ut som den på videon.
    1. Du har helt fria händer vad gäller färger. Stad-boxarna som markeras (target, closest, furthest) måste ändra både färg bakgrunds- och textfärg.
    2. Du måste studera index.css för att ta reda på avstånd och storlekar.

JS

9. Du får inte skriva 38 gånger samma kod (en gång för varje stad). Du måste lösa det med loopar.
10. Det måste finnas åtminstone en funktion som gör något meningsfullt.
11. Kontrollera att din webbsida visar rätt information när användaren skriver en stad som inte finns på databasen. (Se bild notFound.png) (Glöm inte fliken!).
12. Kontrollera att din webbsida visar rätt information när användaren skriver en stad som finns på databasen (se bild found.png) (Glöm inte fliken!).

Kommentarer

13. Det får inte förekomma några kommentarer i koden.
    1. Eventuella kommentarer som kan ha funnits med från början måste tas bort.

Filstorlek

14. Zip-filen som du lämnar in måste vara mindre än 200Kb. Ladda gärna ner den från Github.
    1. Ta inte med några filer än just de som behövs för webbplatsen.

Publicering

15. Du måste publicera din sida på webben (på samma sätt som projektet i DU1). Länk-elementet till din sida på webben finns redan på plats i index.html. Du måste uppdatera href-attributet.

Github

16. Projektet ska utvecklas som en Github-repository från start. Du måste ha minst 15 meningsfulla commits. Om du gör det från början så blir det en naturlig del av din kodningsprocess. Det ger dig också en mycket funktionell backup av din kod.
    Länk-elementet till ditt Github-repo finns redan på plats i index.html. Du måste uppdatera href-attributet.